

Comparative Analysis and Visualization of Genomic Sequences Using VISTA Browser and Associated Computational Tools

Inna Dubchak

Summary

This chapter discusses VISTA Browser and associated computational tools for analysis and visual exploration of genomic alignments. The availability of massive amounts of genomic data produced by sequencing centers stimulated active development of computational tools for analyzing sequences and complete genomes, including tools for comparative analysis. Among algorithmic and computational challenges of such analysis, i.e., efficient and fast alignment, decoding of evolutionary history, the search for functional elements in genomes, and others, visualization of comparative results is of great importance. Only interactive viewing and manipulation of data allow for its in-depth investigation by biologists.

We describe the rich capabilities of the interactive VISTA Browser with its extensions and modifications, and provide examples of the examination of alignments of DNA sequences and whole genomes, both eukaryotic and microbial. VISTA portal (<http://genome.lbl.gov/vista>) provides access to all these tools.

Key Words: Comparative genomics; alignment; visualization; genome browser; VISTA.

1. Introduction

Ongoing sequencing of a large number of prokaryotic and eukaryotic genomes provides biologists with invaluable datasets for investigating the evolution of individual species, differences and similarities between various species, and functional characteristics of genomes. Comparative analysis of genomes makes

an important contribution to solving these and many other problems (1–3). In most cases, this analysis is based on the alignment of genomic sequences followed by investigation of the level of conservation and the search for sequence signals specific to a particular genomic function. There are several approaches to each step of such studies, but regardless of the particular approach, there is a need to visualize the results of this comparative analysis.

Alignment is probably the most investigated area of computational biology, but it is still a subject of intensive work by many groups. There are several types of pair-wise alignments, i.e., global, local, or a combination of global and local, described in detail elsewhere (4). The availability of several assemblies of large genomes made possible the development of whole-genome alignment techniques (5,6), which generated a number of precomputed alignments that are available to the community. All techniques are unified by the common principles of finding the most similar genomic intervals (anchors) followed by extending these regions and chaining alignments to make them contiguous. The basepair level of visualization of alignments provides investigators with the most detailed comparative data, the same holds true for multiple alignments. At the larger scale, visual presentation of rearrangements, inversions, gap composition, and order of fragments of a draft sequence in the alignment are important for understanding the biology of a particular genomic interval.

One of the main purposes of comparative genomics is to provide a detailed analysis of conservation among orthologous intervals in different species. Defining which genomic intervals have been subject to negative (purifying) selection can bring us closer to understanding functions of different genomic elements. Methods for calculating conservation in alignments range from a simple window-based approach in PipMaker and VISTA (7,8) to the phylogenetic hidden Markov model Phastcons (9), to another statistical model, Gumby (10). Visualization of sequence conservation is a critical aspect of comparative sequence analysis because manual examination of alignment on the scale of long genomic regions is highly inefficient. This is why alignment-browsing systems are specifically designed to identify well-conserved segments. Different methods for calculating segments of conservation define the type of visual presentation, for example PIPMaker (7) represents the level of conservation in ungapped regions of BLASTZ local alignment as horizontal dashes; VISTA (8,11) and SynPlot (12) display comparative data in the form of a curve, where conservation is calculated in a sliding window of a gapped global alignment; PhastCons also generates a contiguous curve (9), and Gumby scores (10) are presented as the histogram-like Rank VISTA plot.

Internet-based genome browsers, emerging relatively recently, present the most essential tools for investigating genomic sequences because they integrate all sequence-based biological information on genes or genomic regions. They are easy to use and very efficient in retrieving large amount of relevant biological data. UCSC Browser ([13](#)), Ensembl ([14](#)), and MapView at National Center for Biotechnology Information ([15](#)) provide comprehensive data related to a number of vertebrate, invertebrate, and other genomes. In contrast, VISTA Browser is highly specialized and was built to show the results of comparative analysis of genomic sequences based on DNA alignments, both whole-genome and interval-based. Here, we present this computational tool with all the internal and external extensions and demonstrate its capabilities by analyzing several genomic intervals.

VISTA presentation of comparative data is easy to interpret both on a small and a large scale, i.e., at different levels of resolution. All VISTA programs and servers use the same type of visualization, making interpretation of alignments easy. Because VISTA tools are being constantly improved and enhanced, new options and capabilities can be found on the website. The VISTA support group (vista@lbl.gov) will help users explore these new options and answer questions.

2. VISTA Browser for Precomputed Whole-Genome Alignments

Whole-genome alignments accessible through VISTA Browser are based on the local/global approach developed in the group ([6,16,17](#)). These alignments are available for a number of vertebrates, invertebrates, plants, and others species. The list of whole-genomes alignments is constantly being updated by the VISTA group when new assemblies become available.

Results of VISTA comparative analysis are also available for a number of bacteria. Precomputed full scaffold alignments for microbial genomes are presented as a component of Integrated Microbial Genomes ([18](#)) developed in the Department of Energy's Joint Genome Institute, and are also available through the VISTA portal.

2.1. How to Access the Browser

As any other genome browser, VISTA Browser provides a view of a particular interval of a base (reference) genome. Thus, as the first step, the user needs to choose a genomic interval on the selected base genome.

Access the VISTA portal page online at <http://genome.lbl.gov//vista> and click the “VISTA Browser” link in the “Precomputed whole genome alignments” section, or use the direct link to the VISTA Browser gateway

(<http://pipeline.lbl.gov>). Detailed help pages are available online (<http://pipeline.lbl.gov/help.shtml>).

Select the “Base genome” from the pull-down menu on the left (**Fig. 1A**). Base genomes are identified by the name of a species and a date of assembly. After the Base genome is selected, a list of all available genome for this alignments will appear on the gateway page.

Define a position on the base genome. The user can input a position on a chromosome or a contig, as well as supply a gene name. The gene name should correspond to the annotation datasets used for a particular base genome. The gateway page describes which annotation are used for each base genome in the browser, i.e., RefSeq for human, mouse, and *Drosophila melanogaster*, FlyBase for *D. melanogaster*, TIGR annotation for rice, and others. An example of an input is shown in **Fig. 1A**, where *D. melanogaster* is selected as the Base genome, and an arbitrary interval, chr2L:816,000–828,000, is selected as the Position.

The user can choose either “VISTA Browser” or “VISTA tracks on UCSC Browser” as methods to view the results. Description of the differences between them will follow. VISTA Browser requires Java software to be installed on the computer (*see Note 1*).

If the user entered a chromosome/contig position or the name of a gene with a unique match, selecting “Go” will take the user directly to the browser. If a gene name is entered without a unique match, the user will be directed to a page that lists all entries that contain the search term.

2.2. VISTA Browser Display

The display consists of three main sections: a Control Panel on the left hand side, the central browser window(s), and a horizontal toolbar at the top. Here, we describe what these three sections consist of and how to use them.

2.2.1. How to Use “Control Panel” to Obtain a Desirable Display of a Genomic Region

Figure 1B–F illustrates the main functions of the Control Panel. **Figure 1B** displays the window that appears on the desktop of the computer when the browser is accessed through the gateway at <http://pipeline.lbl.gov> (*see above*). The conservation plot displayed on the right is based on the alignment of the base genome *D. melanogaster* with the genome of *Drosophila pseudoobscura* (the second species that is indicated below the plot on the right).

In the section with the five pull-down menus on the left, the name of the base genome can be seen, position on the genome, the annotation track used in

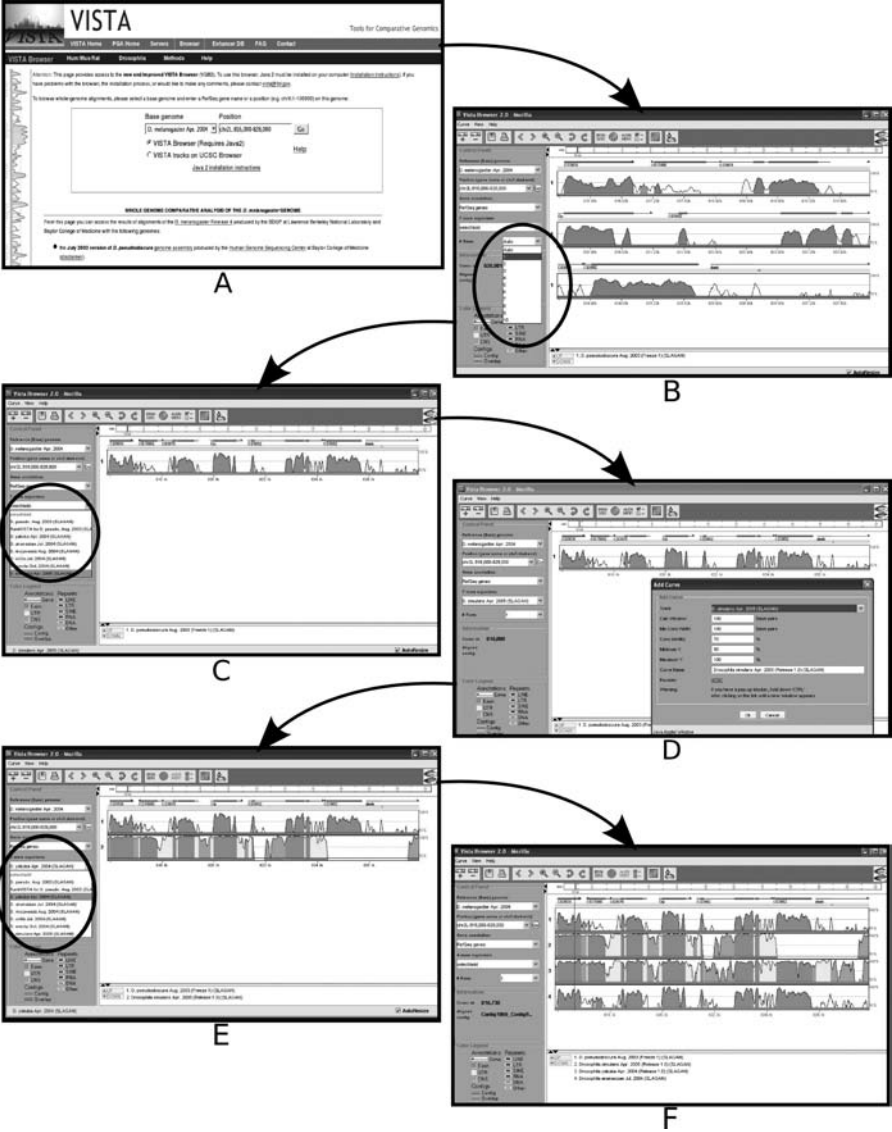


Fig. 1. Accessing VISTA Browser and using the control panel features. (A) Gateway to the browser, selecting a base genome and the interval of interest. (B) Changing the number of rows in the display through the “# rows” menu. (C) Adding a new alignment window through the “select/add” menu. (D) Selecting display parameters for this new alignment window. (E) Adding more alignment windows. (F) Display of 12 kilobasepair interval of the alignments of *D. melanogaster* with *D. simulans*, *D. yakuba*, and *D. ananassae*.

the display, and the number of rows in the plot display (“Auto” is a default). Each of these menus provides the user with a choice of options, for example, a user can replace the RefSeq annotation track with the FlyBase annotation track.

Selecting “1” as the number of rows (**Fig. 1B**) changes a three-row continuous view of the genomic interval to a one-row view (**Fig. 1C**). Next, the “select/add” menu allows the user to view what other alignments are available for the *D. melanogaster* genome. Selecting *Drosophila simulans* in this menu will open a small window that allows the user to choose display parameters (see **Note 2** on selecting display parameters) for the plot of the alignment of *D. melanogaster* and *D. simulans* (**Fig. 1D**). After changing the parameters or using the default parameters, clicking OK will cause the browser to display conservation for two alignments on the same interval of the base genome (**Fig. 1E**). **Figure 2F** shows the browser display after adding two more VISTA windows, the *D. yakuba* and *D. ananassae* alignments to the base genome.

Among the choices in the select/add menu, will be the RankVISTA plots for some of the alignments. Rank VISTA is an alternative way of scoring conservation in alignments that could be useful in some applications (**10**).

In the Information section on the left are the coordinates of the cursor on the base genome and the name of the chromosome or contig of the second species aligned in this position. This name displayed is for a selected plot (see **below** on how to select a plot), or for the default alignment if no plot is selected.

If the displayed genomic interval has masked repeats, the Color Legend box indicates how different kinds of repeats are displayed above the plot.

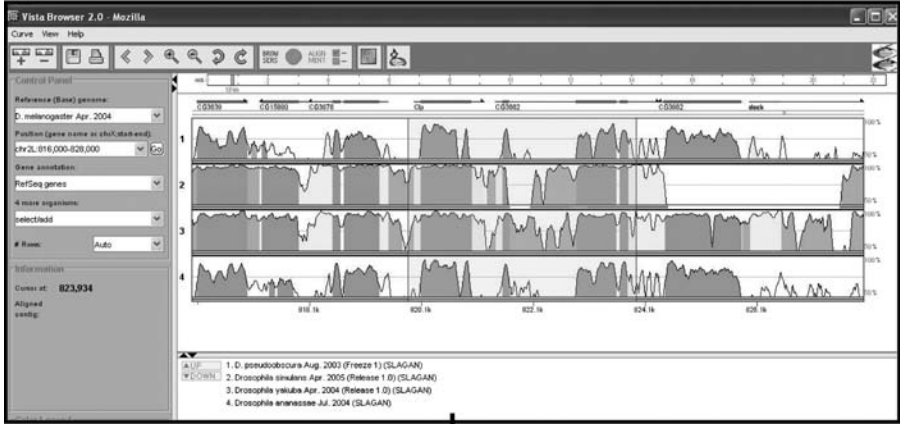
2.2.2. How to Interact With VISTA Tracks

The VISTA conservation window (for a pair-wise alignment) or several stacked windows (for several pair-wise alignments with the same genome as a base) occupy a central position in the Browser. Conservation is displayed in a standard VISTA format of peaks and valleys (see **Note 2**), and the height of each peak is indicative of the level of conservation in this area.

The horizontal bar on the top of the central section depicts the length of the entire chromosome and shows the location of the investigated interval on this chromosome.

Arrows on the top of the plots show the position and direction of genes, with their exonic intervals in blue and UTRs in turquoise, according to a selected annotation. Thus in VISTA plots, peaks depicting conserved sequences (CNSs) are blue if they are in exonic intervals of the base genome, turquoise if they overlap with UTR, or red for all unannotated sequences, i.e., intronic, intergenic, or without clear assignment.

A



B

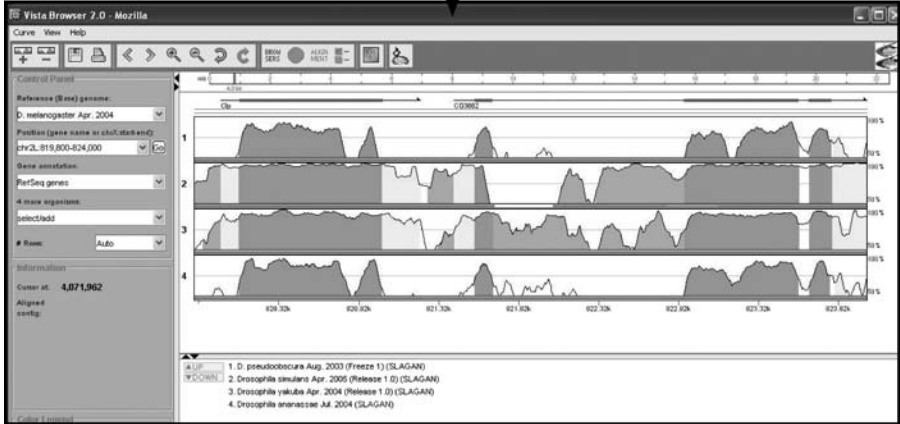


Fig. 2. VISTA Browser has a capability to zoom into the interval of interest by holding the left mouse button down (A). View of the 4.2-Kbp long genomic fragment of Chromosome 2L of *D. melanogaster* (B) is obtained by selecting a desired interval from the 12-Kbp sequence (A, shaded).

The bar below the plot is gray for continuous uninterrupted alignment, red where several intervals of the second genome are aligned to the same interval of the base genome (overlap, at chr2L:823,000–825,000 interval of *D. melanogaster*/*D. simulans* alignment) or where the alignment is interrupted (for example chr2L:824,200–826,500 interval in the same alignment).

Holding the left mouse button down and selecting an area on the base genome allows for zooming in on the interval of interest ([Fig. 2](#)).

Left-clicking any plot selects it, and that selection is necessary for a number of manipulations described next. Selected plots are shaded gray.

2.2.3. Browser Toolbar

Different control options are available either through the Toolbar, or a menu at the top of the Browser. Keeping the cursor over any of the buttons in the Toolbar shows a description of the option.

The buttons are:

Add VISTA Curve: works the same way as “select/add” menu in the Control Panel (**Subheading 2.2.2.**).

Remove VISTA Curve: one of the curves should be selected to use this option.

Save as: displays a window with a selection of formats (pdf, jpeg, or gif) for saving the plots to a file.

Print.

Scroll backwards and forward on the base genome.

Zoom in and out.

Return to previous and next position on the base genome.

Browsers: link to the same interval on the base genome displayed in the alternative browser(s). For some genomes, this button will bring up the UCSC browser with additional VISTA curves/control options ([Fig. 3](#)).

Relevant browsers also include the JGI browser for a number of species, RGD for the rat genome, and others.

To use the following three buttons it is necessary to select one of the plots: Alignment details (1): gives access to a page with detailed comparative information, also referred to as “Text Browser.”

Alignment: shortcut to a text file with an alignment.

Curve parameters: opens a window for changing conservation parameters used for building the VISTA plot, the same as the window in [Fig. 1D](#).

Right-clicking on the curve opens a selection window that gives access to some of the options of the Toolbar (Details, Parameters, Alignment, Add/Remove), with an additional option of changing the base genome.

2.2.4. Text Browser

This page links the alignments to other sequence-based information. The user will find the coordinates of conserved regions, their sequences, annotations, and other available data. [Figure 4](#) shows the most basic set of options in the “Text

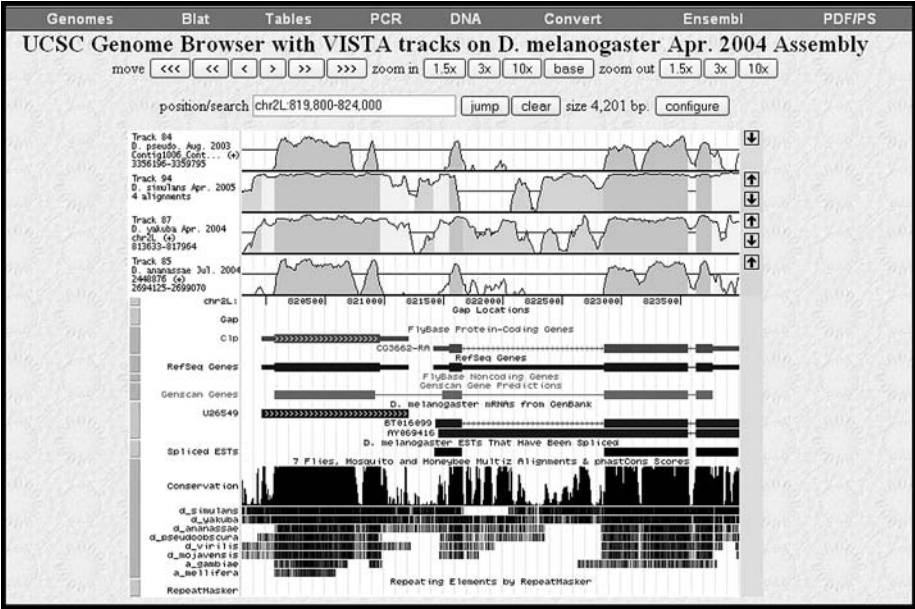


Fig. 3. VISTA Tracks, accessible through the VISTA Browser, display results of VISTA comparative analysis in the context of the whole genome annotation on the mirrored UCSC *D. melanogaster* browser.

Browser,” obtained from the VISTA plot of *D. melanogaster* vs *D. ananassae* (Fig. 1F). The names of participating genomes as well as the program used for the alignment are shown in the top banner.

Below the banner are the coordinates of the currently displayed region and a link back to VISTA Browser, an alternative browser (VISTA Tracks on UCSC in this case), and a pull-down menu with a choice of annotation.

Links in the next row give access to the coordinates of annotated genes in the interval, as well as the coordinates of CNSs. The user will notice that when the conserved regions are displayed, their lengths are actually web links. Clicking on the links will bring up the conserved sequences from both of the participating organisms.

In the main table listed next, each alignment generated for the base organism is displayed. Columns, except for the last one, refer to the sequences that participate in the alignment. The last column contains detailed information on the whole alignment.

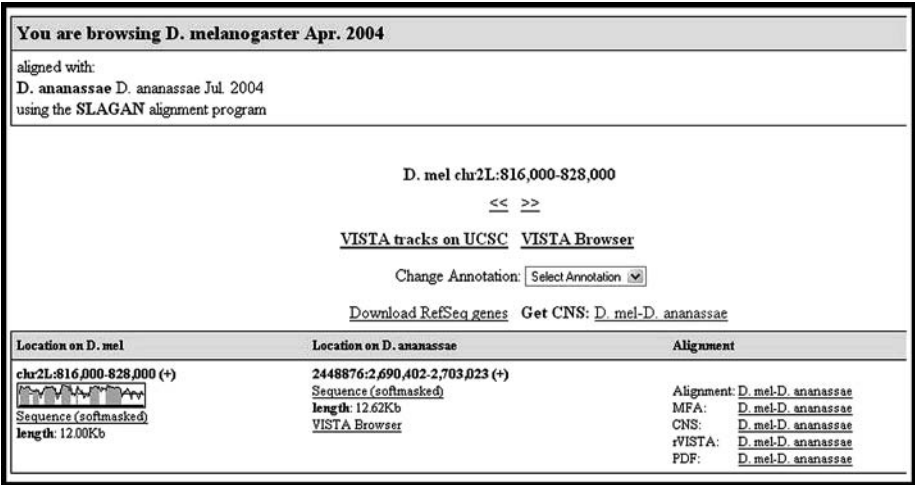


Fig. 4. Detailed information display (“Text Browser”) provides access to the data underlying the VISTA graph of the genomic interval chr2L:816-828000 of *D. melanogaster* aligned with *D. ananassae*.

Each row is a separate alignment, and displays pairs of genomic intervals of the two organisms participating in this alignment. Presence of only one row in Fig. 4 shows the most straightforward case of unambiguous pair-wise alignment. More complicated cases are described in Subheading 2.2.5.

The first cell of each row contains a small image of the VISTA plot of this alignment, which is helpful when several alignments are compared for an interval and the user wants to evaluate relative quality of those including alignment overlaps. “Sequence” links to a FASTA-formatted DNA segment that participates in the alignment. Clicking on the “VISTA Browser” link will launch the browser with the associated species as the base.

The last column provides links to the alignments in different formats, a list of conserved regions from this alignment, and links to static pdf-formatted plots of this alignment.

2.2.5. Additional VISTA Browser and Text Browser Features for Special Cases of Alignment

Text Browser design allows for flexibility in presenting information relevant to participating sequences and their alignment. Next are several special cases:

1. When the Shuffle-Lagan program is used for comparing user-submitted sequences or microbial genomes, there will be a link to dot-plots of the alignments produced.

2. When several intervals of a second species are aligned to a particular interval of the base genome with or without overlap (*see Subheading 2.2.2.*), the first column will display several VISTA pictures for each subinterval of the alignment.
3. In case of a multiple alignment, there will be more than one column with the data on the aligned to the base genome species. Each column will provide details on a particular organism.
4. If the examined region of the base genome is shorter than 20 kb, Text Browser will provide a rVISTA (Regulatory VISTA, *see Subheading 3.*) link to start this analysis.
5. If the examined region is long enough for the Rank VISTA evaluation of conservation, the link to this tool will be found in Text Browser.

If Text Browser displays new links not described in this chapter, Help pages will provide detailed description of these modules.

3. VISTA Services for User-Submitted Sequences

VISTA Browser has been built to visualize alignments of any length, thus in addition to displaying comparison of the whole genomes it is used for comparative analysis of user-submitted sequences. VISTA portal (<http://genome.lbl.gov/vista>) offers a choice of several automatic servers described briefly next. More details on the VISTA servers are available in our previous publications, for example in **ref. 8**. VISTA pages also provide extensive help on selecting a type of analysis and finding optimal parameters for a particular project.

In Genome VISTA, a single sequence (draft or finished) is compared with whole genome assemblies. For a submitted sequence, the server finds candidate orthologous regions on the base genome, and provides detailed comparative analysis.

mVISTA is designed to perform pair-wise or multiple alignments of DNA sequences from two or more species up to megabases long and to visualize these alignments together with their annotations. Depending on the project, a user can choose one of the three alignment programs: AVID (**19**) for global pair-wise and multiple pair-wise alignment (one of the sequences can be in a draft format), LAGAN (**20**) for global pair-wise and multiple alignment of finished sequences, or Shuffle-LAGAN (**16**) for global alignment with synchronized detection of rearrangements and inversions.

rVISTA (regulatory Vista) (**21**) combines searching the major transcription factor binding site database TRANSFACTM Professional from Biobase (**22**) with a comparative sequence analysis. It can be used directly or through links in mVISTA, Genome VISTA, or VISTA Browser.

Phylo-VISTA (23) allows a user to visualize submitted multiple sequence alignment data while taking the phylogenetic relationships between sequences into account.

4. Notes

1. How to install Java. VISTA Help section provides a detailed instruction on this installation (http://pipeline.lbl.gov/vgb2/help/java_win_instructions.shtml). The latest version of J2SE from the Java download page of Sun Developer Network will be needed (<http://java.sun.com/j2se/1.4.2/download.html>).
2. How VISTA curves are calculated. The Vista curve is calculated as a windowed-average identity score for the alignment. A variable sized window (Calc Window) is slid across the alignment and a score is calculated at each base in the coordinate sequence. That is, if the Calc Window is 100 bp, then the score for every point X is the percentage of exact matches between the two alignments in a 100-bp wide window centered on that point X. Because of resolution constraints when visualizing large alignments, it is often necessary to condense information about 100 or more basepairs into one display pixel. This is done by only graphing the maximal score of all the basepairs covered by that pixel.
3. How to choose display parameters. The parameters selected for visualization of alignments have a significant effect on the VISTA results. A user can vary the following parameters (**Fig. 1D**): (1) a window for calculating the VISTA curve (Calc Window); (2) window size for finding CNSs (Min Cons Width); (3) percent of identical nucleotides in the window for finding CNSs (Cons Identity); (4) minimum level of Cons Identity shown on the plot (Minimum Y); (5) maximum level of Cons Identity shown on the plot (Maximum Y). Parameter (1) defines smoothness of the plot, selection of parameters (2) and (3) depends on the similarity of compared sequences. The default parameters of 100 bp for a window and 70% for similarity normally need to be reduced for distant species with lower level of conservation, and increased for higher than human/mouse similarity. Generally it takes several trials to retrieve CNSs with meaningful level of conservation. In many cases, precomputed Rank-VISTA provides an additional list of highly conserved elements calculated by a different technique. Rank-VISTA parameters are also adjustable, and their description can be found in the Help section.

Acknowledgments

The author is grateful to Michael Cipriano and Alexander Levin for their help with the manuscript. The VISTA project is an ongoing collaborative effort of a large group of scientists and engineers. It has been developed and maintained in the Genomics Division of Lawrence Berkeley National Laboratory. The names of all contributors are found at the VISTA website (<http://genome.lbl.gov/vista>).

The project was partially supported by the grant no. HL88728, Berkeley-PGA, under the Programs for Genomic Application, funded by the US National

Heart, Lung, and Blood Institute, and performed under Department of Energy Contract DE-AC0378SF00098, University of California.

References

- 1 Miller, W., Makova, K. D., Nekrutenko, A., and Hardison, R. C. (2004) Comparative genomics. *Annu. Rev. Genomics Hum. Genet.* **5**, 15–56.
- 2 Hardison, R. C. (2003) Comparative genomics. *PLoS Biol.* **1**, 156–160
- 3 Ureta-Vidal, A. Ettwiller, L., and Birney, E. (2003) Comparative genomics: genome-wide analysis in metazoan eukaryotes. *Nat. Rev. Genet.* **4**, 251–262.
- 4 Pollard, D. A., Bergman, C. M., Stoye, J., Celniker, S. E., and Eisen, M. B. (2004) Benchmarking tools for the alignment of functional noncoding DNA. *BMC Bioinformatics* **5**, 6–22.
- 5 Schwartz, S., Kent, W.J., Smit, A., et al. (2003) Human-mouse alignments with BLASTZ. *Genome Res.*, **13**, 103–107.
- 6 Couronne, O., Poliakov, A., Bray, N., et al. (2002) Strategies and tools for whole genome alignments. *Genome Res.* **13**, 73–80.
- 7 Schwartz, S., Elnitski, L., Li, M., et al., and NISC Comparative Sequencing Program. (2003) MultiPipMaker and supporting tools: alignments and analysis of multiple genomic DNA sequences. *Nucleic Acids Res.* **31**, 3518–3524.
- 8 Frazer, K. A., Pachter, L., Poliakov, A., Rubin, E. M., and Dubchak, I. (2004) VISTA: computational tools for comparative genomics. *Nucleic Acids Res.* **32**, W273–W279.
- 9 Siepel, A., Bejerano, G., Pedersen, J. S., et al. (2005) Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res.* **15**, 1034–1050.
- 10 Ahituv, N., Prabhakar, S., Poulin, F., Rubin, E. M., and Couronne, O. (2005) Mapping cis-regulatory domains in the human genome using multi-species conservation of synteny. *Hum. Mol. Genet.* **14**, 3057–3063.
- 11 Mayor, C., Brudno, M., Schwartz, J. R., et al. (2000) VISTA: visualizing global DNA sequence alignments of arbitrary length. *Bioinformatics* **16**, 1046–1047.
- 12 Chapman, M. A., Donaldson, I. J., Gilbert, J., et al. (2004) Analysis of multiple genomic sequence alignments: a web resource, online tools, and lessons learned from analysis of mammalian SCL loci. *Genome Res.* **14**, 313–318.
- 13 Kent, W. J., Sugnet, C. W., Furey, T. S., et al. (2002) The human genome browser at UCSC. *Genome Res.* **12**, 996–1006.
- 14 Birney, E., Andrews, D., Caccamo, M., et al. (2006) Ensembl 2006. *Nucleic Acids Res.* **34**, D556–D561.
- 15 Wheeler, D. L., Church, D. M., Lash, A. E., et al. (2001) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.* **29**, 11–16.
- 16 Brudno, M., Malde, S., Poliakov, A., et al. (2003) Glocal alignment: finding rearrangements during alignment. *Bioinformatics* **Suppl 1**, I54–I62.
- 17 Brudno, M., Poliakov, A., Salamov, A., et al. (2004) Automated whole-genome multiple alignment of rat, mouse, and human. *Genome Res.* **14**, 685–692.

- 18 Markowitz, V. M., Korzeniewski, F., Palaniappan, K., et al. (2006) The integrated microbial genomes (IMG) system. *Nucleic Acids Res.* **34**, D344–D348.
- 19 Bray, N., Dubchak, I., and Pachter, L. (2003) AVID: a global alignment program. *Genome Res.* **13**, 97–102.
- 20 Brudno, M., Do, C. B., Cooper, G.M., et al., and NISC Comparative Sequencing Program. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 21 Loots, G., Ovcharenko, I., Pachter, L., Dubchak, I., and Rubin, E. (2002) rVISTA for comparative sequence-based discovery of functional transcription factor binding sites. *Genome Res.* **12**, 832–839.
- 22 Matys, V., Kel-Margoulis, O.V., Fricke, E., et al. (2006) TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**, D108–D110.
- 23 Shah, N., Couronne, O., Pennacchio, L. A., et al. (2004) Phylo-VISTA: interactive visualization of multiple DNA sequence alignments. *Bioinformatics* **20**, 636–643.

Comparative Genomic Analysis Using the UCSC Genome Browser

Donna Karolchik, Gill Bejerano, Angie S. Hinrichs, Robert M. Kuhn, Webb Miller, Kate R. Rosenbloom, Ann S. Zweig, David Haussler, and W. James Kent

Summary

Comparative analysis of DNA sequence from multiple species can provide insights into the function and evolutionary processes that shape genomes. The University of California Santa Cruz (UCSC) Genome Bioinformatics group has developed several tools and methodologies in its study of comparative genomics, many of which have been incorporated into the UCSC Genome Browser (<http://genome.ucsc.edu>), an easy-to-use online tool for browsing genomic data and aligned annotation “tracks” in a single window. The comparative genomics annotations in the browser include pairwise alignments, which aid in the identification of orthologous regions between species, and conservation tracks that show measures of evolutionary conservation among sets of multiply aligned species, highlighting regions of the genome that may be functionally important. A related tool, the UCSC Table Browser, provides a simple interface for querying, analyzing, and downloading the data underlying the Genome Browser annotation tracks. Here, we describe a procedure for examining a genomic region of interest in the Genome Browser, analyzing characteristics of the region, filtering the data, and downloading data sets for further study.

Key Words: Comparative genomics; UCSC Genome Browser; UCSC Table Browser; cross-species alignments; evolutionary conservation; orthology.

1. Introduction

As the variety of sequenced genomes available in the public domain continues to grow, increasing attention is being paid to the analysis of conservation patterns between species to identify shared functional elements, which

stand out as having diverged less than surrounding sequence. The University of California Santa Cruz (UCSC) Genome Bioinformatics group has played a significant role in the comparative analyses of vertebrate genomes, beginning with the initial draft assembly of the mouse genome, in which it was discovered that 5% of the human genome, most of it nonprotein coding DNA, is under negative selection (1–3).

We have integrated the basic tools and methodologies developed for these types of investigations into the UCSC Genome Browser (4,5), where they are freely available to the worldwide scientific community. These tools have proven to be valuable to scientific investigators for obtaining and analyzing conserved regions from a variety of organisms (6–12).

The UCSC Genome Browser (<http://genome.ucsc.edu>) (Fig. 1) is a popular web-based tool that provides a simple, intuitive interface for quickly finding and viewing a section of genome sequence and an extensive set of annotation “tracks,” enabling rapid visual analysis and correlation of the data. The Genome Browser database (13) contains data for dozens of species, including several key model organisms (Table 1). The annotation set, which contains data generated by both UCSC and external collaborators, encompasses a large variety of gene prediction, gene regulation, expression, and comparative genomics data. The underlying data may also be queried and downloaded as text using the UCSC Table Browser (14). More advanced users can upload their own data sets into the browser using the custom annotation tracks feature or download selected data for analysis in their local computing environment.

The tracks in the Genome Browser’s Comparative Genomics annotation group are particularly valuable when comparing the genomic characteristics of different species. The chain and net pairwise alignment tracks (15,16) may be used to look for orthologous regions between organisms, large-scale rearrangements, duplications and deletions, and processed pseudogenes; the chains can also be used to examine paralogs. The net data serve as input to the multiple alignments (17) that form the basis of the Conservation track. This annotation displays a measure of evolutionary conservation among a set of species based on a phylogenetic hidden Markov model approach, phastCons (11), highlighting regions of the genome that may be functionally important. The Most Conserved track, present on selected genome assemblies, provides a simplified view of the Conservation track, emphasizing the parts of the genome most likely conserved by purifying selection.

The comparative genomics annotations in the Genome Browser are continually maturing as new species are added and the annotation algorithms are refined. Initial versions of the human Conservation track were based on the

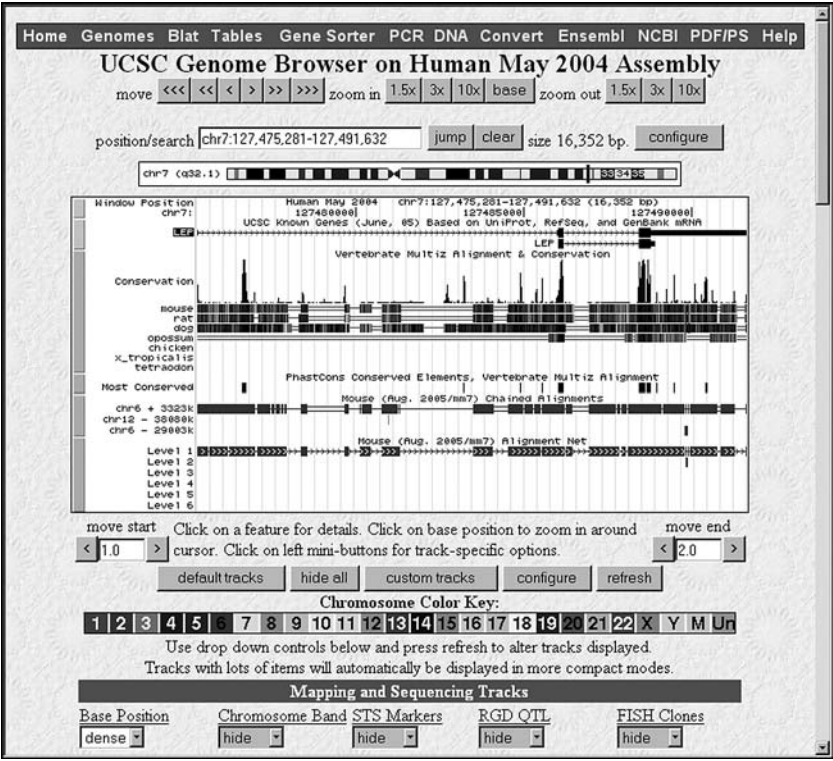


Fig. 1. The UCSC Genome Browser displaying the region of the LEP gene on the May 2004 human genome assembly. The annotation tracks image, central to the display, shows a collection of annotation data sets aligned to the reference sequence at the positions indicated at the top of the image. Two variants of the gene are displayed in the UCSC Known Genes track, labeled “LEP” to the left of the features. The taller blocks represent the coding exons, the attached half-height blocks indicate the 5’ and 3’ UTR, and the arrowed lines connecting the blocks show introns. The Mouse Chained Alignments track shows aligning regions of the August 2005 mouse genome assembly; the Mouse Alignment Net track organizes the best-scoring chains and categorizes them by level. The Conservation track shows pairwise alignments of seven species to the human genome (bottom) and a histogram indicating a combined measure of evolutionary conservation in the species shown. The most highly conserved regions are highlighted in the Most Conserved track. The groups of pull-down menus at the bottom of the figure (partially shown) control the display settings for each track. Navigation and configuration controls above and below the image allow easy maneuvering and customization of the display. The chromosome color key indicates the chromosome location of alignments from other species in the comparative genomics tracks.

Table 1
Genome Assembly Data Available in the UCSC Genome Browser Database in Early 2006

Clade	Organism	Genome browser assemblies
Vertebrate	Human	3 available, 12 archived
	Chimp	2 available
	Rhesus macaque	2 available
	Dog	2 available
	Cow	2 available
	Mouse	2 available, 6 archived
	Rat	2 available, 2 archived
	Opossum	1 available
	Chicken	1 available
	Frog (<i>Xenopus tropicalis</i>)	1 available
	Zebrafish	2 available, 1 archived
	Tetraodon	1 available
	Fugu	1 available
Deuterostome	<i>Ciona intestinalis</i>	2 available
	<i>Strongylocentrotus purpuratus</i>	1 available
Insect	Drosophila	11 different species available
	Honey bee	1 available
	<i>Anopheles gambiae</i>	2 available
Nematode	<i>Caenorhabditis elegans</i>	2 available
	<i>Caenorhabditis briggsae</i>	1 available
Other	Yeast (<i>Saccharomyces cerevisiae</i>)	1 available

multiple alignment of 3 species; this has grown to 17 species in early 2006 (**Fig. 2**), and will undoubtedly continue to expand as more sequenced genomes become available.

In this chapter, we present an overview of the UCSC Genome Browser and explain its use in viewing, analyzing, filtering, and downloading areas of comparative genomics interest using the Genome Browser tool suite. We examine regions of orthology between two species, using the human and mouse genomes as an example, and areas of possible conservation within a larger set of species. We then use the Table Browser to construct a set of conservation scores and download it for further analysis, exploring two techniques for filtering data sets. We also describe how to incorporate customized data sets into the analysis.

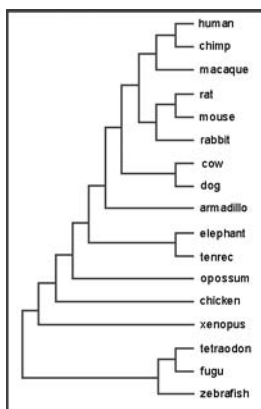


Fig. 2. Multiple alignment pairings underlying a Conservation track based on 17 species.

2. Materials

The UCSC Genome Browser can be accessed by any Internet browser that supports JavaScript, running on a computer with access to the Internet.

3. Methods

The methods described in this procedure use the human genome assembly as the reference sequence; however, these techniques can be applied to most of the vertebrate assemblies and several of the invertebrate genomes included in the Genome Browser database. The Genome Browser software and data are constantly evolving; therefore, slight differences may be noted between the methods described next and the actual online software. If the user is unable to perform any of the methods or has questions about a technique, contact us at genome@soe.ucsc.edu. Additional information is available through the Help, FAQ, Training, and Contact Us links on the UCSC Genome Bioinformatics homepage (<http://genome.ucsc.edu>).

3.1. Open the UCSC Genome Browser to a Specified Region

1. Open the UCSC Genome Bioinformatics homepage (<http://genome.ucsc.edu>) in an Internet browser. This page offers links to a wide variety of genome-browsing tools and information (*see* **Note 1**).
2. Select the “Genome Browser” option from the menu in the left-hand sidebar.
3. On the Gateway page, select the clade, genome, and assembly of interest. The following methods use the Human May 2004 (hg17) genome assembly.

4. Type one or more search terms or a genomic position in the position or search term box, then click the submit button (*see Note 2* for a description of legitimate search terms). For this procedure, we use the gene symbol “LEP.” The Gateway displays a page listing items in the database that match the search criteria and links to the corresponding coordinate locations on the reference sequence. In some instances, only a single match is found; in these cases, the Genome Browser will open directly and **step 5** may be skipped.
5. Click the link to the item of interest; in this example, we use the first Known Genes link, LEP (NM_000230). The Genome Browser displays a graphical image showing a set of annotation tracks aligned to the reference genome coordinates specified in the query, together with controls to navigate through the sequence, configure the image display and fine-tune the graphical display of specific tracks (**Fig. 1**) (*see Note 3*). The reference coordinates are shown in the Base Position track at the top of the image, also referred to as the “ruler.” The menu bar at the top of the page provides easy access to the same genomic region in other UCSC tools (the Blat, Tables, Gene Sorter, and PCR links), as well as links to other genome-browsing tools (Ensembl, National Center for Biotechnology Information), a DNA sequence-retrieval utility (DNA), a coordinate conversion utility (Convert), and a utility that prints a high-quality PDF or postscript image of the annotation tracks (PDF/PS).

3.2. Browse the Reference Sequence and Configure the Display

1. Click the zoom in and zoom out buttons to expand or reduce the displayed coordinate range 1.5-, 3-, or 10-fold. The move buttons shift the coordinates in the indicated direction by 10, 50, or 95% of the displayed size. To scroll the image left or right while keeping the position of the opposite end static, click the move start or move end arrows; the amount of scrolling can be increased or decreased by editing the number in the text box. Quickly change the displayed genomic region by typing a new search term into the position/search box, then clicking the jump button. *See Note 4* for navigation shortcuts.
2. Each assembly in the Genome Browser contains many annotation tracks that are hidden by default in the graphical image because of space constraints. Tracks are clustered into groups that reflect the primary focus of the data. The track controls section at the bottom of the page shows a complete set of the annotation groups and tracks available in the selected coordinate range. To change the display mode of a track, choose the desired setting on the track control’s display menu, then click the refresh button to display the changes in the graphical image (*see Note 5*).
3. Click the configure button to change display characteristics, such as the image width and the text size in the graphical image, and to hide or show groups of annotation tracks, the track control section, the chromosome ideogram, and image labels (*see Note 6*). Click the submit button to apply the changes to the browser session. Modifications made on the configuration page are retained in future sessions on the same Internet browser until they are reset.

4. Click the default tracks button to restore the default track settings.

3.3. Examine Pairwise Alignments for Evidence of Orthology

1. Find the pull-down display menus for the Mouse Chain and Mouse Net tracks in the Comparative Genomics track controls group. Within this section, the chain and net tracks are displayed in order of least-to-most similarity to the current genome (see **Note 7**). Change the Mouse Chain and Mouse Net display settings to “full,” then click the refresh button to display the expanded tracks in the browser (**Fig. 1**).

The Mouse Chain track shows chains of alignment blocks depicting genomic regions potentially derived from the same sequence in the common ancestor, joined by either a single line, indicating a gap most likely due to a deletion in the aligning sequence or an insertion in the reference sequence, or double lines, representing locations where there is intervening DNA in both human and mouse that cannot be aligned well. The aligned blocks in a chain are shown in the same order and orientation in both the human and mouse genome. It is not uncommon for such a chain of alignment blocks to extend for many megabases, providing very strong evidence that the human and mouse regions evolved from the same segment in the genome of the common ancestor of the two species, i.e., that they are orthologous. Multiple overlapping chains represent paralogs in the aligning species for this region. These are often the result of tandem, segmental, or retrotranspositional duplications.

The Mouse Net track organizes multiple overlapping chains and categorizes them by level. Level 1 indicates the highest-scoring chains spanning the region; these most likely represent the orthologous region in mouse. In cases where a gap exists in the top-level chain, it is filled (if possible) by a level 2 chain, and so on. Some of these may also represent orthologous regions, e.g., in the case of the likely inversion shown in **Fig. 3**. In a color display, the color of a chain indicates the chromosomal source of the aligning sequence, as listed in the chromosome color key below the annotation image.

2. Click the mini-button to the left of each track in the graphical display to view information about the track, including a description of the track data, the methods used to generate the data, display conventions, information about the track’s contributors, and selected references (see **Note 8**). For some tracks this page also presents options for fine-tuning the display. Click the Genome Browser link to return to the main Genome Browser page.
3. Click on an area of the Mouse Chain track to view detailed information about the chained alignments. Note that most of the alignment information, with the exception of the “Approximate score within browser window” value, refers to the entire chain or gap, not just the portion displayed in the window. To view the entire chain or gap in the Mouse browser, click the “Mouse position” link; to examine only the portion of the alignment displayed in the Human browser image, click the “Open Mouse browser” link. The “View details of parts of chain within browser window”

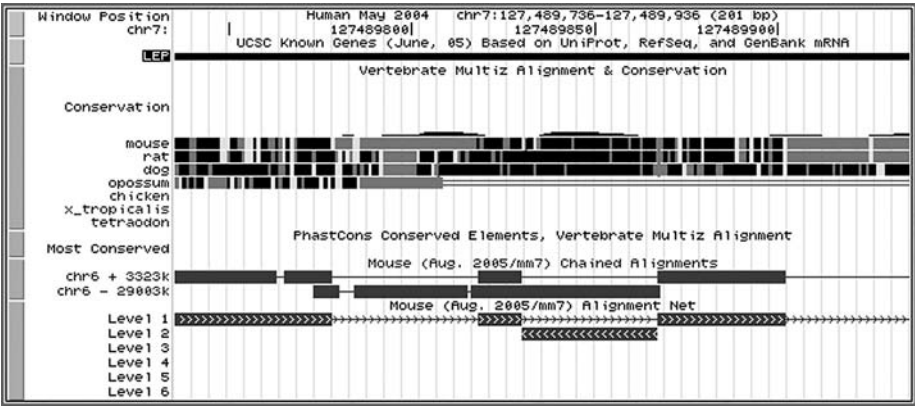


Fig. 3. A zoomed-in look at the chain and net tracks in Fig. 1, showing the subregion chr7:127,489,736-127,489,936 of the May 2004 human genome assembly. A gap in the top-level chain has been filled in by an inverted chain at Level 2, which may also represent an orthologous region.

link shows a base-level representation of the pairwise alignment, including a base-by-base comparison between the human and mouse assemblies. The “View table schema” link displays the MySQL structure and sample data records of the primary table underlying the annotation. Click the Genome Browser link to return to the main Genome Browser page.

- Click on the highest-scoring chain at level one of the Mouse Net track, then click the “Open Mouse browser” link. This displays the region in the mouse genome that is most likely to be orthologous to the region displayed in the human Genome Browser (Fig. 4). Click on a gap (line) within the Mouse Net track to view information that may be useful in determining the cause of the gap or, for lower level chains, the genomic rearrangement.
- To find further supporting evidence for a region of apparent orthology, it may be useful to examine other Genome Browser tracks. For example, the human genome and many of the model organisms have a Known Genes track (18), an annotation that shows known protein-coding genes and homologous genes in other species. To display this track, find the pull-down display menu for the Known Genes track (if available) in the Genes and Gene Prediction Tracks track controls section and change the display setting to “pack” or “full,” then click the refresh button. Click on an individual gene in the track to display detailed information about the gene, then click the “Other Species” link (if present) in the table at the top of the page. The homologous genes in this section are based on protein rather than DNA alignments. Browsers for many nonhuman species also contain a Human Proteins track that shows the best mapping, based on a translated alignment, of each human Known Gene to the nonhuman species.

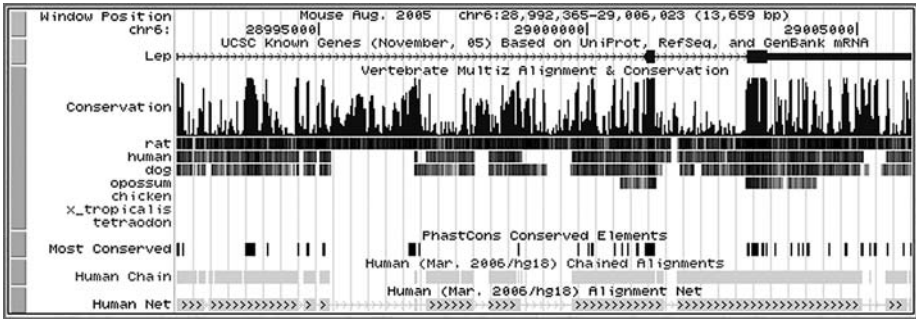


Fig. 4. Region in the Aug. 2005 mouse genome that is most likely orthologous to the human genome region displayed in Fig. 1. This image was obtained by clicking on the top-level chain in the Mouse Net track, then clicking the Open Mouse browser link on the track details page.

3.4. Examine Evolutionary Conservation Among Multiple Species

1. Find the pull-down display menu for the Conservation track in the Comparative Genomics track controls section. By default, the track display should be set to “pack” mode; if not, change the mode and click the refresh button. The Conservation track shows a measure of evolutionary conservation among the displayed species, highlighting putative functional regions of the genome. Genomic elements that are very conserved between distant species may indicate strong negative selection for function, although there is no simple correlation between conservation and function.

The Conservation track is comprised of two parts (Fig. 1). The bottom section displays pairwise alignments of numerous species to the reference sequence. Darker areas reflect regions in which the aligned basepair matches the reference sequence; gaps denote areas where no alignment was found. Note the correspondence with the net tracks, which were used to generate the pairwise inputs to the multiple alignment on which this track is based. The top section of the Conservation track shows a combined measure of evolutionary conservation in the species shown, based on scores assigned by the phastCons phylogenetic hidden Markov model (11) to multiple alignments generated by multiz (17).

2. Click the mini-button to the left of the Conservation track to open the track’s description page. This annotation track has a large number of configurable display options (see Note 9). To apply configuration changes and return to the main Genome Browser page, click the Submit button; otherwise, click the Genome Browser link.
3. Click on a region in the Conservation track to view detailed information about the currently displayed region, including base-level depictions of the multiple-species alignments displayed in the annotation tracks image (see Note 10). Click the Genome Browser link to return to the main graphical display.

4. Find the pull-down display menu for the Most Conserved track in the Comparative Genomics track controls section and change the display setting to “dense,” then click the refresh button. The Most Conserved track shows predictions of discrete conserved elements in the reference sequence. Conserved elements are defined using a two-state hidden Markov model and are scored for the probability of conservation against a null model of neutral evolution. Higher scores indicate a greater likelihood of conservation.
5. The Most Conserved track can be filtered to show only those scores that meet or exceed a threshold. To set a minimum threshold for the displayed data, specify a minimum score (e.g., 500) in the filter at the top of the track description page, then click the Submit button. Using a threshold to screen scores may point out some spurious scores resulting from DNA contaminants present in the aligning sequences. The chains and net tracks may also be used to visually inspect for contaminating sequence.
6. Click on an element in the Most Conserved track to view detailed information about the element, including its raw logarithmic odds (lod) score and a transformed lod score between 0 and 1000 ([11](#)). The details page also lists the scores and positions of the top-scoring elements in the currently displayed window. Click the Genome Browser link to return to the main graphical display.

3.5. Download Conservation Scores Using the Table Browser

1. On the main Genome Browser page, click the Tables link on the top menu bar to open the Table Browser, a powerful, flexible tool for querying, analyzing, and downloading the data underlying the Genome Browser annotation tracks ([Fig. 5](#)). By default, the Table Browser is automatically set to the organism, assembly, and genomic region currently displayed in the Genome Browser.
2. The group and track pull-down menus list the same set of annotation groups and tracks displayed in the Genome Browser for the selected assembly. For this example, choose the “Comparative Genomics” option in the group menu, the “Conservation” option in the track menu, and the “phastCons17way” option in the table menu (*see Note 11*).
3. The region setting defines the scope of a Table Browser query: genome-wide, the ENCODE regions ([19](#)), chromosome-wide, or a specific region within a chromosome. Click the “region: position” button to limit the query to the genomic range specified in the position box. By default, the position is set to the coordinate range last accessed by an application in the Genome Browser suite. To choose a different position, type in a search or position term, e.g., “lep,” then click the “lookup” button to convert the term into a coordinate range (*see Note 12*). A link may be selected from a list of several choices, as described in **Subheading 3.1., step 5**.

Fig. 5. The UCSC Table Browser, set up to display score data from the Conservation track. Click the Help link in the top menu bar to view the Table Browser User's Guide. A brief summary of the Table Browser controls can be found at the bottom of the page (not shown).

4. Select the “data points” option in the output format menu, then click the “get output” button (see **Note 13**). The Table Browser displays the conservation scores for each base in the selected region of the reference sequence. To save these data to a file, type a file name into the output file text box and select the desired file type returned option prior to running the query. Click the Tables link to return to the main Table Browser page.
5. The multiple alignments underlying the Conservation track may also be viewed in the Table Browser. Select the group and track options, as described in **Subheading 3.5., step 2**, then select the table name beginning with “multiz” (for example, “multiz17way” in the May 2004 human genome assembly). Select the “MAF—multiple alignment format” output format, then click the “get output” button. The Table Browser displays the multiple alignment sequences composing the currently selected region in the Conservation track, similar to the multiple-species alignment information displayed by the Genome Browser in **Subheading 3.4., step 3**.

3.6. Filter Data Using a Minimum Threshold and Save to a Custom Track

1. On the main Table Browser page, retain the “Comparative Genomics” group setting; select the “Most Conserved” option in the track menu and the “phastConsElements” option in the table menu.
2. Click the “describe table schema” button to view the structure of the MySQL table in which the phastConsElements data are stored in the Genome Browser database, as well as sample data records and a description of the associated Genome Browser track (see **Note 14**). Click the Tables link to return to the main Table Browser page.
3. Select the query region as described in **Subheading 3.5., step 3**.
4. Click the “filter: create” button to display a list of the fields and filter options available for the phastConsElements table. To set up a filter that returns only those records that meet or exceed a minimum transformed lod score, select the “>=” option from the pull-down menu to the right of the “score” field, then type in a score between 0 and 1000 (e.g., 500). This sets a minimum threshold for the score data, similar to the Genome Browser filter set up in **Subheading 3.4., step 5**. Click the submit button to activate the filter and return to the main Table Browser page (see **Note 15**).
5. Click the summary/statistics button to display a profile of the table items that match the current query. Analysis of these statistics can be used to fine-tune the filter criteria to increase or decrease the number of matches. Click the Tables link to return to the Table Browser main page.
6. Choose the “custom track” option in the output format menu. Custom annotation tracks are a convenient way to save the results of a query for future use in the Table Browser or to load a customized data set from the user’s research into the browser for viewing and analysis (see **Note 16**).
7. Click the get output button. The Table Browser presents options for configuring the custom track label and display settings. Edit the track display information as desired; retain the default “Whole Gene” setting for this example. Click the “get custom track in table browser” button to load the custom track into the current Table Browser session (see **Note 17**). If no records match the query criteria, the Table Browser displays a message to this effect; in such a case, the filter may be modified to refine the query results by clicking the “filter: edit” button, making the desired changes, then resubmitting the query.
8. To view the data saved in the loaded custom track, select the “Custom Tracks” option from the top of the group menu on the main Table Browser page. Select the newly created custom track and table from the track and table menus. Select the “all fields from selected table” option, erase the file name (if present) in the output file box, then click “get output”. Note that, as expected, all the conservation scores in the custom track exceed the threshold set in the filter in **step 4**.

3.7. Intersect Data From Two Tables

1. Select the custom track created in the previous section. Click the “intersection: create” button. The Genome Browser displays an intersection configuration page offering several overlap combinations (*see* **Note 18**). Select the “Genes and Gene Prediction Tracks” option from the group menu and the “Known Genes” option from the track menu. The table menu will default to the primary Known Genes table, knownGene. For this example, retain the default intersection settings. Click the submit button to activate the intersection.
2. On the main Table Browser page, set the output format to “BED—browser extensible data” (*see* **Note 19**). Click the “get output” button.
3. Retain the default settings on the BED configuration page and click the “get BED” button. The Table Browser displays those items from the custom track that have coordinates overlapping exons in the Known Genes track. If no overlaps are found, try using a lower threshold in the filter (**Subheading 3.6., step 4**) or expanding the query region (**Subheading 3.5., step 3**).

4. Notes

1. In addition to the Genome Browser and Table Browser tools described in this procedure, the user will find several other tools that may be useful in the research: Blat (**20**), which quickly maps sequences to a genome assembly; the Gene Sorter (**21**), which shows relationships (expression, homology, and so on) among groups of genes; VisiGene, which supports browsing through a large collection of *in situ* mouse and frog images to examine expression patterns; the Proteome Browser (**22**), which offers a wealth of information about a selected protein; and an *in silico* PCR tool that provides a fast search of a sequence database with a pair of PCR primers. The Help link—available in the top menu bar of most pages on the website—displays an online User’s Guide containing detailed information about the UCSC tools. The FAQ link provides access to a collection of frequently asked questions, many taken from the archives of the user-support mailing list (*see* <http://www.soe.ucsc.edu/mailman/listinfo/genome>). Additional information can be found via the Training link, which provides access to online and onsite Genome Browser training materials, and the Publications link, which lists selected publications by the UCSC Genome Bioinformatics Group and its collaborators.
2. Examples of legitimate search terms include a gene name, an accession of an mRNA, EST, or clone, an STS marker, a chromosomal range, or one or more keywords from the GenBank description of an mRNA. The Gateway page for each genome assembly includes a list of sample search terms specific to that assembly.
3. The first time the Genome Browser is opened in a given Internet browser, it displays a standard set of tracks using the default application configuration. The setting may be reconfigured to reflect the user’s preferences (**Subheading 3.2.**). Configuration preferences set during a session are retained in subsequent sessions in the same Internet browser if cookies are enabled.

4. To zoom in threefold centered on a particular coordinate, click a position in the Base Position line at the top of the image. To quickly zoom in and view the base composition of the sequence underlying the current annotation track display, click the base button.
5. All Genome Browser tracks have at least three display mode options: hide—the track is not displayed in the graphical image; dense—the track features are collapsed into a single line; and full—each feature within the track is displayed on a separate line. Many tracks have two additional display options: pack—each feature is separately displayed and labeled, but not necessarily on a separate line, and squish—similar to pack mode, but features are displayed unlabeled at half-height. Dense displays are useful for getting an overview of the annotation's density without the clutter of individual features. The squish and pack display modes are useful for viewing feature details of densely populated tracks while conserving space.
6. The configuration page provides a convenient way to hide or display entire groups of tracks, or to hide the entire track display control section if it is preferential to display only the graphical image on the Genome Browser page. Exercise caution when selecting the “show all” option; on assemblies with a large amount of annotation data, this may exceed the Internet browser's capacity, causing it to freeze or terminate.
7. In future revisions of the Genome Browser, the individual pairwise annotation tracks may be merged into a set of combined net and chain tracks.
8. Alternatively, the description page can be displayed by clicking the label above the track's pull-down display menu in the track controls section of the main Genome Browser page.
9. Click the “Graph configuration help” link for detailed information about each option. In addition to the text description, most Conservation track description pages display an illustration depicting the order in which the pairwise alignments were multiply aligned prior to the assignment of conservation scores ([Fig. 2](#)).
10. If the displayed coordinate range is greater than 30,000 bases, the Genome Browser will be unable to display base-level information on the track details page. In this instance, use the zoom in buttons or click on the ruler to reduce the size of the displayed region below the 30,000-base limit. To view a graphical representation of the base-level alignments, zoom in on the region of interest until the pairwise alignment graphs are replaced by bases or click the “zoom in base” button. An explanation of the numbers and symbols used to denote gaps in the graphical representation can be found at the bottom of the track details page.
11. Many annotation tracks, such as the Conservation track, are based on data from multiple tables joined on common fields. In these instances, the primary data table underlying the track is listed first in the table menu. The “All Tracks” and “All Tables” option in the group menu provide convenient shortcuts if the name of the track or table to be opened is already known.

12. The Table Browser supports the same list of position search terms supported in the Genome Browser. Use caution when querying large regions; the Internet browser session may time out. In this situation, subdivide the query into smaller regions and combine the data results.
13. The Table Browser limits the output size of queries using the “data points” format to 100,000 lines. To increase this limit, click the “filter: create” button, select a larger output size from the pull-down menu, then click the submit button to apply the new limit. The “Using the Table Browser” section on the main Table Browser page describes the output format options. Only a subset of options is available for a given data type. Some data operations restrict the use of certain formats; for example, the “all fields from selected table” and “selected fields from primary and related tables” options may not be used to display data derived from the intersection of two tables. For more information on special data formats such as browser extensible format (BED), multiple alignment format (MAF), and Gene Transfer Format (GTF), see the “Data File Formats” section in the FAQ.
14. In some instances, this page also displays other tables in the database that are joined to the current table by a common field.
15. Filters are specific to a given table within a given assembly. Once set, a filter is preserved within the Table Browser session until a different table is selected or the filter is removed. When a filter is active on the currently selected table, an edit button displays next to the filter label. To modify an existing filter, click the “filter: edit” button; to remove it, click “filter: clear”.
16. Custom annotation tracks provide a convenient way to save different snapshots of the annotation data for comparison—for example, data captured at different filter settings. Custom annotation data may also be loaded into the Genome Browser using the add custom tracks option on the Gateway page. To load a data into the Table Browser, first load and display the track in the Genome Browser, then click the Tables option in the Genome Browser menu bar to automatically load the track into the Table Browser. Once loaded, a track is retained for 48 h after its last access or until the session is terminated. To remove a loaded custom track from a Table Browser session, select the “Custom Tracks” option from the group menu, select the custom track in the track menu, then click the “remove custom track” button displayed next to the table menu. For more information about creating and using custom annotation tracks, see the “Creating custom annotation tracks” section in the Genome Browser User’s Guide.
17. The Table Browser presents numerous options for saving custom track data. The “get custom track in table browser” button saves the data set in a temporary table and adds an option for the track to the track and table pull-down menus. The “get custom track in file” option saves the data to the file designated by output file on the main Table Browser page or outputs the data to the screen if no file is specified. The “get custom track in genome browser” option opens the Genome

Browser to the coordinate range specified by the Table Browser and displays the track in a special Custom Tracks group.

18. When setting up a Table Browser intersection, the user is required to select a second table for the intersection and the type of data combination. An intersection yields different results, depending on which of the two tables is selected first. There are two general types of data combinations: those that retain the alignment structure of the table with which the user is intersecting and those that perform intersections at the basepair level, thereby replacing the alignment structure with a list of coordinate ranges. When the basepair level intersection is selected, the user may optionally choose to complement one or both tables, which will have the effect of including only those data records not included in the complemented table(s). The intersection options may be limited by the data structure of the table selected for the intersection. If one or both of the tables are based on exon or block structure, only the exons or blocks are intersected, not the entire span.
19. The output options “all fields from selected table” and “selected fields from primary and related tables” are not available when an intersection is active.

Acknowledgments

The UCSC Genome Browser project is funded by grants from the National Human Genome Research Institute (NHGRI), the Howard Hughes Medical Institute (HHMI), and the National Cancer Institute (NCI). We would like to acknowledge the excellent work of the Genome Browser technical staff who maintain and enhance the Genome Browser database and software, the many collaborators who have contributed annotation data to the project, and our loyal users for their feedback and support.

References

- 1 Waterston, R. H., Lindblad-Toh, K., Birney, E., et al. (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**, 520–562.
- 2 Chiaromonte, F., Weber, R. J., Roskin, K. M., Diekhans, M., Kent, W. J., and Haussler, D. (2003) The share of human genomic DNA under selection estimated from human-mouse genomic alignments. *Cold Spring Harbor Symp. Quant. Biol.* **68**, 245–254.
- 3 Roskin, K. M., Diekhans, M., and Haussler, D. (2003) Scoring two-species local alignments to try to statistically separate neutrally evolving from selected DNA segments. *Proc. 7th Int’l Conf. on Research in Computational Molecular Biology (RECOMB ’03)*, 257–266.
- 4 Hinrichs, A. S., Karolchik, D., Baertsch, R., et al. (2006) The UCSC Genome Browser database: update 2006. *Nucl. Acids Res.* **34**, D590–D598.
- 5 Kent, W. J., Sugnet, C. W., Furey, T. S., et al. (2002) The human genome browser at UCSC. *Genome Res.* **12**, 996–1006.

- 6 Bejerano, G., Pheasant, M., Makunin, I., et al. (2004) Ultraconserved elements in the human genome. *Science* **304**, 1321–1325.
- 7 Bejerano, G., Haussler, D., and Blanchette, M. (2004) Into the heart of darkness: large-scale clustering of human non-coding DNA. *Bioinformatics* **20**, I40–I48.
- 8 Woolfe, A., Goodson, M., Goode, D. K., et al. (2005) Highly conserved non-coding sequences are associated with vertebrate development. *PLoS Biol.* **3**, 0116–0130.
- 9 Glazov, E. A., Pheasant, M., McGraw, E. A., Bejerano, G., and Mattick, J. S. (2005) Ultraconserved elements in insect genomes: a highly conserved intronic sequence implicated in the control of homothorax mRNA splicing. *Genome Res.* **15**, 800–808.
- 10 Bejerano, G., Siepel, A. C., Kent, W. J., and Haussler, D. (2005) Computational screening of conserved genomic DNA in search of functional noncoding elements. *Nat. Methods* **2**, 535–545.
- 11 Siepel, A., Bejerano, G., Pedersen, J. S., et al. (2005) Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res.* **15**, 1034–1050.
- 12 Pedersen, J. S., Bejerano, G., Siepel, A., et al. (2006) Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Comput. Biol.* **2**, e33.
- 13 Karolchik, D., Baertsch, R., Diekhans, M., et al. (2003) The UCSC Genome Browser database. *Nucl. Acids Res.* **31**, 51–54.
- 14 Karolchik, D., Hinrichs, A. S., Furey, T. S., et al. (2004) The UCSC Table Browser data retrieval tool. *Nucl. Acids Res.* **32**, D493–D496.
- 15 Kent, W. J., Baertsch, R., Hinrichs, A., Miller, W., and Haussler, D. (2003) Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Pro. Natl. Acad. Sci. USA* **100**, 11,484–11,489.
- 16 Schwartz, S., Kent, W.J., Smit, A., et al. (2003) Human-Mouse alignments with BLASTZ. *Genome Res.* **13**, 103–107.
- 17 Blanchette, M., Kent, W. J., Riemer, C., et al. (2004) Aligning multiple genomic sequences with the Threaded Blockset Aligner. *Genome Res.* **14**, 708–715.
- 18 Hsu, F., Kent, W.J., Clawson, H., Kuhn, R.M., Diekhans, M., and Haussler, D. (2006) The UCSC Known Genes. *Bioinformatics* **22**, 1036–46.
- 19 The ENCODE Project Consortium. (2004) The ENCODE (ENCyclopedia Of DNA Elements) project. *Science* **306**, 636–640.
- 20 Kent, W. J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.* **12**, 656–664.
- 21 Kent, W. J., Hsu, F., Karolchik, D., et al. (2005) Exploring relationships and mining data with the UCSC Gene Sorter. *Genome Res.* **15**, 737–741.
- 22 Hsu, F., Pringle, T. H., Kuhn, R. M., et al. (2005) The UCSC Proteome Browser. *Nucleic Acids Res.* **33**, D454–D458.

Comparative Genome Analysis in the Integrated Microbial Genomes (IMG) System

Victor M. Markowitz and Nikos C. Kyrpides

Summary

Comparative genome analysis is critical for the effective exploration of a rapidly growing number of complete and draft sequences for microbial genomes. The Integrated Microbial Genomes (IMG) system (img.jgi.doe.gov) has been developed as a community resource that provides support for comparative analysis of microbial genomes in an integrated context. IMG allows users to navigate the multidimensional microbial genome data space and focus their analysis on a subset of genes, genomes, and functions of interest. IMG provides graphical viewers, summaries, and occurrence profile tools for comparing genes, pathways, and functions (terms) across specific genomes. Genes can be further examined using gene neighborhoods and compared with sequence alignment tools.

Key Words: Comparative genome data analysis; integrated microbial genomes; occurrence profiles; microbial genome data management; comparative genome data analysis; gene occurrence profile; functional occurrence profile; gene model validation; integrated microbial genomes.

1. Introduction

Microbial genome analysis is a growing area that is expected to lead to advances in healthcare, environmental cleanup, agriculture, industrial processes, and alternative energy. According to the Genomes Online Database, as of April 2007 close to 500 microbial genomes have been sequenced to date, whereas more than 1000 additional projects are ongoing or in the process of being launched (*1*). As the genomic community is rapidly moving toward

the generation of complete and draft sequences for several hundred microbial genomes, comparative data analysis in the context of integrated genome data sets plays a critical role in understanding the biology of the newly sequenced organisms. Conversely, individual organism-specific genome analysis carried out in isolation cannot support timely analysis of newly released genomes.

Microbial genomes are sequenced by organizations worldwide, follow an annotation process (gene prediction and functional characterization) that is often specific to each sequencing center, and end up in one of the public sequence data repositories, such as GenBank in the United States, EMBL in Europe, and DDBJ in Japan.

Genome sequence data include information on gene coordinates, transcription orientation, locus identifiers, gene names, and protein functions. Analyzing microbial genomes requires however additional functional annotations, such as motifs, domains, pathways, and ontology relationships, which are provided by diverse, usually heterogeneous, data sources, such as Pfam (2), InterPro (3), COG (4), CDD (5), KEGG (6), and Gene Ontology (GO) (7). Resources such as EBI Genome Reviews (8) and RefSeq (9) include such additional functional annotations, sometimes after reannotating the sequences from the public sequence data sources. These resources share common goals, but contain different collections of genomes or data with different degrees of resolution regarding the same genomes. These differences are the result of diverse annotation methods, curation techniques, and functional characterization employed across microbial genome data sources.

Comparative genome data analysis is critical for effective exploration of the rapidly growing number of complete and draft sequences for microbial genomes. For example, the efficiency of the functional characterization of genes in newly sequenced genomes can be substantially improved if this characterization involves methods based on observed biological evolutionary phenomena. Thus, genes with related (coupled) functions are often both present or both absent within specific genomes and tend to be collocated (on chromosomes) in multiple genomes (10). The effectiveness of comparative analysis depends on the availability of powerful analytical tools and the efficiency of the integration, which in turn is determined by the phylogenetic diversity of the organisms, the quality of their annotations, and the level of detail in cellular reconstruction. The efficiency of the integration depends on its breadth (in terms of the number of genomes it involves) and depth (in terms of different annotations it captures). Integration of available genomic data provides the context for comparative genome analysis, and is becoming the single most important element for understanding the biology of the newly sequenced organisms. Analyzing genomes

in the context of other (e.g., phylogenetically related) genomes is substantially more efficient than analyzing each genome in isolation.

The Department of Energy's Joint Genome Institute (JGI) is one of the major contributors of microbial genome sequence data, currently conducting about 23% of the reported archaeal and bacterial genome projects worldwide. Individual microbial genomes are sequenced and assembled to draft level at JGI's production facility, and finished either at JGI's production facility, Lawrence Livermore, or Los Alamos National Labs. Both draft and finished genomes pass through the automatic Genome Analysis Pipeline (11) at Oak Ridge National Lab, which generates gene models and associates automatically predicted genes with functional annotations, such as InterPro protein families, COG categories, and KEGG pathway maps.

Before publication or submission to GenBank, scientific groups interested in a specific genome further review and curate the microbial genome data in collaboration with Oak Ridge National Lab's Computational Biology group and JGI's Genome Biology Program. As previously mentioned, the efficiency of microbial genome review, curation, and analysis increases substantially when individual microbial genomes are examined in the context of other genomes. Providing such a framework, to ensure timely analysis of the genomes sequenced at JGI, is one of the main goals of the Integrated Microbial Genomes (IMG) system (12). IMG aims at providing high levels of data diversity in terms of the number of genomes integrated in the system from public sources, data coherence in terms of the quality of the gene annotations, and data completeness in terms of breadth of the functional annotations.

2. The IMG System

The IMG system provides support for comparative analysis of microbial genomes in an integrated genome data context. IMG integrates microbial and selected eukaryotic genomic data from multiple data sources. A high level of genome diversity is ensured by collecting data from public sources, such as EBI Genome Reviews, National Center for Biotechnology Information's RefSeq, and EMBL Nucleotide Sequence Database.

The data model underlying the IMG system provides the structure required for integrating and managing microbial and selected eukaryotic genomic data collected from multiple data sources. The system incorporates in a coherent biological context several data types: (1) primary genomic sequence information, (2) computationally predicted and curated gene models, (3) precomputed gene relationships (which are sequence similarity based, gene context based, and so on), and (4) functional annotations and pathway information. The user interface is organized in a manner that allows navigation over the microbial

genome data space along its three key dimensions representing genomes, genes, and functions, respectively.

Genomes (organisms) are identified and organized either based on their taxonomic lineage (domain, phylum, class, order, family, genus, species, strain) or other organism specific properties, such as phenotypes, ecotypes, disease, and relevance. For each genome, the primary DNA sequence and its organization in scaffolds or contigs, are recorded. Genomic features, such as predicted coding sequences and some functional RNAs, are recorded with start/end coordinates. Predicted genes are grouped based on sequence similarity relationships: ortholog and paralog gene relationships are currently computed based on bidirectional best hit single-linkage. COGs provide an additional clustering of orthologous groups of genes in IMG.

Genes are further characterized in terms of molecular function and participation in pathways. Metabolic pathways are modeled in IMG as ordered lists of reactions and consist usually of one to four reactions. A reaction can include compounds which are reactants (substrates, products) catalyzed by enzymes, and physical entities such as proteins, protein complexes, electrons, and so on. Nonmetabolic pathways are modeled in IMG as lists of functions. Pathways are combined into networks via reactions that share common components. Networks can be further combined into more complex networks. Note that networks are different from KEGG maps, which represent complex networks. Pathways are associated with genes via gene products that function as enzymes that serve as catalysts for individual reactions of metabolic pathways. The association of genes with pathways in IMG is based on a controlled vocabulary of terms. IMG terms are defined by domain experts as part of the process of including IMG pathways into the system. The IMG pathways are consistent with the BioPAX ([13](#)) level 1 data exchange format in order to facilitate sharing these data across different systems. In addition to the IMG terms and pathways, resources, such as COG, Pfam and InterPro, are used for the functional characterization of genes. Finally, pathways, reactions, and compounds are included from KEGG and LIGAND.

The first version of IMG was released on March 1, 2005. The current version of IMG (IMG 1.4, as of March 1, 2006) contains a total of 699 genomes consisting of 395 bacterial, 30 archaeal, 15 eukaryotic genomes, and 259 bacterial phages.

3. Comparative Genome Data Analysis in IMG

Data analysis in IMG is set in a multidimensional data space, whereby genes form one of the dimensions and are characterized in the context of other dimensions, in particular individual organisms (genomes), functions, and networks of

pathways. Genes are directly associated with genomes (via gene prediction), as well as with functions and pathways (via functional characterization). An organism is associated with a specific function f or pathway p if its genome has a gene that is associated with f or p , respectively. Genes can be grouped (clustered) in terms of their sequence similarity or associations with functions and pathways.

Each dimension in the microbial genome data space is characterized by one or several category attributes whose values can be used to specify a classification hierarchy. For example, phylogeny serves as a category attribute for organisms and is used to specify their phylogenetic tree classification. Phenotypic attributes, such as origin of the sample used for sequencing (e.g., ocean, groundwater, and so on) can also serve as category attributes for organisms.

Microbial genome data analysis operations allow navigating the multidimensional data space along one or several dimensions and can be set in the context of specific (i.e., subsets of) organisms, functions, or pathways. Organism (genome) selections help focus the analysis on a subset of interest, especially in terms of phylogenetic or phenotypic relationships. For example, a set of interest may include all the strains within a specified species. Similarly, function selections focus the analysis on a subset of interest, such as functions involved in lipid metabolism pathways. Finally, gene selections reduce the scope of analysis to genes with certain properties, such as genes sharing a common function or genes that are collocated on the chromosome.

An important type of analysis operation regards examining so-called occurrence profiles (14,15) of objects of interest (e.g., functions) selected from one dimension of the multidimensional data space, across objects (e.g., organisms) selected from another dimension of the data space.

Consider two dimensions of the data space representing functions and organisms, respectively. The occurrence profile for a function of interest (e.g., enzyme), f , shows the pattern of f across organisms y_1 to y_n in the form of a vector $(L_1, \dots, L_n)$ where L_i represents the set of y_i genes that are associated with f . Similarly, the profile for a gene, x , across organisms y_1 to y_n has the form of a vector $(L_1, \dots, L_n)$ where L_i represents a set of y_i genes that are associated with x , where the association of y_i genes with x is based on a specific sequence similarity method.

The number of genes in a set L_i, k_i , is called gene abundance and vectors of the form $(k_1, \dots, k_n)$ are called abundance profiles. Presence profiles are a special case of abundance profiles, whereby in each vector of the form $(k_1, \dots, k_n)$, k_i is replaced by either “a” (absent) if k_i is zero or “p” (present) otherwise. **Figure 1** shows an example of abundance profiles for genes x_1 to x_4 across organisms y_1 to y_8 .

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
x_1	2	1	1	3	0	0	1	0
x_2	1	1	2	2	0	0	1	0
x_3	0	1	1	0	0	0	0	0
x_4	1	1	1	1	2	1	2	1

Fig. 1. Abundance profile example.

Profiles for objects that are aggregations (compositions) of other objects consist of all the profiles for their component objects. For example, the profile of a metabolic pathway consists of the profiles for the enzymes involved in the pathway, whereas the profile of a network consists of the profiles of its component pathways.

Analysis based on occurrence profiles usually involves: (1) examining the profiles for objects of a given type across objects of another type; or (2) finding objects of a given type that either have a predefined presence profile or whose presence profile is similar to the presence profile of a given object of the same type, across objects of another type.

For example, examining the profiles of the genes of a specific organism, y , in the context of other related organisms, $y_1, \dots, y_k$ allows determining what y may have in “common” with $y_1, \dots, y_k$. Sequences with sufficient degree of similarity are deemed to encode the same gene, and accordingly are considered “common” to or “present” in selected organisms. For the example shown in [Fig. 1](#), organism y has gene x_4 in “common” with organisms y_1 to y_8 ; and genes x_1 and x_2 have the same presence profile across genomes y_1 to y_8 . Note that an organism having multiple genes (e.g., three genes of y_4 in [Fig. 1](#)) corresponding to a specific gene in another organism (e.g., gene x_1 in [Fig. 1](#)) is the result of the similarity method employed (e.g., homology) in computing profiles. Finding a unique orthologous gene in an organism corresponding to another gene in a different organism is straightforward only for singly copy genes. For other genes, establishing orthologous relationships across organisms is complicated by the fact that most genes undergo either gene duplications or fusion events, with subsequent losses of some of the duplicated copies adding to the complexity of determining such relationships.

Occurrence profile operations can be used for analyzing biological phenomena such as gene conservation or gain, for a specific organism (e.g., y)

in the context of other organisms (e.g., $y_1, \dots, y_k$). For the example shown in **Fig. 1**, gene x_4 is conserved across y_1 to y_8 , whereas gene x_3 is gained with respect to y_1 and y_4 to y_8 .

Occurrence profiles are critical in the process of understanding the biology of the microbial genome under study. This process is based on observed biological evolutionary phenomena: genes with related (coupled) functions (1) are often both present or both absent within specific genomes that have these functions; (2) tend to be collocated (on chromosomes) in multiple genomes; (3) might be fused into a single gene in some genomes; or (4) are cotranscribed under the same regulator (10).

Consider the example shown in **Fig. 2**, where pathway p involves reactions R_1, R_2, R_3 , and R_4 : genes x_1, x_2 , and x_4 of genome G_1 are associated with pathway p via enzymes e_1, e_2 , and e_4 , respectively; genes z_1, z_2, z_3 , and z_4 of genome G_2 are associated with pathway p via enzymes e_1, e_2, e_3 , and e_4 , respectively; if gene x_3 is similar (i.e., determined to be related via significant sequence similarity) to gene z_3 , then, following the rules previously listed, x_3 may be associated with p via enzyme e_3 .

For the example shown in **Fig. 1**, suppose that gene x_1 is functionally characterized, whereas x_2 is not; then the fact that genes x_1 and x_2 have similar occurrence profiles across organisms y_1 to y_8 , may help characterize x_2 , which may participate in a similar biological process as gene x_1 .

Finding objects that have a specific presence profile are used for identifying certain (e.g., unique) genes in an organism in the context of other organisms. For example, consider finding genes of a target organism in terms of presence

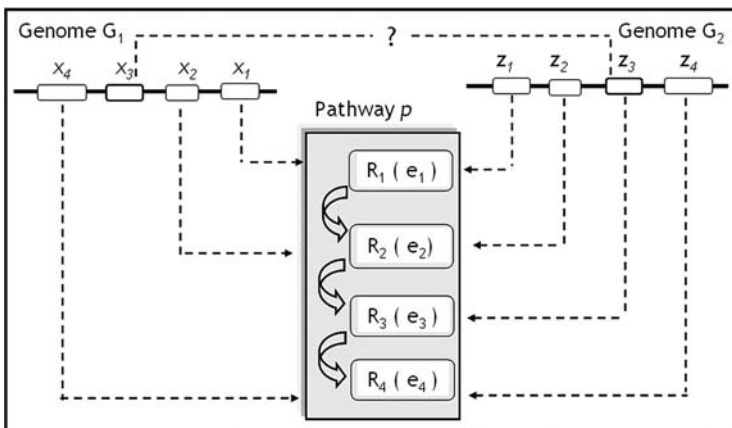


Fig. 2. Example of functional characterization of genes.

or absence of homologs (or orthologs) in other reference organisms. Reference organisms can be defined based on some biological property, such as phylogenetic relationship, shared phenotype, or ecological environment. For example, if the reference organisms are phylogenetically related then finding genes that have a specific profile could be used to identify preserved, gained, or lost genes. Although the preserved genes are shared by all organisms in a phylogenetic lineage and therefore are likely to be inherited from the last common ancestor, gene gain and loss in the target organism (or group of organisms) can be related to the specific adaptation to the ecological environment of these organisms. A potential application of the occurrence profiles is the identification of genes and other genomic properties that can be used to distinguish between different species or strains of the same species of pathogens using a variety of molecular diagnostics tools.

Occurrence profiles involving functions, pathways, and other genomic data are used in comparative analysis in a way similar to that previously discussed for genes. For example, occurrence or abundance profiles of certain COGs (such as signal transduction histidine kinase, serine/threonine protein kinase, and phosphatase) can provide a broad overview of protein families present or absent in the genomes of interest, whereas occurrence profiles of Pfam domains found in these proteins could provide additional information on the signals sensed by the proteins.

4. Occurrence Profile Analysis in IMG

Comparative genome data analysis in IMG is set in the context of integrated microbial genomes. IMG allows exploring the microbial genome data space along three key dimensions: genomes (organisms), functions, and genes. Comparative analysis for genomes is provided in IMG through a number of tools that allow genomes to be compared in terms of organism-specific summaries (statistics), genes, and functional annotations. Next, we discuss the occurrence profile analysis tools provided by IMG in more detail. Note that all the examples provided in this section are based on IMG 1.4 (March 2006). IMG's content and user interface are extended on a regular basis, therefore these examples may be different for subsequent versions of IMG.

4.1. Analysis Context

The context for occurrence profile analysis is defined by the set of genomes, genes, and functions of interest selected by the user. By default this context involves all the genomes, genes, and functions in the system.

Genome (organism) selections provide the option of focusing the analysis on a subset of genomes of interest, such as strains within a specified species. Genomes can be selected using a keyword-based Genome Search in conjunction with a number of filters, such as phenotype, ecotype, disease relevance, or phylum. Organisms can also be selected from an alphabetical or phylogenetically organized list available in the Organism Browser. Genome selections can be saved in order to set or reset the analysis context.

Genes can be selected using keyword-based gene search, sequence similarity search, or gene profile-based selection. Gene Search allows finding genes based on partial or exact matches to a string of characters in specified IMG fields such as gene name or locus tag. Similarity searches are implemented via BLASTp (Basic Local Alignment Search Tool protein-vs-protein), BLASTx (DNA-vs-protein), BLASTn (DNA-vs-DNA), or tBLASTn (protein-DNA-vs-DNA-protein). Users can define similarity thresholds and select the target database. Gene profile-based selection is provided by the Phylogenetic Profiler, which is discussed in more detail next. Gene selections can be saved in a gene specific Analysis Cart called Gene Cart (similar to shopping carts of commercial websites) in order to set or reset the analysis context.

Functional roles of genes in IMG are characterized by a variety of annotations, including their COG membership, association with Pfam domains, and association with enzymes in KEGG pathways. Functional annotations can be searched using keywords and filters, with the selected functions leading to a list of associated genes either directly or via a list of organisms. COG categories and KEGG pathways also can be searched and browsed separately. Function selections can be saved in a function specific Analysis Cart in order to set or reset the analysis context.

In summary, the analysis context is defined by the set of genomes, genes, and functions of interest selected by the user, where the set of genomes is maintained using a genome list, whereas genes and functions are maintained using Analysis Carts.

4.2. Occurrence Profile Computation Tools

As discussed in the previous section, occurrence profiles are specified in a two-dimensional data space, where one dimension represents a set of genes or functions, x_1 to x_n , whose profiles are computed in the context of the other dimension, which represents a set of organisms, y_1 to y_m . The occurrence profile for a gene or function of interest, x , consists of a vector of the form $(L_1, \dots, L_n)$ where L_i represents the set of genes of y_i that are either (1) similar to x (if x is a gene) or (2) genes of y_i that are associated with x (if x is a function).

Occurrence profile results can be displayed as two-dimensional matrices or projected on a phylogenetically organized list of organisms.

Next, we present several examples of employing IMG occurrence profiles in data analysis together with alternative visual presentations of the profile results.

4.2.1. COG-Based Functional Occurrence Profiles Example

The following example illustrates how functional occurrence profiles are used in comparative genome analysis. In this example, such a profile is used to examine the presence of a specific pathway (i.e., CO₂ fixation) in a set of selected organisms, namely in the archaeal class of *Methanomicrobium archaea*. These organisms can first be selected using IMG's phylogenetic-based Genome Browser as shown in **Fig. 3 (i)** and then saved in order to focus the analysis context as previously discussed.

The first step in one of the CO₂ fixation pathways is catalyzed by a CO dehydrogenase/acetyl-CoA synthase enzyme. A keyword search on expression “CO dehydrogenase/acetyl-CoA synthase” with COG as a filter (*see* [Fig. 3 \[ii\]](#))

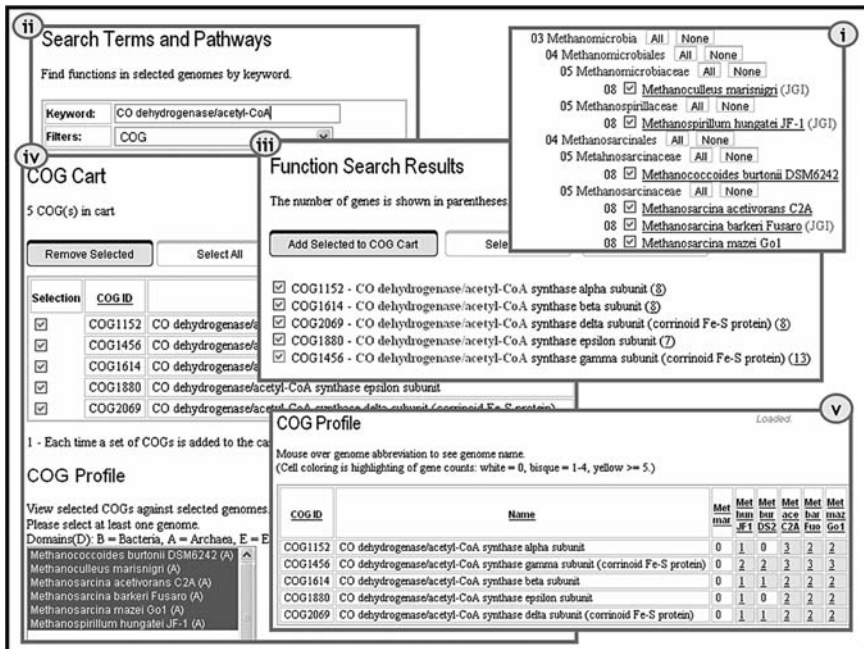


Fig. 3. Finding genes responsible for carbon fixation in methanomicrobia archaea organisms.

retrieves a list of five COGs corresponding to different subunits of CO dehydrogenase/acetyl-CoA synthase, as shown in **Fig. 3 (iii)**. After these COGs are saved with the COG Cart (*see* **Fig. 3 (iv)**), their occurrence profiles across the methanomicrobia organisms are displayed in a tabular format as shown in **Fig. 3 (v)**, with each row displaying the profile of a specific COG across the selected organisms. Each cell in the profile result table contains a link to the associated list of genes and displays the count (abundance) of genes in this list. Colors are used to represent visually gene abundance, whereby white, bisque and yellow represent gene counts of zero, one to four, and more than four, respectively.

In this example, the occurrence profile result suggests that, with the exception of one organism, CO dehydrogenase/acetyl-CoA synthase is present in these organisms, which means that they rely on this pathway for CO₂ fixation.

4.2.2. KEGG-Based Functional Occurrence Profiles Example

The next example illustrates how functional occurrence profiles can be used for comparing phylogenetically related organisms. In the example shown in **Fig. 4**,

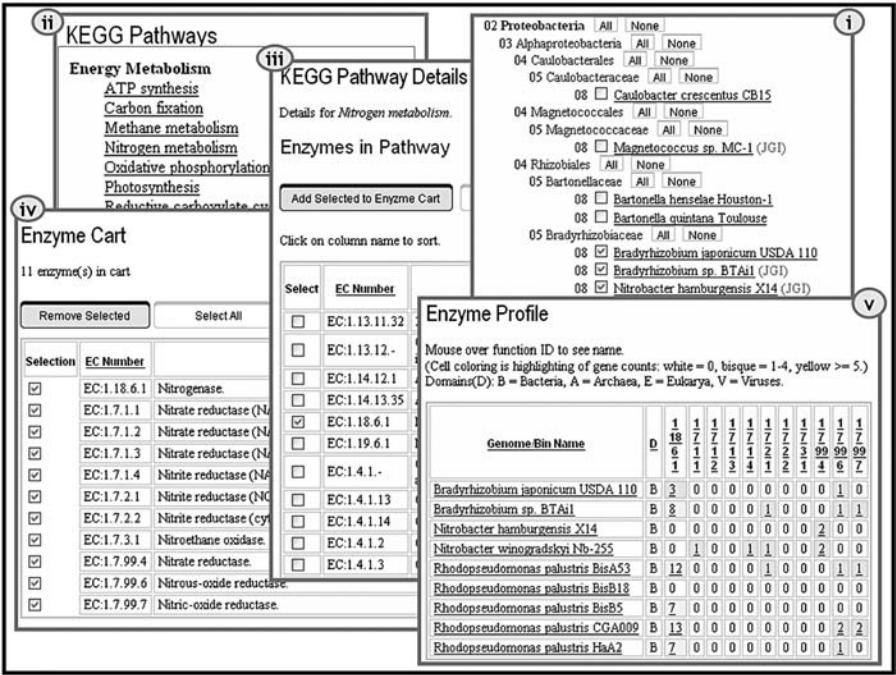


Fig. 4. Examining nitrogen metabolism in *Bradyrhizobiaceae* organisms.

occurrence profiles of the enzymes participating in nitrogen metabolism are analyzed across the organisms that belong to the family of bradyrhizobiaceae. These organisms are first selected using IMG's phylogenetic-based Genome Browser as shown in **Fig. 4 (i)** and saved in order to reduce the analysis context as previously discussed.

Starting with the KEGG Pathway Browser (*see* **Fig. 4 (ii)**), enzymes in the Nitrogen Metabolism pathway are selected with the KEGG Pathway Details as shown in **Fig. 4 (iii)**. A set of enzymes, including nitrogenase, different versions of nitrate reductase, and nitrite reductase, is then saved with the Enzyme Cart as shown in **Fig. 4 (iv)**. The occurrence profiles of these enzymes across the bradyrhizobiaceae family are displayed in a tabular format as shown in **Fig. 4 (v)**, with each column displaying the profile of a specific enzyme across selected organisms. Each cell in the profile result table contains a link to the associated list of genes and displays the count (abundance) of genes in this list. Note that the occurrence profile tools in IMG provide two alternative display options (functions vs genomes and genomes vs functions) as illustrated in this and previous examples.

In this example, the analysis of occurrence profiles shown in **Fig. 4 (v)** suggests that nitrogen metabolism may be different across these organisms.

4.2.3. Gene Occurrence Profiles Example

The following example illustrates how gene occurrence profiles can be used to examine metal binding in *Shewanella*. First, metal binding-related functions are found with IMG's Function Search using Pfam or InterPro as filters. For example, Pfam 02805 is associated with a list of genes that include *Shewanella* genes that are related to metal binding. These genes are saved using Gene Cart, as shown in **Fig. 5 (i)**. In this example, the presence profiles for genes are displayed in the form of vectors where each position in the vector corresponds to an organism, as shown in **Fig. 5 (ii)**: the organisms are phylogenetically ordered to facilitate comparison of closely related organisms. Presence of an ortholog of a gene in a given organism is indicated by a domain letter, "B" for bacteria, "A" for archaea, and "E" for eukarya, whereas the absence of the gene is indicated by a dot ("."). One can mouse over the letter or dot to see the organism name along with its phylum. For the example shown in **Fig. 5**, the occurrence profiles for the *Shewanella* genomes are highlighted (*see* **Fig. 5 (iii)**).

For a single gene, IMG also provides the Phylogenetic Distribution Viewer, which presents the abundance profile for that gene across the phylogenetically organized list of organisms. The abundance of the selected gene is indicated

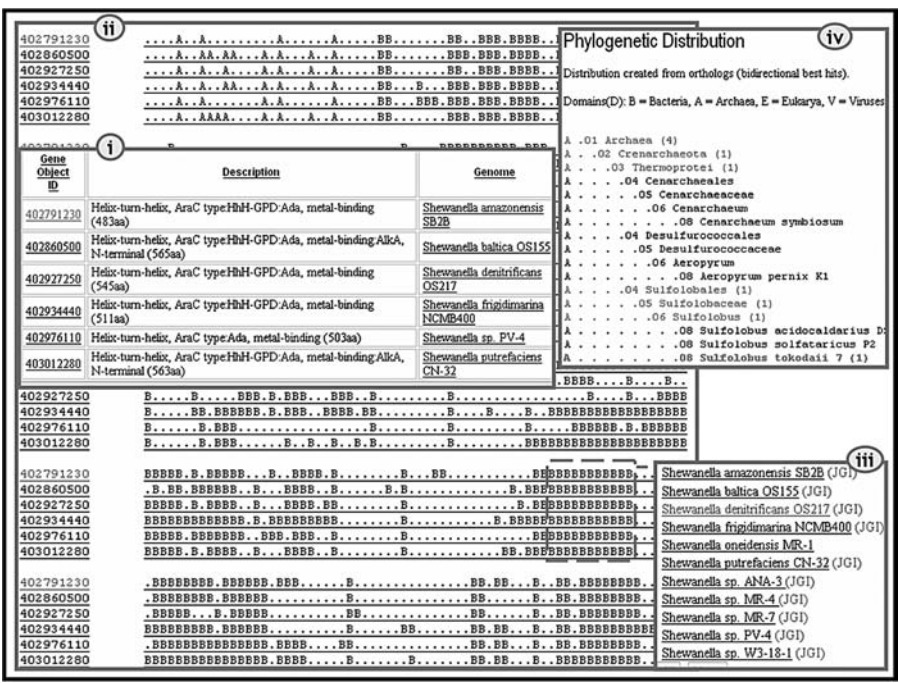


Fig. 5. Gene phylogenetic occurrence profile and distribution viewer examples.

by the count of homologous genes at each taxonomic level as shown in Fig. 5 (iv).

4.3. Occurrence Profile Selection Tools

Occurrence profiles can be used for finding objects (e.g., genes, functions) that share a specific presence profile across a set of organisms. IMG's Phylogenetic Profiler is a tool that allows finding genes in a target organism that share the same gene presence profile, where presence or absence of genes is based on (homologous) gene similarity, with cutoffs used to define the similarity relationship.

In the example shown in Fig. 6, the Phylogenetic Profiler is used to find genes from a *Burkholderia mallei* strain that have no homologs in a *Burkholderia pseudomallei* strain. Similarity cutoffs can be used to fine-tune the selection. The list of genes with the specified profile is then provided as a selectable list as shown in Fig. 6.

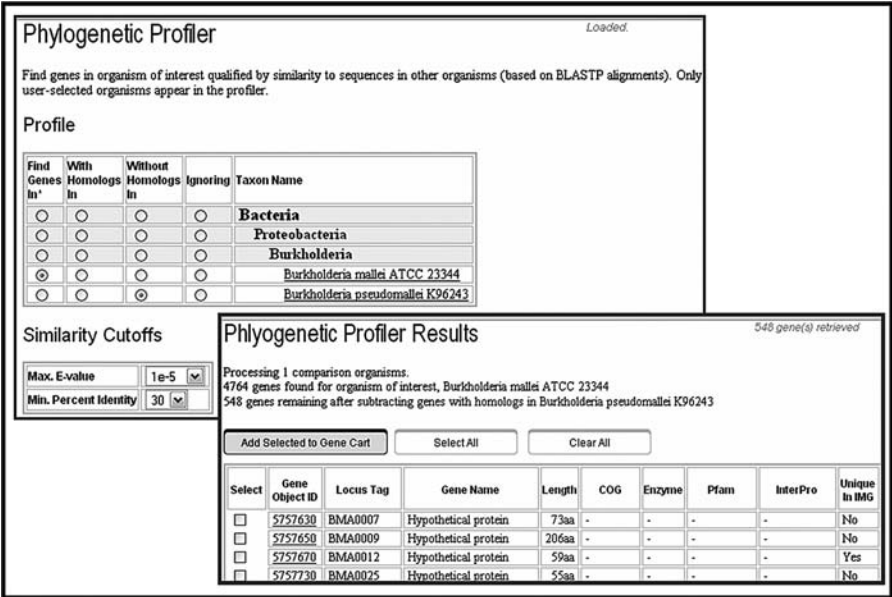


Fig. 6. Finding *Burkholderia mallei* genes without homologs in *Burkholderia pseudomallei*.

The Phylogenetic Profiler can be used, for example for finding unique, common, or lost genes in the (query) organism of interest compared to a target group of organisms. In the example shown in Fig. 6, 548 genes are found to be unique in *B. mallei* ATCC 23344 (*B. mallei*) with respect to *B. pseudomallei* K96243 (*B. pseudomallei*). As we discuss next, such gene profile-based selections provide the context for analyzing phylogenetically related genomes and reviewing their gene models.

4.4. Interpreting Occurrence Profile Results

Occurrence profile results involve organisms, functional roles (e.g., Pfam families, COGs, enzymes), and sets of genes, each of which can be further examined.

For a set of selected organisms comparative summaries are provided using the Organism Statistics as illustrated in the left panel of Fig. 7, where summaries for the *B. mallei* and *B. pseudomallei* strains previously mentioned are presented in the context of other related *Burkholderia* strains. These summaries include the total number of genes and enzymes, and the number of genes with various characteristics, such as genes associated with KEGG pathways, COGs, Pfam,

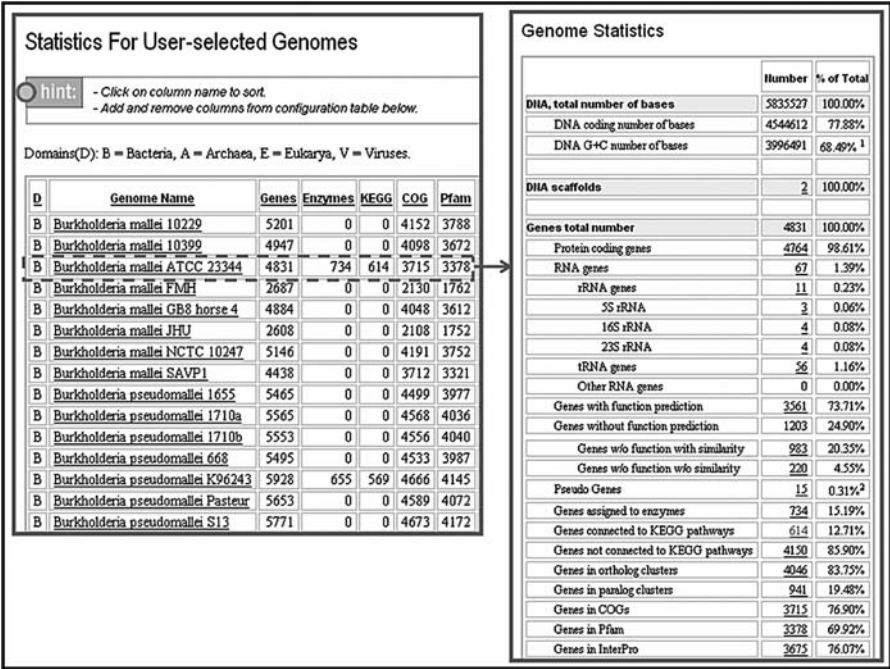


Fig. 7. Examining organism statistics for *Burkholderia mallei* and *Burkholderia pseudomallei* strains.

and InterPro domains. Such summaries can be configured by selecting the properties that are of comparative interest.

Individual organisms can be further examined using the Organism Details that includes various statistics of interest, such as the number of genes in the organism that are associated with KEGG, COG, Pfam, InterPro, or enzyme information, as shown in the right panel of Fig. 7. For each organism one can also examine the associated list of scaffolds and contigs: for each coordinate range, a Chromosome Viewer allows displaying genes colored according to COG functional categories.

Individual COG pathways or general categories can be examined using the COG Browser, which provides a hierarchical listing of the COG general categories (i.e., amino acid transport and metabolism) and individual pathways (i.e., arginine biosynthesis). The COG Pathway or Category Details lists the COGs of the selected pathway/category and the number of organisms with genes that belong to these COGs. For a given COG, the “organism counts”

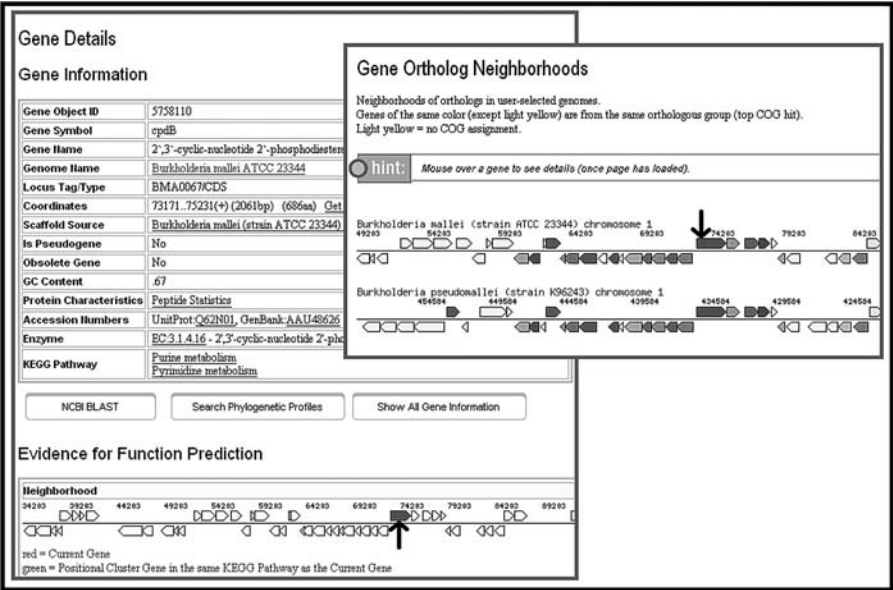
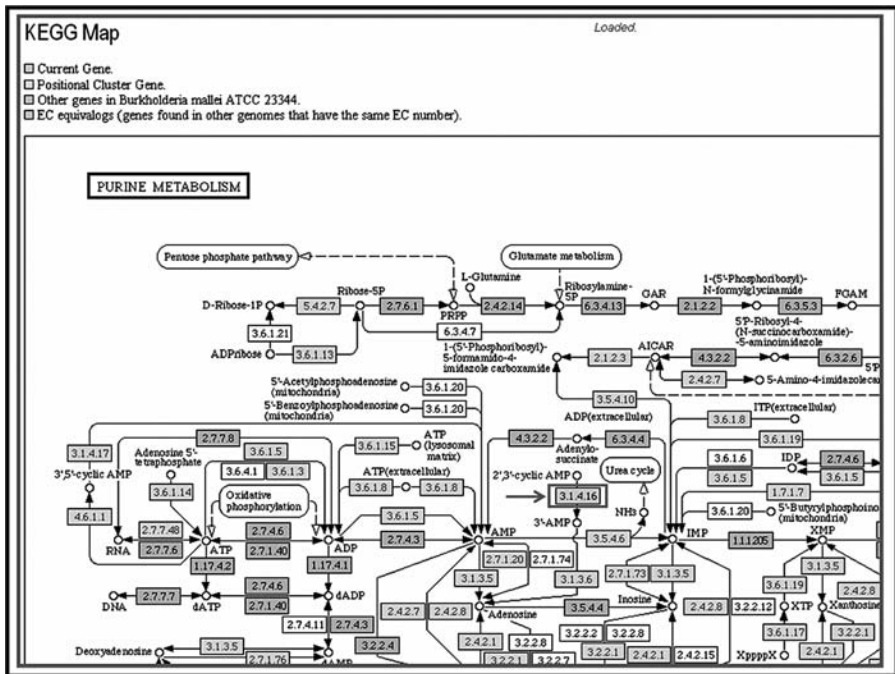


Fig. 8. Gene details and gene ortholog neighborhoods for a *Burkholderia mallei* gene.

are linked to a list of organisms and their associated “gene counts.” KEGG pathways can be explored in a similar manner using the KEGG Pathway Details.

Individual genes can be analyzed using Gene Details, as illustrated in Fig. 8. A Gene Information table includes gene identification, locus information, biochemical properties of the product, and associated KEGG pathways. Gene Details also includes evidence for the functional prediction: gene neighborhood, COG, InterPro, and Pfam, and precomputed lists of homologs, orthologs, and paralogs. The gene neighborhood displays the target gene with its neighboring genes in a 25-kb chromosomal window, as shown in Fig. 8, where the target gene is pointed out by an arrow.

The Gene Ortholog Neighborhoods, also shown in Fig. 8, includes the gene neighborhood of orthologs of the target gene (pointed out by an arrow) across several organisms: each gene’s neighborhood appears above and below a single line showing the genes reading in one direction on top and those reading in the opposite direction on the bottom; genes with the same color indicate association with the same COG group. For each gene, locus tag, scaffold coordinates, and COG group number are provided locally (by placing the cursor over the gene),



whereas additional information is available in the Gene Details associated with each gene.

A gene can be also examined in the context of its associated pathways, through links to KEGG maps available on the Gene Information table. On such a map, the EC numbers are color-coded and linked to the Gene Details for the associated genes, as illustrated in **Fig. 9**, which displays the Purine Metabolism KEGG map for the *B. mallei* gene shown in **Fig. 8** (pointed out by an arrow).

4.5. Gene Model Validation

The following example illustrates how occurrence profile results can assist in gene model validation. Consider the *B. mallei* and *B. pseudomallei* genomes previously mentioned. The result of the Phylogenetic Profiler indicates that, although *B. mallei* is approx 20% smaller than *B. pseudomallei* (4764 vs 5855 protein coding genes, respectively), it has 548 unique genes (*see* [Fig. 6](#)). This high number of unique genes (more than 11.5% of the total number of predicted

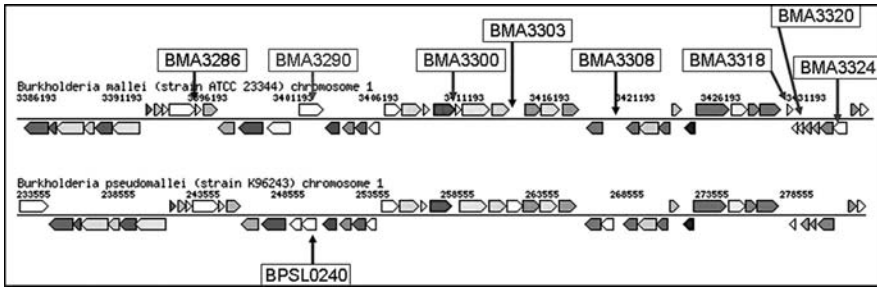


Fig. 10. Gene ortholog neighborhoods for a region of *Burkholderia mallei* and *Burkholderia pseudomallei*.

genes) suggests that a large percentage of the coding capabilities of *B. mallei* is distinct compared to *B. pseudomallei*. However, examining these genes using IMG's Ortholog Neighborhoods, as illustrated in Fig. 10, suggests that most of the differences in gene content between *B. mallei* and *B. pseudomallei* are owing to inconsistencies of the gene models. Detailed analysis of these 548 genes subsequently revealed that:

1. Genes *BMA3300*, *BMA3308*, *BMA3320*, and *BMA3324* appear as unique in *B. mallei*, although each of them has an ortholog in *B. pseudomallei*; these *B. mallei* genes seem to be unique because their ortholog in *B. pseudomallei* was not identified as a valid gene.
2. Genes *BMA3286* and *BMA3303* in *B. mallei* and *BPSL0240* in *B. pseudomallei* are functional genes that were erroneously identified as pseudogenes because they supposedly contain authentic frameshifts or stop codons; analysis of their BLAST hits against orthologs in other *Burkholderia* genomes available in IMG shows that they encode full-length proteins with no frameshifts or stop codons and their identification as pseudogenes was based on the alignment to multidomain homologs–fusion proteins.
3. Gene *BMA3290* indicates a gene in *B. mallei*, which is longer than all its homologs and is likely to have an incorrect start codon; indeed, analysis of this region and its comparison to the regions of synteny in other *Burkholderia* genomes shows that the start codon of *BMA3290* is incorrect; moreover, a gene in a different frame was missed as a result of erroneous prediction of the gene start.

Although Phylogenetic Profiler shows that *B. mallei* and *B. pseudomallei* have 10 different genes in this region, in fact there is only a 2-gene difference: a transposase in *B. mallei*, which is absent from *B. pseudomallei* and an ortholog of *BPSL0240*, which is a pseudogene in *B. mallei*. Thus, the comparative analysis of the genes in *B. mallei* and *B. pseudomallei* indicates an

up to 90 % error rate (either false-positive genes in one genome or false-negatives in the other genome) in the results because of the difference in gene prediction algorithms used to identify coding sequences in these two genomes.

5. Conclusion

Effective microbial genome data analysis across biological data management systems involves providing support for comparative analysis in an integrated data context. We presented the comparative analysis capabilities provided by the IMG system, in particular those that are based on occurrence profiles.

The comparative analysis capabilities in IMG are based on techniques that follow observed biological evolutionary phenomena regarding functional coupling of genes (10). Some IMG tools have similarities to analogous tools in microbial genome data analysis systems such as WIT (16), ERGO (17), MBGD (18), SEED (19), Microbes Online (20), and PUMA2 (21). However, IMG has also a number of unique comparative analysis capabilities. Thus, instead of restricting users to a predefined collection of metabolic pathways compiled from the literature and mostly comprising model organisms, IMG provides users with the opportunity to define their own pathways and functional categories by employing Analysis Carts regardless of existing annotations. Such user-defined pathways can be further analyzed using a variety of tools, such as COG, Enzyme, and Pfam Profiles, and the Phylogenetic Profiler. These tools were specifically developed in order to enable the analysis of genomes that are poorly characterized, are phylogenetically distant from model organisms, and cannot be analyzed efficiently using traditional pathway databases.

The first version of IMG was released in March 2005, followed by quarterly releases consisting of data content updates and analytical tool extensions. A data warehouse framework was used in developing IMG, and was found to provide an effective environment for developing a system that needs to support the integration and management of data from diverse sources, where data are inherently imprecise and tend to evolve over time. The data warehouse environment has provided an established framework for modelling and reasoning about genomic data.

IMG data content extensions have focused on data quality in terms of the coherence of annotations, based on sound validation and correction procedures, as well as corroboration of annotations from other public microbial genome

data resources. IMG's occurrence profile tools have proved to be effective in the detection and subsequent correction of annotation errors.

We plan to further enhance the occurrence profile tools in IMG. First, we plan to extend the occurrence profile based selection to include additional biological objects, such as gene clusters (e.g., COGs), enzymes, and chromosomal gene clusters. Note that unlike the profile-based selection of genes, no target organism needs to be selected for functional features such as COGs and enzymes that are common to all organisms. To support the selection of chromosomal gene clusters, we plan to extend the content of IMG by precomputing these clusters. Second, we plan to develop improved occurrence profile viewers in order to increase their usability. For example, we are considering presenting occurrence profile results in a hierarchical (tree) phylogenetic context, which would enhance these tools' ability to support examining biological phenomena of interest, such as gene loss and lateral gene transfer. The existing phylogenetic distribution viewer (*see* [Fig. 5 \[iv\]](#)), lays out the taxonomy of each organism in a text-based format, which has expressivity limitations. A more intuitive, and therefore more effective, way to represent this type of information in a phylogenetic context could be based on the 16S ribosomal RNA tree.

IMG will continue to be extended through quarterly updates, whereby it aims at continuously increasing the number of genomes integrated in the system from public resources and JGI, following the principle that the value of genome analysis increases with the number of genomes available as a context for comparative analysis. IMG will also continue to address the needs of the scientific community for comprehensive data content and powerful, yet intuitive, comparative analysis tools.

Acknowledgments

We thank Krishna Palaniappan, Ernest Szeto, Frank Korzeniewski, Iain Anderson, Natalia Ivanova, Athanasios Lykidis, Kostas Mavrommatis, Phil Hugenholtz, Anu Padki, Kristen Taylor, Xueling Zhao, Shane Brubaker, Greg Werner, and Inna Dubchak for their contribution to the development and maintenance of IMG. With their comments and suggestions, Krishna Palaniappan and Iain Anderson helped improve the examples in this chapter. Eddy Rubin and James Bristow provided, support, advice, and encouragement throughout the IMG project. IMG uses tools and data from a number of publicly available resources, their availability and value is gratefully acknowledged. The work presented in this paper was supported by the Director, Office of Science, Office of Biological and Environmental Research, Life Sciences Division, US Department of Energy under contract no. DE-AC03-76SF00098.

References

- 1 Liolios, K., Tavernarakis, N., Hugenholtz, P., and Kyrpides, N. C. (2006) The Genomes On Line Database (GOLD) v.2: a monitor of genome projects worldwide. *Nucleic Acid Res.* **34**, D332–D334.
- 2 Bateman, A., Coin, L., Durbin, R., et al. (2004) The Pfam Protein Families Database. *Nucleic Acids Res.* **32**, D138–D141.
- 3 Mulder, N. J., Apweiler, R., Attwood, T. K., et al. (2005) InterPro, progress and status in 2005. *Nucleic Acids Res.* **33**, D201–D205.
- 4 Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997) A genomic perspective on protein families. *Science* **278**, 631–637.
- 5 Marchler-Bauer, A., Panchenko, A. R., Shoemaker, B. A., Thiessen, P. A., Geer, L. Y., and Bryant, S. H. (2002) CDD: a database of conserved domain alignments with links to domain three-dimensional structure. *Nucleic Acids Res.* **30**, 281–283.
- 6 Kanehisa, M., Goto, S., Kawashima, S. Okuno, Y., and Hattori, M. (2004) The KEGG resource for deciphering the genome. *Nucleic Acids Res.* **32**, D277–D280.
- 7 Gene Ontology Consortium. (2004) The Gene Ontology Database and Informatics Resource. *Nucleic Acids Res.* **32**, 258–261.
- 8 Kersey, P., Bower, L., Morris, L., et al., (2005) Integr8 and genome reviews: integrated views of complete genomes and proteomes. *Nucleic Acid Res.* **33**, D297–D302.
- 9 Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2005) NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts, and proteins. *Nucleic Acid Res.* **33**, D501–D504.
- 10 Bowers, P. M., Pellegrini, M., Thompson, M. J., Fierro, J., Yeates, T. O., and Eisenberg, D. (2004) Prolinks: a database of protein functional linkages derived from coevolution. *Genome Biol.* **5**, R35.
- 11 Hauser, L., Larimer, F., Land, M., Shah, M., and Uberbacher, E. (2004) Analysis and annotation of microbial genome sequences. *Genet. Eng.* **26**, 225–238.
- 12 Markowitz, V. M., Korzeniewski, F., Palaniappan, K., et al. (2006) The Integrated Microbial Genomes (IMG) system. *Nucleic Acids Res.* **34**, D344–D348.
- 13 BioPAX. (2006) Biological Pathways Exchange. <http://www.biopax.org/>.
- 14 Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D., and Yeates, T. O. (1999) Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc. Natl. Acad. Sci.* **96**, 4285–4288.
- 15 Osterman, A. and Overbeek, R. (2003) Missing genes in metabolic pathways: a comparative genomic approach. *Chem. Biol.* **7**, 238–251.
- 16 Overbeek, R., Larsen, N., Pusch, G. D., et al. (2000) WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Res.* **28**, 123–125.
- 17 Overbeek, R., Larsen, N., Walunas, T., et al. (2003) The ERGO genome analysis and discovery system. *Nucleic Acid Res.* **31**, 164–171.

- 18 Uchiyama, I. (2003) MBGD: microbial genome database for comparative analysis. *Nucleic Acid Res.* **31**, 58–62.
- 19 Overbeek, R., Begley, T., Butler, R. M., et al. (2005) The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acid Res.* **33**, 5691–5702.
- 20 Alm, E. J., Huang, K. H., Price, M. N., et al. (2005) The microbes online web site for comparative genomics. *Genome Res.* **15**, 1015–1022.
- 21 Maltsev, N., Glass, E., Sulakhe, D., et al. (2006) PUMA2: grid-based high-throughput analysis of genomes and metabolic pathways. *Nucleic Acids Res.* **34**, D369–D372.

WebACT

An Online Genome Comparison Suite

James C. Abbott, David M. Aanensen, and Stephen D. Bentley

Summary

Comparison of related genomes is an enormously powerful technique for explaining phenotypic differences and revealing recent evolutionary events. Genomes evolve through a host of mechanisms including long- and short-range intragenomic rearrangements, insertion of laterally acquired DNA, gene loss, and single-nucleotide polymorphisms. The Artemis Comparison Tool (ACT) was developed to enable the intuitive visualization of the consequences of such events in the context of two or more aligned genomes. WebACT is an online resource designed to allow the alignment of up to five genomic sequences within the ACT environment without the need for local software installation. Comparisons can be carried out between uploaded sequences, or those selected from the EMBL or RefSeq databases, using BLASTZ, MUMmer, or Basic Local Alignment Search Tool (BLAST). Precomputed comparisons can be selected from a database covering all the completed bacterial chromosome and plasmid sequences in the Genome Reviews database ([1](#)). This allows the rapid visualization of regions of interest, without the need to handle the full genome sequences.

Here, we describe the process of using WebACT to prepare comparisons for visualization, and the selection of precomputed comparisons from the database. The use of ACT to view the selected comparison is then explored using examples from bacterial genomes.

Key Words: BLAST; MUMmer; BLASTZ; genome; comparison; visualization; database; precomputed; bacteria; plasmid; chromosome.

1. Introduction

The study of the similarities and differences between the genomic organization of a number of related bacterial species and strains provides a valuable means of inferring evolutionary relationships. It is especially useful when comparing, for example, related bacterial strains with varying degrees of pathogenicity, because the differences can often point to mechanisms by which the pathogen may be adapting to a particular niche within the host. Similarly the comparison of genomes from soil or marine bacteria may give insights into how the genome has evolved to adapt to a particular nutrient supply. The Artemis Comparison Tool (ACT) allows the visualization of such genetic differences but can also help us to understand how the differences have been generated, be it by intragenomic recombination or by interaction with external sources of DNA.

1.1. Sequence Comparisons

The comparison of two sequences to identify regions of similarity by searching for a series of bases which are the same, or at least, highly similar, is a fundamental process in biological sequence analysis. Sequence alignments can be split into two categories: global alignments, where the sequences are aligned with the maximum number of matching bases along their full length, and local alignments, where the best subsequences matches are identified. Global alignments are most appropriate for comparisons between fairly similar sequences of a similar length, i.e., different bacterial strains from the same genus, whereas local alignments are useful for sequences, which have regions of similarity interspersed with dissimilar regions, genomic rearrangements, or differing lengths (2). Algorithms to determine optimal global and local alignments using a technique known as dynamic programming were developed by Needleman and Wunsch (3), and Smith and Waterman (4), respectively. Dynamic programming assesses each pair of bases in the two sequences, and assigns the pair a score obtained from a matrix of predetermined scores. Matching bases are assigned positive scores, whereas mismatches incur a negative score. Gaps can also be inserted in the alignment, but at the cost of an additional negative score for each gap position. The optimal alignment is that which has the highest score once all the possibilities have been assessed. Various improvements have been made to these algorithms over the years, including decreasing the number of steps required in the algorithm, and the introduction of affine gap penalties (where a penalty is applied for opening a gap, and a second lower penalty for each time it needs to be extended), resulting in improvements in the quality of the alignments, memory requirements and the speed of the computations (discussed in ref. 2.). Despite these enhancements, determination of long alignments using

these methods was still not practical because of the high memory requirements and compute time required. Additionally, these algorithms may not be reliable when aligning homologous sequences with long insertions or deletions, because the gap penalties assigned may not be biologically meaningful (5).

To satisfy the requirement for improved performance in sequence alignment, algorithms, which use a heuristic approach were introduced, i.e., an approach which always locates similar regions, but does not guarantee an optimal alignment. By far the best known of these is BLAST, the Basic Local Alignment Search Tool (6), which was developed for searching databases for related sequences, but can also perform pairwise alignments. BLAST gains a considerable performance increase by identifying “seed matches,” the location of “words,” which are common to the two sequences, where a word is simply a subsequence of a defined length. Each of these seed matches is then extended using an algorithm related to dynamic programming, vastly reducing the number of alignments that need to be calculated (6). BLAST includes a number of other performance optimizations (summarized in ref. 7). Although far faster than Smith-Waterman alignments, BLAST still has a run-time that does not scale in a linear fashion with sequence length, and can have excessive memory requirements when applied to genome-scale sequence comparisons (i.e., ref. 8).

A number of algorithms have been introduced in recent years, which take different approaches to solving the problem of full-genome alignments (reviewed in ref. 9). BLASTZ (10), for example, uses the same overall approach as BLAST, by finding short seed matches, which are then extended to form gapped alignments. A number of differences make BLASTZ more appropriate for comparing genomic sequences, however, including the use of an empirically derived scoring matrix, the option of only including matching regions that occur in the same order and orientation, and a number of performance enhancements specifically targeted at long genomic sequences (10,11). In place of BLASTs locally optimized Smith-Waterman style alignments, BLASTZ uses an “X-drop” approach designed to avoid the inclusion of comparatively poor internal segments of alignments (12). Additionally, BLASTZ is implemented in such a way that the amount of memory available should never prove limiting (10).

A somewhat different approach is taken by MUMmer, a fast global alignment algorithm. MUMmer uses a data structure known as a suffix-tree to quickly identify all subsequences longer than a specified cut-off that are identical between the two sequences (13). These matches can then be clustered, allowing sequences containing substantial genomic rearrangements to be aligned.

The initial anchor matches are then chained together to create a set of anchors, reducing the size of the alignment problem (13). The latest version of the software (MUMmer 3) no longer requires the initial subsequence matches to be unique, improving identification of repeat regions (8).

Obtaining a comparison is only half the battle, however. The programs previously discussed produce textual outputs, in various formats. Direct interpretation of these is time-consuming, and can be complicated. Displaying these results in a graphical form provides a far more readily interpreted set of results.

1.2. The Artemis Comparison Tool

ACT (14) is an interactive, graphical DNA sequence comparison viewer, which permits the visualization of pairwise comparisons created using BLASTN and TBLASTX. The output of other algorithms, such as BLASTZ or MUMmer previously discussed, can also be used, but requires the data to undergo an additional software reformatting stage. Although sequence comparisons for ACT are performed in a pairwise manner, multiple comparisons between a number of sequences can be stacked. For example, a three-way comparison can be visualized where pairwise alignments have been performed between sequences 1 and 2, and sequence 2 and 3. The order of the sequences in such multiway comparisons can have a significant impact upon the interpretation of the results, because regions of similarity between sequences 1 and 3 are not explicitly identified in the previously described example. A thorough analysis of a group of sequences will therefore require the comparisons to be visualized with a range of sequence orders, necessitating the production of a greater number of comparisons, and increasing the complexity of the operation from a user perspective.

1.3. WebACT

WebACT (<http://www.webact.org>) is designed to permit biologists to visualize comparisons between multiple genomic sequences (15). Comparisons can either be selected from a database of precomputed comparisons, generated on-the-fly from submitted sequences, or reloaded from previous WebACT comparisons. The WebACT workflow is illustrated in Fig. 1., which will be referred to throughout the following methods. Up to five sequences can be included in a comparison. ACT can be launched directly from WebACT, with the selected sequences and comparisons automatically loaded. WebACT results can be saved for use offline with a standalone copy of ACT, or reloading into WebACT at a later date.

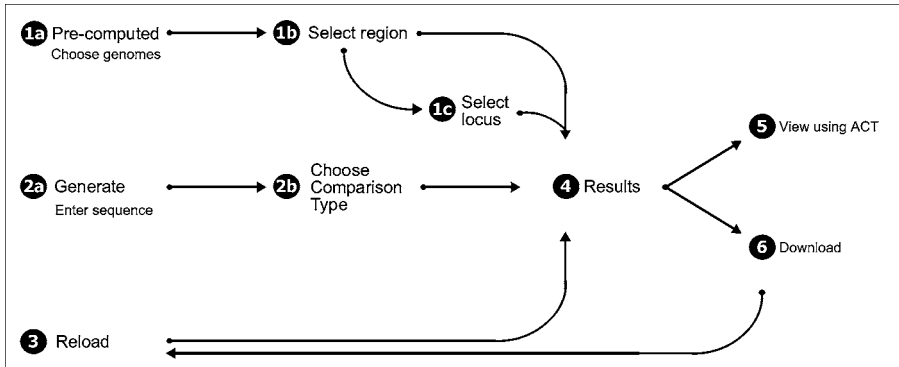


Fig. 1. The WebACT workflow.

The WebACT database gives access to precomputed comparisons between the sequences of the EBI's Genome Reviews database (*1*). Genome Reviews contains completed genomic sequences (either chromosomal or plasmids), which carry more consistent annotations than those found in the corresponding EMBL or Genbank entries. Precomputed comparisons between these sequences are carried out using BLAST, after "chunking" the sequences into 100-kb fragments with a 1-kb overlap (to avoid the problems associated with running BLAST on long sequences), using an all-against-all approach. Selection of a precomputed comparison is a two-stage process, where first, the sequences to be included in the comparison are selected, then the regions of those sequences is specified (**Fig. 1**). It is not necessary to visualize complete genome sequences when using the WebACT database, indeed in many cases it is preferable not to. A five-way comparison consisting of full-length genomic sequences can result in more than 60 Mb of data being downloaded to a client computer, which can be an issue when using older hardware or low-speed network connections. WebACT instead allows a region of a comparison to be selected according to the genomic location (in bases), or alternatively a region can be defined as a specified flanking region surrounding a named gene.

Generation of on-the-fly comparisons can be carried out between up to five sequences, using a choice of BLASTZ, MUMmer, or National Center for Biotechnology Information BLAST. A series of preconfigured settings are available tailored to specific kinds of queries, i.e., sequences less than 1 Mb or closely related sequences, however the application also allows full access to the available parameters of each program, enabling experienced users to customize comparison parameters.

2. Materials

1. Windows PC, Apple Mac (OS X), or UNIX computer with internet access.
2. Web browser: WebACT has been tested using the following browsers: Mozilla Firefox 1.5, Internet Explorer 6, Opera 8.1, and Konqueror (Linux only). JavaScript needs to be enabled within the browser to ensure full functionality of the interface.
3. Java Runtime Environment including Java Web-Start. ACT is implemented using the Java programming language, and requires a Java Runtime Environment (JRE), v1.4 or newer, to be installed on the users computer. Java Web Start is a technology that permits Java programs from a remote server to be run on the local machine. Java is available from <http://www.java.com>, with instructions on installation.

3. Methods

Worked examples are used to describe the use of WebACT with both prebuilt comparisons from the WebACT database, and comparisons generated on-the-fly.

1. The visualization of a comparison between three *Bordetella* genomes from the WebACT database. Viewing full-length genome comparisons will be demonstrated, as will the selection of the region surrounding a particular gene (*ampG*).
2. On-the-fly comparison of two gene clusters from *Streptococcus pneumoniae* for the biosynthesis of differing polysaccharide capsule structures. Sequences will be selected from the public databases for the generation of comparison files and subsequent visualization in ACT.

3.1. The WebACT Interface

WebACT can be accessed by visiting the address <http://www.webact.org> using a supported web browser. The page is laid out with a navigation bar along the top (Fig. 2), which provides access to the different methods of obtaining a comparison. Online documentation and examples are available by clicking on the “Instructions” tab. Throughout the WebACT interface, pop up tool-tips are available containing additional help regarding the use of particular features.



Fig. 2. WebACT's navigation bar.

3.2. Prebuilt Comparisons: *Bordetella*

This example demonstrates the selection of a comparison between three *Bordetella* genomes from the WebACT database, and the visualization of both the full-genome comparisons and the region surrounding the *ampG* gene.

3.2.1. Selection of Sequences

1. From the WebACT homepage (<http://www.webact.org>), click the “Pre-computed” tab to view a comparison from the database. The “Sequence selection” page will be displayed (Figs. 3 and 1A).
2. The number of sequences to include in the comparison is selected using the menu labeled “How many sequences do you wish to compare?” at the top of the page—select “3” from this menu. The page will be updated to display a series of menus allowing the selection of three sequences
3. It is necessary to select the genus of interest prior to selecting the sequences. For comparisons where all the sequences are from the same genus, an option is available in the “Selection Options” at the top of the page (“Select sequences from single genus”) to present a single “Genus” menu, which applies to all the sequences in the comparison. Select this option. The page will be updated to display a single “Genus” menu.
4. Select “*Bordetella*” from the “Genus” menu. The page will be updated to display a list of the *Bordetella* sequences from the database in each of the “Sequence” menus.
5. A separate “Sequence” menu is present for each sequence to be included in the comparison. Each entry on the “Sequence” menu includes the strain the sequence

WebACT | Select Sequences [Contact us](#)

How many sequences do you wish to compare? 2

Selection Options

Include plasmid sequences ☐

Select sequences from single genus ☐

Sequence 1 -

Genus Acinetobacter (1 sequence)

Sequence Acinetobacter sp. (strain ADP1) chromosome (CR543861)

Sequence 2 -

Genus Acinetobacter (1 sequence)

Sequence Acinetobacter sp. (strain ADP1) chromosome (CR543861)

Next

Fig. 3. The Prebuilt comparison sequence selection page.

Fig. 4. Selection of sequence ranges for a prebuilt comparison.

was obtained from, and the Genome Reviews accession number for this sequence. Select the following sequences from the “Sequence” menus:

- a. Sequence 1: *Bordetella pertussis* (BX240248).
- b. Sequence 2: *Bordetella bronchiseptica* (BX470250).
- c. Sequence 3: *Bordetella parapertussis* (BX470249).

Click the “Next” button to continue to select the sequence regions to include in the comparison. A new page will be displayed allowing selection of the sequence region (see [Figs. 4](#) and [1B](#)).

3.2.2. Selection of Precomputed Comparison Sequence Region

1. It is possible to define a single set of criteria, which are applied to all the selected sequences, specifying the region of the sequences to be displayed. Alternatively, a separate set of criteria can be defined for each sequence. In this instance, we wish to apply the same criteria to all the sequences, so leave the “Set the same range for all sequences” option selected.
2. The default region to be displayed is “Full sequence.” Because we wish to view a comparison between the full genome sequences, leave this option selected, and click the “Next” button.
3. The “Results” page will be displayed ([Figs. 5](#) and [1\[3\]](#)).

3.2.3. Visualization of Precomputed Comparison Using ACT

1. At the top of the “Results” page is a graphical representation of the selection, with each sequence represented by a gray bar, the length of that is proportional to that of the selected sequence. Below this are a set of options that affect the comparison data to be loaded. The hits to be displayed can be restricted on the

WebACT | [Select Sequences](#) | [Select Region](#) | [Results](#) [Contact us](#)

Results

Overview of Selection (Mouse over for sequence details)

↓	BX470248	
↓↑	BX470250	
↑	BX470249	

Comparison Options

Open overview in separate window: ☐

Show hits outside selected region ☐

Select e-value cut-off:

Select score cut-off:

If your browser asks you either Open or Save a .jnlp file, select 'Open' to view the comparison

[Start ACT](#) [Download files](#)

Fig. 5. Results page showing a prebuilt comparison between three *Bordetella* genomes.

basis of both the e -value of the hit (the probability of the alignment occurring by chance), or the alignment score of the hit. Filtering out hits with low scores or high e -values is useful when visualizing full genome sequences, because a large number of low-scoring hits can obscure the large-scale organization of the genome. Increase the score cut-off by selecting “2500” from the “Select score cut-off” menu. Alternatively, the filters can be left on their default values, and the data filtered within ACT.

- Click the “Start ACT” button, which will run ACT using Java Web Start (*see Note 1*). The first time ACT is launched, the software will be downloaded, but this will then be stored on the local computer, so will not be downloaded again unless an updated version of the software is available. ACT is then launched, (**Fig. 1[4]**) and the selected sequences and comparisons are loaded. Comparison data can also be downloaded by clicking the “Download files” button for offline use or reloading into WebACT at a later date (*see Note 2* and **Fig. 1[5]**).
- When the ACT window opens the initial view shows the start of all the sequences, in this case corresponding to the origin of replication for the three genomes. Each genome is displayed as forward and reverse DNA strands with features such as coding sequences displayed as colored blocks. Coding sequences can be viewed on specific coding frames by selecting “Show Frame Lines” under the “Display” menu, though screen size can become an issue. The red blocks are a graphical representation of the comparison file corresponding to the coordinates of the matching region in each sequence with the color intensity relating to the strength of the

match. Where the matching region is inverted in one sequence the comparison block appears blue.

4. The simplest method for moving through the sequences is using the horizontal scroll bars above each entry. By default, the entries are locked so they will scroll together. Entries can be unlocked under the comparison view specific menu (available through a “right-mouse-click” in the comparison panel), which allows customization of the alignment view. There are several methods for moving to, or selecting, specific positions or features in the genomes based on some prior knowledge. These are found under the “Select” and “Goto” menus and are too numerous to describe here, except to say that the “Feature Selector” and “Navigator” are particularly useful (*see* **Note 3**, and **Subheading 3.2.1.4**). If a region or feature of interest has been located or selected in one genome, select “View Selected Matches” in the comparison view menu to view all the regions which match that region/feature. This will bring up a window listing all the relevant matches. Double clicking one of them will centralize it in the window.
5. The view can be zoomed using the scroll bar alongside each sequence panel. When zooming out to view large regions it is often advisable to reduce the number of matches displayed. If a filter was not preapplied via the webpage (in stage 1 of this procedure) the data can be filtered either by using the scroll bar to the right of the comparison view (which filters on length of match up to 999 bases), or by selecting “Set Score Cutoff” or “Select Percent ID Cutoffs” in the comparison view menu. If a filter was not preapplied, set the minimum score cut-off to greater than 2000 then proceed to zoom out to the whole genome view (**Fig. 6**). To speed up the redraw on these detailed images the annotated features can be deselected under the “Entries” menu prior to zooming out.
6. The three *Bordetella* genomes are clearly related and the ACT comparison reveals some interesting features of their evolution from a common ancestor (**16**). It is thought that *B. bronchiseptica* is closest to the ancestral genome, with the other two having undergone different levels of genome reduction and rearrangement. The rearrangements are more pronounced in *B. pertussis* as a result of recombination between the large numbers of insertion sequences in the genome. The genome reductions appear to relate to niche adaptation. Although all three species are pathogens that cause similar diseases, *B. bronchiseptica* has a broad host range and causes the mildest disease, *B. parapertussis* only causes disease in humans and sheep, whereas *B. pertussis* is strictly a human pathogen and is the etiological agent of whooping cough.

3.2.4. Selection and Visualization of the *ampG* Region

1. To focus on a particular region of the sequences, it is not necessary to create a new sequence selection. Instead, click the “Select Region” link at the top of the page (*see* **Figs. 5** and **1[1B]**). The “Select Region” page will be displayed again.

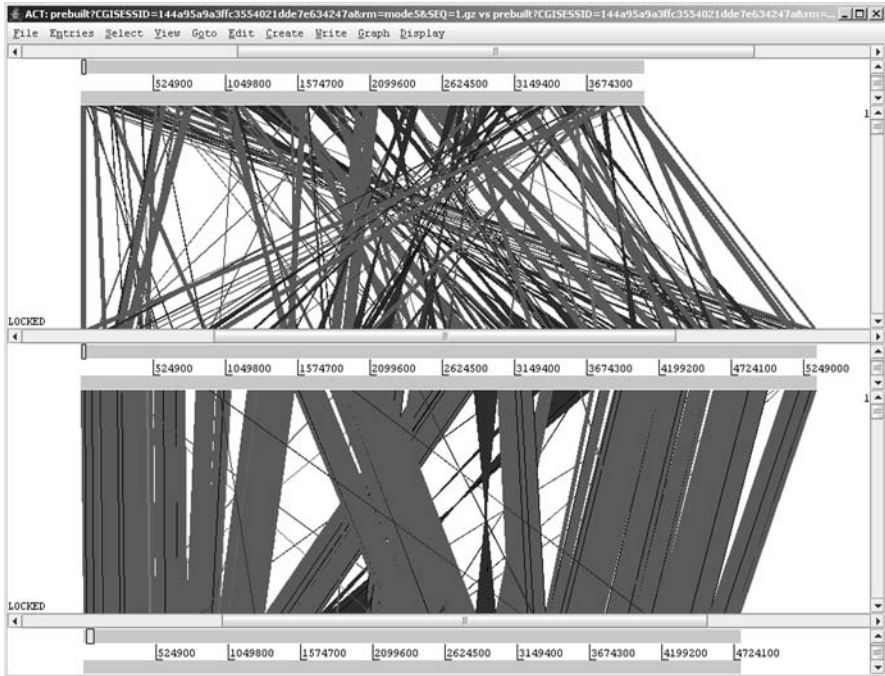


Fig. 6. *Bordetella* full genome comparisons viewed in ACT.

2. To select the region surrounding a named gene, it is necessary to enter the name of the gene in question in the “gene name” box. Rather than typing the gene name, a browseable list of the genes identified on the selected sequences is available. Click the “browse” button to open the “Browse Genes” window (see **Note 4**).
3. Scroll down the list to find “*ampG*,” select this gene and click the “Select” button. The selected gene name will be entered in the “gene name” box.
4. The amount of sequence to be included on either side of the selected gene is controlled by the adjacent option (“flanking sequence”). Change this value to 40,000, and click the “Next” button.
5. Unless the requested gene has more than one locus, the “Results” page will be displayed (see **Note 5**). The graphical overview of the sequence selection now shows three sequences of similar length. The location of the selected gene is indicated by a light blue marker on each sequence (see **Note 6**). Any previous changes made to the “Comparison Options” should have been retained. Reset the “Select score cut-off” to its default value of “250.”
6. The comparison of the *ampG* region is best viewed with the sequences in a different order from that used for the full genome comparison. Sequences can be reordered

from the “Results” page using the arrows to the left of the graphical overview. Click the “down” arrow adjacent to the top sequence (BX240248) on the graphical overview (see **Note 7**). This sequence should be seen to swap places with the sequence in the middle of the set (BX240250).

7. Click the “Start ACT” button to view the comparison. Again, the initial view is of the beginning of the three sequences. Scroll along to the *ampG* gene (all three sequences should be locked so will scroll together). You will see from the blue comparison blocks that, in the *B. pertussis* genome, the *ampG* region is inverted. To flip the *B. pertussis* sequence right-mouse-click in the either comparison view panel and select “Flip Subject Sequence,” or “Flip Query Sequence” as appropriate.
8. It is apparent that the *B. pertussis* genome has an insertion sequence in the promoter region of the *ampG* gene. This renders the promoter inactive. The gene encodes a specific permease that is involved in the recycling of a glycopeptide fragment released during normal cell wall turnover. The effect of this mutation is a build up of the glycopeptide in the supernatant. The glycopeptide is cytotoxic in cell culture and is commonly referred to as tracheal cytotoxin. Thus, the insertion sequence has subverted a housekeeping pathway to allow production of a pathogenicity determinant.

3.3. Comparison Generation: *S. pneumoniae*

The example describes the creation of a comparison between two entries uploaded into WebACT from the public DNA database. Each entry contains the DNA sequence and annotation for a gene cluster from *S. pneumoniae* encoding the biosynthesis of a particular polysaccharide capsule structure. Each strain of *S. pneumoniae* carries 1 version of the gene cluster out of a possible 90 (17). The different capsule types are conventionally determined by serotyping. The capsule forms the outer coating of these bacterial cells and differences in their structure affect interactions with the human host.

1. Select the “Generate” tab at the top of the page (see **Fig. 1[2A]**). The “Enter Query” page will be displayed.
2. As for prebuilt comparisons, the number of sequences to include in the comparison is selected using the menu labeled “How many sequences do you wish to compare?” at the top of the page—select “2” from this menu. The page will be updated to display data entry sections for each of the sequences to be included.
3. Running comparisons can take a significant amount of time, which is dependent upon the number and length of the sequences submitted, the algorithm selected, and the number of other users of the system. An e-mail notification can therefore be sent once the job has completed. To enable this option, enter an e-mail address in the “e-mail address” box.
4. In this example, the sequences to be compared will be selected from the EMBL database, by entering their accession numbers in the relevant boxes. Sequences

can also be provided by uploading sequences in EMBL or FASTA formats (*see Note 8*). Enter the following accession numbers into the following “Enter an EMBL or RefSeq accession number” boxes:

- a. Sequence 1: CR931649.
- b. Sequence 2: CR931652.

After entering the accession numbers, click the “Next” button at the bottom of the page.

5. WebACT permits a number of factors that affect how comparisons are carried out to be altered via the “Comparison Options” page (**Figs. 7** and **1[2b]**). A number of preconfigured comparison types are available, which are selected according to the choices made for the options labeled “Sequence Characteristics.” Alternatively, the choice of algorithm and parameters to be used can be defined by checking the “Show advanced options” box. In this case, because the sequences are only 17 kb long, select the option labeled “Are your sequences shorter than 1Mb?”
6. Click the “Submit” button to launch the comparison. While the comparison is running, a progress bar will be displayed, providing information regarding the current status of the job. Once the comparison has completed, the “Results” page will be displayed. If e-mail notification was requested, a link will be present in the mail, which is sent upon completion of the job and will load the “Results” page in the browser.
7. The results page is essentially the same as that presented for prebuilt comparisons, albeit with a reduced range of options. Click the “Start ACT” button to view the comparison using ACT.
8. The capsule gene clusters displayed are both less than 20 kb so the complete alignment can be viewed by zooming out one step (**Fig. 8**). These gene clusters are for serotypes 10A (top) and 10F (bottom). It is immediately clear from the comparison blocks that these gene clusters share extensive similarity in both DNA sequence and gene order. Click on a red block to see the match details displayed

WebACT | Enter Sequences | Set Options

Contact us

Set Comparison Options

Show advanced options ☐

An appropriate comparison type will be selected according to the sequences characteristics
Select 'show advanced options' to manually select comparison parameters

Sequence Characteristics

Are your sequences shorter than 1 Mb? ☐

Are your sequences 'distantly related'? ☐

Comparison type: BLASTZ: Fast comparison

Fig. 7. On-the-fly comparison options.

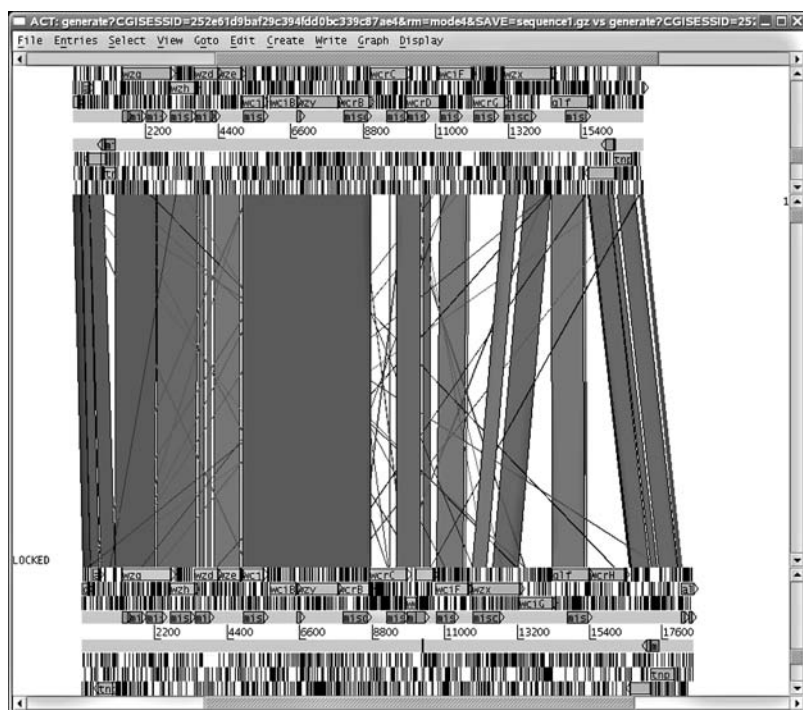


Fig. 8. Comparison of *Streptococcus pneumoniae* sequences from EMBL database viewed in ACT.

in the top left corner. It is also clear that some genes are present in one cluster but absent from the other. To view the annotation information, select a feature, then “View Selected Features” in the “View” menu. The 10A cluster includes a glycosyl transferase gene not present in 10F and the 10F cluster includes genes encoding a glycosyl transferase and an acetyl transferase not present in 10A. These enzymes are involved in the production of an oligosaccharide repeat unit which will be polymerized to form the mature capsule. The differential gene content of these clusters is reflected in the structure of the repeat unit synthesized by each serotype (17). The comparison also indicates where orthologous genes are present in both gene clusters but their sequences are divergent. In this case, the most divergent regions of the DNA sequence do not have red blocks assigned, though this view will vary according to the sensitivity of the search parameters. One interesting example is the gene with the locus_tag SPC10A_0012 from serotype 10A, and the equivalent gene from 10F, SPC10F_0012. These genes both encode glycosyl transferases and are located at the same position in each gene cluster, but the sequence divergence

in the 5' region may indicate differences in substrate specificity of the encoded enzymes.

4. Notes

1. WebACT will attempt to detect an installation of Java Web Start on the local computer, which is required to launch ACT directly from the website. A warning will be displayed on the “Results” page in the event that Web Start could not be detected, and a link to a page providing further information on installing Java Web Start will be displayed. If Web Start is correctly installed, clicking the “Start ACT” button results in a “jnlp” file being downloaded to the browser. Most browsers will ask whether this file should be opened or saved. If Web Start is correctly set up, clicking “open” will launch ACT.
2. A “Download files” button is displayed alongside the “Start ACT” button on the “Results” page, which allows the comparison to be downloaded as a zip file (Fig. 1[5]). This can be reloaded into WebACT at a later date, loaded into a standalone copy of ACT, or shared with colleagues. Zip files for comparisons that have been generated from submitted sequences will contain all the sequences and comparison files necessary to visualize the comparison, whereas those from prebuilt comparisons by default will only contain a small file containing a definition of the comparison, which can be used by WebACT to recreate the comparison when reloaded at a later date. Alternatively, when downloading a zip file from a prebuilt comparison, an additional option will be available labeled “Include data for offline use.” Enabling this option will result in the sequence and comparison files being included in the zip file to allow use with standalone ACT.

Reloading comparisons can be achieved by clicking on the “Reload” tab at the top of the page, selecting the file to reload and clicking the “Submit” button (Fig. 1[6]). Once the data has been uploaded, the “Results” page will be displayed.

It is also possible to view a generated comparison, or a prebuilt comparison saved using the “Include data for offline use,” without reloading the data into WebACT. The saved zip file must first be uncompressed into a new directory. If Java Web Start is correctly configured, double clicking on the file named “WebACT_comparison.jnlp” will load the comparison into ACT. Alternatively, if a standalone copy of ACT is installed on the local machine, the sequences and comparison files can be loaded manually by selecting “open” from the “File” menu within ACT.

3. Many functions in Artemis and ACT have shortcut keys, which are noted in the menus.
4. The lists of gene names are derived from the “gene_name” feature table qualifier in the Genome Reviews entries. A gene will therefore only appear on the list for a given genome if it has been annotated with that name in the database entry. When a region is being selected that applies to all the selected genomes (i.e., the “Set the

same range for all sequences” option is selected), the gene list will only contain genes that have been identified on all the selected genomes. Should a particular gene not be found in this list, selecting the “Set a different sequence range for each sequence” option will produce different lists of genes for each sequence selected. Be aware that the genome annotations included in the WebACT database are from the Genome Reviews database, and, therefore, do not correspond to the original database submissions. Genome Reviews supplies consist data appropriate to large-scale bioinformatics analysis. The drawback is that much of the useful biological information included in the initial annotation is likely to have been removed so it may be useful to refer to the original annotation.

5. In the event that a requested gene has more than one locus, an additional page will be presented after the “Select Region” page (**Fig. 1[1C]**). This will display a list of the different loci for the gene on each sequence, permitting the required locus to be selected. Certain genes may occur many times, i.e., 16S ribosomal RNA is found at 11 different locations in *Bacillus* genomes.
6. When a region is selected by gene name, the position of the gene on the sequence and the amount of flanking sequence requested may result in the required gene appearing off center in the graphical overview. This occurs when the gene is closer to one end of the sequence than the requested flanking sequence. In this case, the selection will be made from the requested gene to the end of the sequence. The amount of sequence selected, and location of the requested gene, is reported in the pop-up tool tip produced when the mouse pointer is placed over the sequence in the overview figure.
7. The order in which sequences are selected can have a significant affect upon the information that can be obtained from a comparison. For example, a three-way comparison consists of pairwise comparisons between sequence 1 and 2, and sequence 2 and 3. There is, therefore, no direct comparison being made between sequences 1 and 3. WebACT permits the order of the sequences to be adjusted for comparisons consisting of three sequences or more. The overview figure on the “Results” page will display up and down arrows adjacent to the sequence accession numbers. Clicking one of these arrows will move the sequence up or down one layer in the sequence “stack.” Although precomputed comparisons allow the instant reordering of sequences, for comparisons generated on-the-fly, it may be necessary for additional comparisons to be carried out to display the sequences in the new order. If it is known in advance that an on-the-fly comparison will be viewed using different sequence ordering, it is recommended to check the “Run extra comparisons to allow sequence reordering” option on the “Enter Query” page. This will ensure that all the possible pairwise comparisons are carried out in the first instance.
8. When uploading sequence files to generate a comparison, the volume of data to be transferred to the WebACT server can be considerable. If certain sequences in the comparison are present in the EMBL or RefSeq databases, try to use these in preference to uploading them, because this should produce much faster results.

If it is necessary to upload sequence files, these can be compressed using either WinZip, or the UNIX gzip utility, which will significantly reduce the time taken to upload the data. Submitted files should each contain a single sequence in EMBL or FASTA format. It is preferable to use EMBL/Genbank format for uploaded sequences, because any genes annotated in the feature table will then be displayed by ACT. Should multiple sequences be present in an uploaded file, only the first will be used.

Acknowledgments

This work was supported by the Faculties of Life Sciences and Medicine, Imperial College London and the Wellcome Trust.

References

- 1 Kersey P., Bower, L., Morris, L., et al. (2005) Integr8 and Genome Reviews: integrated views of complete genomes and proteomes. *Nucleic Acids Res.* **33**, 297–302.
- 2 Mount, D. W. (2001) *Bioinformatics Sequence and Genome Analysis*. Cold Spring Harbour Laboratory Press, Cold Spring Harbour, New York.
- 3 Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino-acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.
- 4 Smith, T. F. and Waterman, M. S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197.
- 5 Huang, W., Umbach, D. M., and Leping, L. (2006) Accurate anchoring alignment of divergent sequences. *Bioinformatics* **22**, 29–34.
- 6 Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) “Basic local alignment tool. *J. Mol. Biol.* **215**, 403–410.
- 7 Korf, I., Yandell, M., and Bedell, J. (2003) *BLAST*. O’Reilly and Associates, Sebastopol, CA.
- 8 Kurtz, S., Phillippy, A., Delcher, A. L., et al. (2004) Versatile and open software for comparing large genomes. *Genome Biol.* **5**, R12.
- 9 Chain, P., Kurtz, S., Ohlebusch, E., and Slezak, T. (2003) An applications-focused review of comparative genomics tools: capabilities, limitations and future challenges. *Brief. Bioinform.* **4**, 105–123.
- 10 Schwartz, S., Zhang, Z., Frazer, K. A., et al. (2000) PipMaker: a web server for aligning two genomic DNA sequences. *Gen. Res.* **10**, 577–586.
- 11 Schwartz, S., Kent, W. J., Smit, A., et al. (2003) Human-mouse alignments with BLASTZ. *Gen. Res.* **13**, 103–107.
- 12 Zhang, Z., Berman, P., Wiehe, T., and Miller, W. (1999) Post-processing long pairwise alignments. *Bioinformatics* **15**, 1012–1019.

- 13 Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. (1999) Alignment of whole genomes. *Nuc. Acids. Res.* **27**, 2369–2376.
- 14 Carver, T. J., Rutherford, K. M., Berriman, M., Rajandream, M. A., Barrell, B. G., and Parkhill, J. (2005) ACT: the Artemis Comparison Tool. *Bioinformatics* **21**, 3422–3433.
- 15 Abbott, J. C., Aanensen, D. M., Rutherford, K., Butcher, S., and Spratt, B. G. (2005) WebACT: an online companion for the Artemis Comparison Tool. *Bioinformatics* **21**, 3665–3666
- 16 Parkhill, J., Sebahia, M., Preston, A., et al. (2003) Comparative analysis of the genome sequences of *Bordetella pertussis*, *Bordetella parapertussis* and *Bordetella bronchiseptica*. *Nat. Genet.* **35**, 32–40.
- 17 Bentley, S. D., Aanensen, D. M., Mavroidi, A., et al. (2006) Genetic analysis of the capsular biosynthetic locus from all 90 pneumococcal serotypes. *PLoS Genet* **2**, e31.

GenColors

Annotation and Comparative Genomics of Prokaryotes Made Easy

Alessandro Romualdi, Marius Felder, Dominic Rose, Ulrike Gausmann, Markus Schilhabel, Gernot Glöckner, Matthias Platzer, and Jürgen Sühnel

Summary

GenColors (gencolors.fli-leibniz.de) is a new web-based software/database system aimed at an improved and accelerated annotation of prokaryotic genomes considering information on related genomes and making extensive use of genome comparison. It offers a seamless integration of data from ongoing sequencing projects and annotated genomic sequences obtained from GenBank. A variety of export/import filters manages an effective data flow from sequence assembly and manipulation programs (e.g., GAP4) to GenColors and back as well as to standard GenBank file(s). The genome comparison tools include best bidirectional hits, gene conservation, syntenies, and gene core sets. Precomputed UniProt matches allow annotation and analysis in an effective manner. In addition to these analysis options, base-specific quality data (coverage and confidence) can also be handled if available. The GenColors system can be used both for annotation purposes in ongoing genome projects and as an analysis tool for finished genomes. GenColors comes in two types, as dedicated genome browsers and as the Jena Prokaryotic Genome Viewer (JPGV). Dedicated genome browsers contain genomic information on a set of related genomes and offer a large number of options for genome comparison. The system has been efficiently used in the genomic sequencing of *Borrelia garinii* and is currently applied to various ongoing genome projects on *Borrelia*, *Legionella*, *Escherichia*, and *Pseudomonas* genomes. One of these dedicated browsers, the Spirochetes Genome Browser (sgb.fli-leibniz.de) with *Borrelia*, *Leptospira*, and *Treponema* genomes, is freely accessible. The others will be released after finalization of the corresponding genome projects. JPGV (jpgv.fli-leibniz.de) offers information on almost all finished bacterial genomes, as compared to the dedicated browsers with reduced genome comparison functionality, however. As of

January 2006, this viewer includes 632 genomic elements (e.g., chromosomes and plasmids) of 293 species. The system provides versatile quick and advanced search options for all currently known prokaryotic genomes and generates circular and linear genome plots. Gene information sheets contain basic gene information, database search options, and links to external databases. GenColors is also available on request for local installation.

Key Words: Genome analysis; genome comparison; bioinformatics; prokaryotic genomes.

1. Introduction

The first complete genome sequences of bacteria were reported for *Haemophilus influenza* and *Mycoplasma genitalium* in 1995 (1,2). Since then the number of known prokaryotic genomes has rapidly increased. As of January 25, 2006, the GOLD database (<http://www.genomesonline.org>) lists 273 completed and 914 ongoing prokaryotic genome projects (3). This quickly growing amount of information has led to increased biological insight for each individual genome. In addition, however, our knowledge can be greatly enriched by comparison of related genomes (4–6). This is particularly true for a better understanding of overall genome structure and for genome evolution. Moreover, genome comparison approaches are supposed to contribute to an acceleration and improvement of the annotation process of newly sequenced genomes. Even though the value of comparative genomics is widely recognized, the number of tools that offer up-to-date information on prokaryotic genomes with an emphasis on genome comparison is still small. Also, existing bioinformatics tools are often not particularly suitable for the bench biologist. We have, therefore, developed and describe here the software/database system GenColors that employs extensive genome comparison both for the analysis of finished genomes as well as for accelerated and accurate annotation of ongoing sequencing projects (7). Special emphasis was given to the development of easy to use and intuitive tools. Originally, GenColors (GENome COMparison by LOW Redundant Sequencing) was designed for the annotation and analysis of new genomes obtained by low-redundancy sequencing. However, the actual features of this system make GenColors a valuable tool for the annotation, analysis, and presentation of bacterial genomes from the earliest to the final stages of a sequencing project and also for setting up genome browsers for finished genomes. There are basically two different types of GenColors genome browsers. Dedicated browsers include a number of related genomes and make extensive use of genome comparison. On the contrary, the Jena Prokaryotic Genome Viewer (JPGV) offers information on all currently known prokaryotic genomes but has restricted genome comparison functionality.

2. Materials

Working with already installed GenColors tools, in-house or on the web, requires nothing else than a JavaScript-enabled web browser and Acrobat Reader for displaying PDF files. For local installation it is necessary to know that GenColors currently includes 86 Perl scripts and 4 Perl modules (www.perl.org). It requires a web server like Apache (www.apache.org), MySQL (www.mysql.com), BioPerl (bio.perl.org) (8), and EMBOSS (emboss.sourceforge.net) (9). Both for user database searches and for the generation of precomputed data the UniProt database (10) has to be locally available. All data is stored in 40 tables distributed over two relational database types. A central database contains data used by all GenColors derivatives. In a second database type information is stored that is specific to a certain GenColors-based genome browser. For speeding up server response some analyses as well as most of the scans against external databases are stored as precomputed data. Automated procedures manage the download process of the most recent versions of the UniProt database, the Basic Local Alignment Search Tool (BLAST) scans (11), and the functional assignment of genes according to the database of Clusters of Orthologous Groups (COGs) of proteins with the program COGNITOR (12).

3. Methods

3.1. Dedicated GenColors Browsers and JPGV

As mentioned in **Subheading 1.**, the GenColors system has been used to set up both dedicated browsers and the JPGV. The system has been efficiently used in the genomic sequencing of *Borrelia garinii* (13) and is currently applied to various ongoing genome projects on *Borrelia*, *Legionella*, *Escherichia*, and *Pseudomonas* strains. One of these dedicated browsers, the Spirochetes Genome Browser (SGB) (sgb.fli-leibniz.de) including *Borrelia*, *Leptospira*, and *Treponema* genomes, is currently freely accessible. The others will be released after finalization of the corresponding sequencing projects. Contrary to the small number of genomes included in the dedicated browsers, the JPGV (jpgv.fli-leibniz.de) offers information on 632 genomic elements (replicons) of 293 species and, thus, covers almost all currently known prokaryotic genomes. To date, we have not yet generated precomputed data for this large number of genomes. Therefore, some of the analysis options that will be described next are not available in JPGV. The functionalities of dedicated browsers and JPGV are listed in **Table 1**.

Table 1
Availability of Analysis Features in the Dedicated Genome Browsers^a

	Dedicated genome browsers	Jena Prokaryotic Genome Viewer
Gene information sheets	+	+
Gene lists	+	+
QuickSearch	+	+
Advanced search	+	+
Sequence search (PROSITE patterns)	+	+
Search via COG functional categories	+	+
BLAST search for similar protein or DNA sequences	+	+
Linear and circular genome plots	+	+
Links to external databases (taken from UniProt)	+	+
Best bidirectional hits	+	—
Gene core sets	+	—
Protein variations and analysis of synonymous and nonsynonymous base substitutions	+	—
Synteny analysis	+	—
Codon and amino acid usage	+	—
Precomputed UniProt hits	+	—

^aFor example, SGB (sgb.fli-leibniz.de), and in the JPGV (jpgv.fli-leibniz.de).

3.2. GenColors Features

3.2.1. Best Bidirectional Hits, Collinear Gene Partnerships, and DNA Sequence Similarity Search

For the analysis of gene catalogues and for a quantitative genome comparison the identification of homologous genes is of utmost importance. The typical bioinformatics approach is to identify such genes by DNA or protein sequence similarity. This approach is also adopted in GenColors. Putative orthologous genes in two different genomic elements are identified by best bidirectional BLAST hits (BBHs) of the corresponding protein sequences. The default sequence identity threshold parameter is 30%. In addition, the length ratio is required to be larger than 0.3. BBHs determined by this approach form the basis for further analyses on protein variation, gene core sets, and synteny. For

the protein pairs identified by a BLAST local alignment, a Needleman-Wunsch global alignment (14) is calculated subsequently adopting the EMBOSS program needle. An alignment viewer calculates statistical data and offers 13 different color schemes for highlighting specific amino acid patterns (see **Note 1**).

This protein sequence-based method is supplemented by two different approaches of DNA sequence comparison. The alignment of two collinear genomic elements allows the identification of potential gene relationships by similar gene localization. This analysis can possibly identify related gene pairs that are not found as protein sequence-based BBHs. The list generated by GenColors indicates whether the relationships identified at the DNA level, that we call gene partnerships, are also found as BBHs. Currently, this type of analysis is only available for the *Borrelia burgdorferi*/*B. garinii* genome pair. Finally, GenColors provides an option for BLAST sequence comparison of any DNA sequence with the browser genome sequences. This tool is especially useful for the analysis of non-genic sequence features. The output list indicates sequence range, scores, and other statistical data as well as full-length genes included in the aligned sequence range or genes that overlap in part with that range.

3.2.2. Protein Variations, Codon, and Amino Acid Usage

Protein sequence pairs identified as BBHs and aligned by the EMBOSS program needle are analyzed in more detail by the protein variations option. The analysis can be done for all protein-coding genes of pairs of complete genomes or of genomic elements as well as with user-defined lists. The output provides statistical information on amino acid insertions, deletions, duplications, and exchange and the alignments can be displayed by the alignment viewer previously mentioned.

The ratio of nonsynonymous to synonymous substitutions in a protein-coding gene may reflect the relative influence of positive or purifying selection and neutral evolution. Therefore, protein sequence information is supplemented by an analysis of synonymous and nonsynonymous base substitutions in the DNA sequences. The calculations are performed by means of the program Syn-SCAN (hivdb.stanford.edu/pages/synscan.html) (15) that adopts a method by Nei and Gojobori (16). The output list includes 10 statistical parameters (see **Note 2**) and in particular the measure $(S_d - N_d)/(S_d + N_d)$, where S_d and N_d stand for the observed synonymous and nonsynonymous substitutions, respectively.

Codon usage and the related amino acid usage data have been correlated with a number of genomic features mostly related to evolution (17) and more

recently to gene expression (18). Within GenColors, one can analyze these data both for individual genes and for complete genomic elements or genomes. In the latter case, a side-by-side comparison for two different species is possible and a start codon statistics is provided.

3.2.3. Gene Core Sets

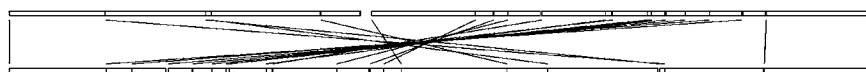
Gene core sets are defined as groups of genes with BBHs for all possible pairs of organisms in the data source. They represent the basic gene repertoire that is common to all genomes under study. The user can define different data sets ranging from two to all genomes included in a specific browser. Also, the sequence identity threshold can be varied. For example, for the genome pair *Leptospira interrogans* serovar *Copenhageni* str. *Fiocruz L1-130* (chromosome I) with a total number of 3436 genes and *Treponema denticola* ATCC 35405 with 2767 genes, the gene core set consists of 456 genes at a sequence identity threshold of 30% and is decreased to only 4 genes at a threshold of 70%. The four genes code for the ribosomal proteins S12, L34, and L36 and for the flagellar motor switch protein FliG. Again, the genes selected by the gene core set analysis can be stored in user-defined gene lists and thus used for further analyses (see Note 3).

3.2.4. Synteny Analysis and Gene Conservation

The term “synteny” describes some kind of similarity between genomic sequences. It was originally used to indicate the presence of two or more loci on the same chromosome (19). In comparative genomics analyses the term “conserved synteny” is widely used indicating the association of orthologous genes in two separate species often regardless of gene order (20). On the other hand, synteny has also been defined as conservation of DNA sequence and of gene order (5). For example, the SyntenyView of the Ensembl Genome Browser shows conservation of large-scale gene order between species pairs (21).

The GenColors system offers an option for an in-depth synteny analysis, which is based on BBHs between protein sequences. We define synteny groups as pairs of syntenic gene groups with a similar gene order on different genomic elements of either the same or of different species, potentially interrupted by up to five genes between each group member (see Note 4). The ordering of the syntenic gene groups on the two genomic elements that are compared may be completely unrelated. In some cases, a more regular pattern is observed, however. For example, the global synteny map of the chromosomes I of *Leptospira interrogans* serovar *lai* str. 56601 and of *Leptospira interrogans* serovar *Copenhageni* str. *Fiocruz L1-130* (22) shown in Fig. 1,

Leptospira interrogans serovar *lai* str. 56601



Leptospira interrogans serovar *Copenhageni* str. *Fiocruz* L1-130

Fig. 1. Global synteny map for the chromosomes I of *Leptospira interrogans* serovar *Copenhageni* str. *Fiocruz* L1-130 and *Leptospira interrogans* serovar *lai* str. 56601. Related syntenic gene groups from each chromosome thus forming a synteny group are linked by a line. When moving the mouse pointer over the boxes representing syntenic gene groups, the number of genes included and the sequence range are displayed.

exhibits conserved first and last synteny groups and a huge inversion in the remaining part of the genomes. The synteny group organization can also be displayed as dotplots for BBHs (Fig. 2A) or for syntenic gene groups (Fig. 2B). Table 2 shows an example for a gene list of a relatively small synteny group from this genome pair. Finally, there is an option for analyzing the gene order within the two syntenic gene groups of a synteny group. In Fig. 3, an example of inverted gene order is shown. Taken together, these options form the basis for a quick but nevertheless thorough synteny analysis that may be helpful in understanding genome structure and for annotation.

The gene conservation option is closely related to the synteny and is also based on BBHs. It provides information on a possible conservation between a gene of one species and all other genes of all browser genomes. As for the synteny analysis this information is determined from the BBHs. The option generates for all genes of a genome a table with the following information:

1. Is there a BBH to another protein in the other genomes included in the data set of a specific browser?
2. Is there a functional assignment of this gene (no occurrence of the terms hypothetical or putative in the description)?
3. Is the gene member of a synteny group?

In summary, the gene conservation option provides a compact overview on protein sequence similarities in all genomes included in a dedicated genome browser.

3.2.5. Gene Lists

Gene lists can be either generated by the gene list option for complete genomes with one or more genomic elements or are generated by search queries. They usually include information on gene name, locus tag, GenBank

description, genomic element, start position, length, strand, and GC-content. The genes can be listed according to all of these features. For the protein variations tool this feature list is even longer including also statistical parameters derived from a comparison of either protein or DNA sequences. In addition to the GenBank descriptions, the UniProt protein name of genes can be shown. Improved annotations are often available from UniProt for genomes annotated years ago. This tool provides thus a comprehensive overview of possible annotation changes in a genome by only one mouse click. The DNA or protein sequences of list genes can also be exported into a multi-FASTA file. Gene lists can be stored and used for further analysis including the generation of circular

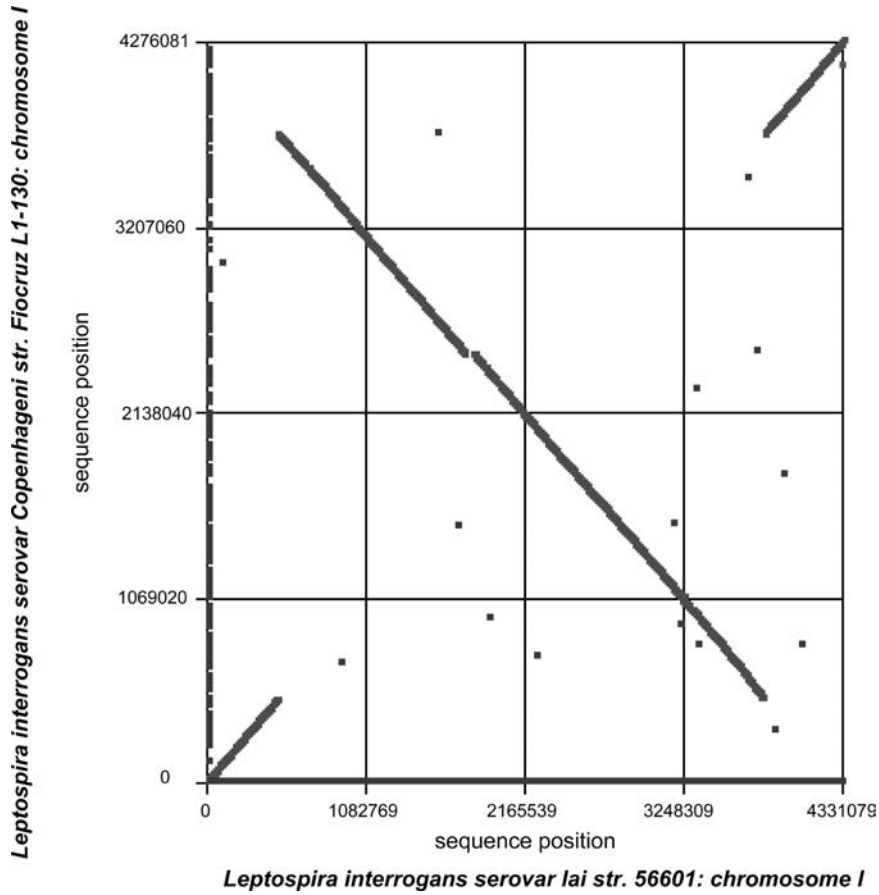


Fig. 2. (Continued)

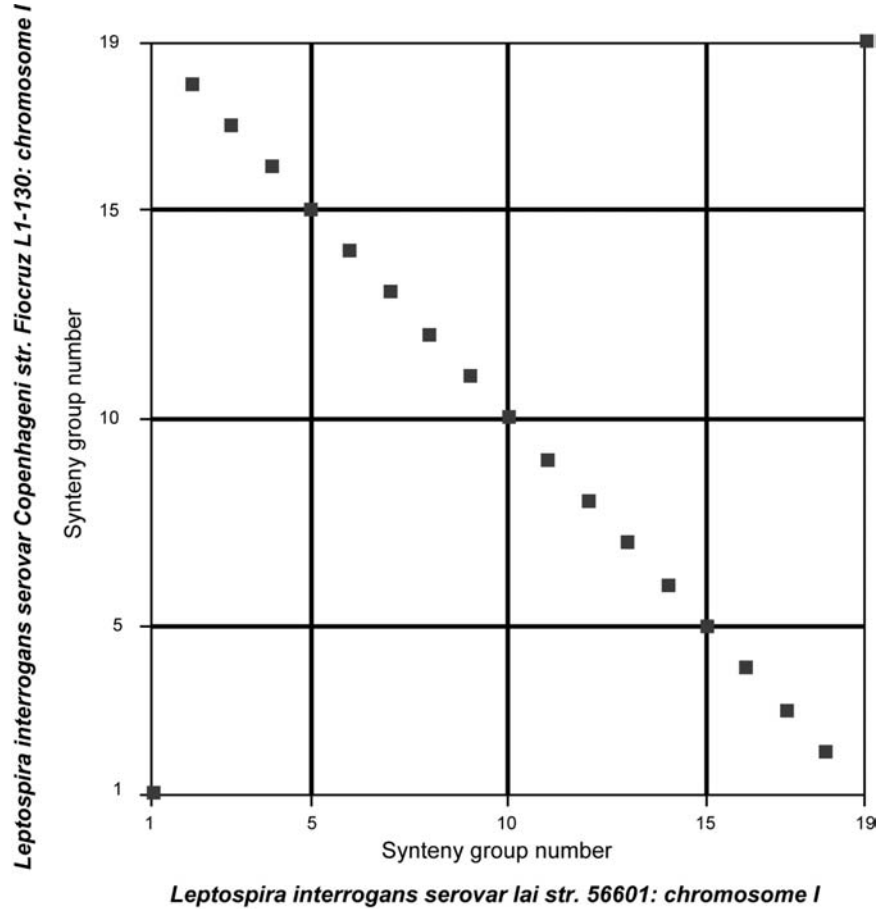


Fig. 2. Dotplots from a synteny analysis of the chromosomes I of *Leptospira interrogans* serovar lai str. 56601 and of *Leptospira interrogans* serovar Copenhageni str. Fiocruz L1-130. (A) Gene dotplot. Dots located on the axes represent genes with no best bidirectional BLAST hits (BBH) counterparts. Dots not located on axes stand for BBHs. In a color representation red dots indicate genes that are members of synteny groups, whereas green dots represent either genes having no BBHs or BBHs that are not synteny group members. On the axes the sequence positions are indicated. (B) Dotplot of syntenic gene groups. Each pair of syntenic regions or gene groups forming a synteny group is represented by one dot irrespective of its sequence length or the gene number included. On the axes the synteny group number is displayed.

Table 2**Genes of Synteny Group 234 for the Genome Pair *Treponema pallidum* and *Treponema denticola* ATCC 35405^a**

<i>T. denticola</i> ATCC 35405, chromosome (1,515,581–1,523,423 bp)	<i>T. pallidum</i> , chromosome (947,148–954,757 bp)
NA (TDE1470) (1), conserved hypothetical protein	TP0874 (1), conserved hypothetical protein
NA (TDE1469) (2), conserved hypothetical protein TIGR00150	TP0875 (2), conserved hypothetical protein
NA (TDE1468) (3), glycoprotease family protein	TP0876 (3), conserved hypothetical protein
NA (TDE1471) (4), conserved hypothetical protein	TP0873 (4), <i>T. pallidum</i> predicted coding region
NA (TDE1467) (5), HD domain protein	TP0877 (5), conserved hypothetical protein
NA (TDE1475) (6), flagellar filament core protein	TP0870 (6), flagellar filament 31 kDa core protein (flaB3)
NA (TDE1477) (7), flagellar filament core protein	TP0868 (7), flagellar filament 34.5 kDa core protein (flaB1)
fliD (TDE1472) (8), flagellar hook-associated protein 2	TP0872 (8), flagellar filament cap protein (fliD)
<i>Non-CDS genes and CDS genes with no BBHs inside this synteny group</i>	<i>Non-CDS genes and CDS genes with no BBHs inside this synteny group</i>
NA (TDE1473), flagellin FlaG, putative	
NA (TDE1474), hypothetical protein	TP0869, <i>T. pallidum</i> predicted coding region TP0869
NA (TDE1476), hypothetical protein	TP0871, <i>T. pallidum</i> predicted coding region TP0871

^aThe numbers in square brackets allow the unambiguous identification of related gene pairs.

or linear genome plots. Working with user-defined lists requires, however, an online registration. There are two exceptions to the list features previously described. The QuickSearch option provides only information on gene name, locus tag, GenBank description and genomic element, and the advanced search option of dedicated browsers returns the gene name, but in addition all BBHs and the best five UniProt hits for each individual gene. To customize the output list that may become very long it is possible to hide either the BBHs or the UniProt hits or both. Gene lists compiled with this latter option facilitates

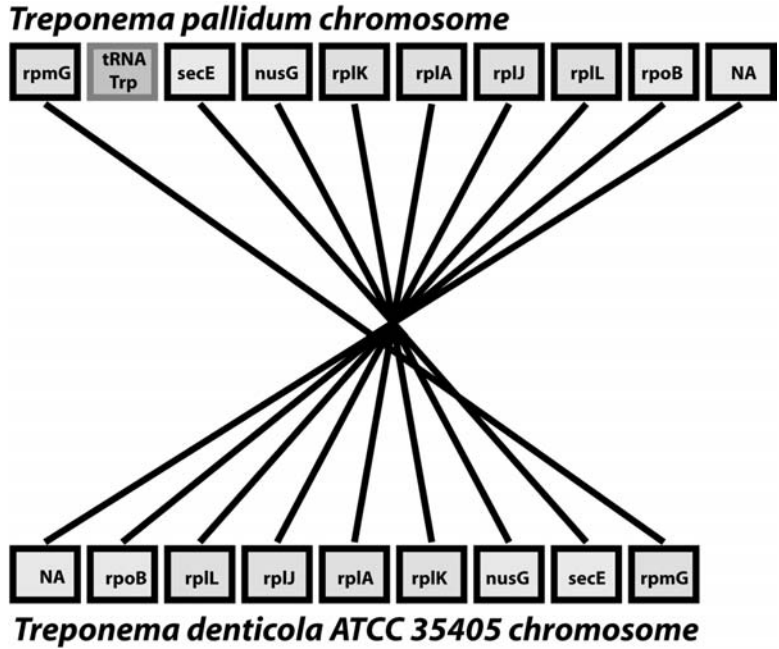


Fig. 3. Inverted gene order in a synteny group located in the sequence ranges 241.900–253.384 and 2.4445.050–2.456.382 of the *Treponema pallidum* and *Treponema denticola* ATCC 35405 chromosomes, respectively. The genes included are: rpmG (ribosomal protein L33), tRNA-Trp, secE (preprotein translocase subunit), nusG (transcription antitermination protein), rplK (ribosomal protein L11), rplA (ribosomal protein L1), rplJ (ribosomal protein L10), rplL (ribosomal protein L7/L12), rpoB (DNA-directed RNA polymerase, β -subunit), NA (putative DNA-directed RNA polymerase, β -subunit). In the original GenColors plot the genes are colored according to the corresponding Clusters of Orthologous Groups functional category. When moving the mouse pointer over a gene box, description, locus tag, strand, and sequence range is displayed.

reannotation because they provide information on BBHs and UniProt matches for possibly all genes in a genome by only one mouse click.

3.2.6. Gene Information Sheets

Gene information sheets summarize all data available for individual genes. On top, the sheet displays a zoomable graph showing the gene environment including also all other features indicated in the GenBank file such as pseudo-genes or signal peptides, for example. The genes are colored according to the

COG of proteins functional classification (12). Given quality data are available, they can be displayed as color-coded graphs of confidence (Phred score [23]) and coverage values (see the *B. garinii* genome in SGB). More detailed information is available from the basepair view, where the bases of the two DNA strands, the amino acids in the six frames and numerical confidence, and coverage data as well as a background coloring is shown for each individual base. There is, however, also a text view version. The menu bar below the gene environment graph offers information on BBHs, gene conservation, syntenies, Swiss-Prot or TrEMBL hits, DNA, or protein sequence BLAST hits within the browser database and codon and amino acid usage. Below this menu bar general gene information is provided that is obtained from the corresponding GenBank file or, for newly sequenced genomes, from the local annotators. For protein-coding genes both the GenBank description and the UniProt protein name are indicated. Of course, the DNA and protein sequences are displayed. Links to external databases, such as InterPro (24) or Gene Ontology (25), for example, are shown if the corresponding protein sequence is included in UniProt. In the remaining part of the information sheet BBHs to all other genomic elements included in the browser database and the five best UniProt matches are displayed. This directly accessible information may accelerate the annotation process substantially. The gene information sheets represent the main starting point for gene annotation.

3.2.7. Search Options

GenColors basically offers two ways of searching. By the QuickSearch option one can retrieve all genes that contain the search string entered in the gene name, locus tag, or description. On the other hand, the advanced search options allows the combination of 20 different search categories, such as gene type, name, description, note, length, geninfo id, locus tag, sequence coverage and confidence, CDS with wrong boundaries, organisms and genomic elements, COG functional categories, external databases, and identifiers of external databases. The latter two options are particularly interesting because they can be used to search for genes for which information in an external database is available. An example would be to identify all genes for which three-dimensional protein structures have been deposited at the Protein Data Bank (26). In addition to keyword-based search options it is also possible to search for sequence motifs adopting the PROSITE syntax (27). With these tools it was very simple, for example, to find out that there are currently about 200,000 hypothetical prokaryotic genes. Taken together, the GenColors search

options represent powerful means for querying the complete currently known “universe” of prokaryotic genomes.

3.2.8. Annotation With GenColors, Data Flow, and Output/Input Interfaces

GenColors can be effectively used for annotating newly sequenced genomes. It can import files in GenBank format both directly from GenBank or from assembly programs, such as GAP4 (28). If quality data are available they can be imported in a tab-delimited table format. After various analyses and preliminary annotations performed by GenColors sequence data of an ongoing genome project can be returned to the assembly program for further finishing including gap closure. We have developed the GenALA (GENome Assembly Linked Annotation) toolkit facilitating the data flow between the assembly program GAP4 and GenColors. This iterative process is performed until the final annotated version of the genomic sequence is obtained. The flowchart in Fig. 4 shows how the sequencing, annotation, analysis, and GenBank deposition procedures are interlinked. GenALA tools can import annotations from foreign sources including GenColors into GAP4 as tags on the assembled sequences, export annotation, and sequence information from a GAP4 project into a GenBank file ready for use with GenColors. It also import sequences and annotations from a GenBank file into GAP4, which then can be used as a backbone for the assembly of related sequences. GAP4 tags are linked to GenColors entries via unique identifiers thus enabling the maintenance of annotations regardless of the condition of the underlying sequence. By this interplay between assembly and annotation, we avoid repetitive annotations from scratch in different states of the finishing procedure and are able to reuse all annotations from the very start of a sequencing project. Fragmented assemblies can undergo directed gap closure owing to information gained from the underlying backbone, if at hand, and/or by the annotation information collected from GenColors. A more detailed description of the GenALA toolkit is available from the corresponding website at genome.fli-leibniz.de/genala/.

Annotation with GenColors will typically include the following steps:

1. Generate a GenBank file from the sequence containing CDS tags for all predicted genes. The user can also include other features that are supported by this format. For data export from GAP4 into the GenBank format one can use the respective GenALA tool.
2. Get GenBank-formatted genome sequences from closely related species and upload these together with the user's sequence into the locally installed dedicated browser system.

3. Start the comparative analyses and store the results as precomputed data (UniProt searches, COG and InterPro scans, BBH analyses).
4. Unify the annotations from the already annotated genomes to a “union reference genome” using the BBH table representations for two-way genome comparisons. The gene names and/or descriptions can be directly transferred from one genome to another one by mouse-clicks.
5. Transfer annotations from the “union reference genome” to your genome the same way as in **step 4**. That way, the gene set of your phylogenetic group of interest is annotated by mouse-clicks only.
6. Extend the annotation to previously unannotated and unique genes. Use the annotation sheets which provide enough detailed information about each (predicted) gene and allow for entry, revision, or removal of the annotations. For retraceability, these changes are logged.
7. Check for errors. If the user has provided quality and coverage values, they can be used to estimate sequence reliability and to mark possible errors in the assembly or sequence. Perform a synteny analysis to detect potentially false-positive gene predictions. Information on missing genes in relation to the union reference genome is easily accessed using the “core gene set” tool.
8. Because all predicted genes receive unique database identifiers, which can be used, e.g., in your assembly tool, you can go through several annotation rounds following the progress of the draft genomic sequence without losing previous information.

3.2.9. Genome Plots

The visualization of genomes can substantially contribute to a better understanding of both the overall genome structure and of selected genome parts. An excellent visualization tool is the commercial software GenVision by DNASTAR that has been used, for example, for displaying genome features of the *Escherichia coli* K12 genome (29). When we started the GenColors development, no freeware tool of this type was available, however. We have, therefore, included an option for circular and linear genome plots in GenColors. Both data of one and the same genome and the characteristics of different genomes can be displayed in one plot. Currently, all GenBank features, such as CDS genes for the positive and negative strands, CDS, RNA, tRNA, rRNA, and miscellaneous RNA genes for both strands as well as repeat regions and the replication origin, for example, can be displayed. In addition, precomputed data on GC content, GC skew, keto, and purine excess are available.

GC skew is a measure of nonrandom base distribution in genomes. It is defined as

$$\text{GC skew} = (G - C)/(G + C) \quad (1)$$

and is calculated over a sliding window of a certain size. In our case, the window size is alternatively 0.1 or 1 kb. G and C are the number of occurrences of guanine and cytosine in the selected window. The GC skew is a derivative function of the base composition along the sequence. In contrast, purine and keto excess are integral functions. The purine excess is calculated as:

$$\text{purine excess}(i) = \text{sum over } 1 \text{ to } i[\text{delta}(A,S) + \text{delta}(G,S) - \text{delta}(T,S) - \text{delta}(C,S)], \quad (2)$$

where S is the base present at the individual sequence positions. The summation is performed over the range between 1 and i. Delta (X,Y) equals 1 for X = Y and 0, if X differs from Y. Interchanging A and T in the formula defines the keto excess. It has been suggested that the minima and maxima of the purine excess-curve correspond to the origin and terminus of replication in prokaryotic genomes (30).

The genome plot option offers a filtering mechanism that allows the display of genes of a certain COG functional category. Given the protein sequences of a genome are included in UniProt, information on cross-referenced databases is available. The Protein Data Bank example has already been mentioned previously. However, visualization is possible for all of the more than 60 databases cross-referenced in UniProt. One further example would be the visualization of genes for which high-quality automated and manual annotation of microbial proteomes in the HAMAP system (31) is available. Finally, genes included in gene lists prepared by the user according to specific criteria can also be visualized. There is a number of options for customizing the graphics output that cannot be described in full detail here. It should be only mentioned that it is possible to mark genome segments and to show relative and absolute genome lengths in multigenome plots. For linear plots the number of basepairs per dot can be selected together with the paper sizes (DIN A0 to DIN A4). Given the boxes representing individual genes are large enough the gene names are shown.

The viewer generates images in PNG, PDF, and PS formats. The bitmap PNG format can be directly used for websites and presentation software that is not able to cope with vector graphics. On the other hand, the vector graphics output can be used for the generation of bitmap images of any resolution (see Note 5).

Examples of circular and linear plots are shown in Figs. 5 and 6. An example of a circular genome plot generated with the GenColors system can also be found in the report on the *Blochmannia pennsylvanicus* genome (32). Finally, it should be noted that during GenColors development a few related genome visualization tools were published. They include, for example,

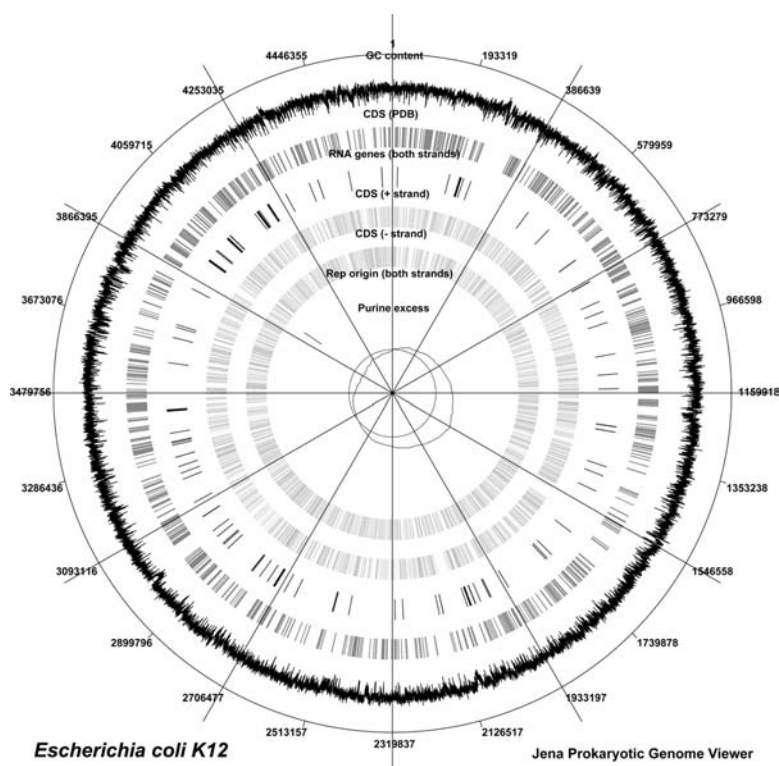


Fig. 5. Circular plot of features of the *Escherichia coli* K12 genome generated by Jena Prokaryotic Genome Viewer. The maxima and minima of the purine excess are located in the sequences ranges (maximum: 1.548.120–1.550.620, minimum: 3.929.072–3.931.572). The orbit descriptions are mostly self-explanatory. CDS [PDB] stands for genes for which three-dimensional protein structures are available in the Protein Data Bank. Note, that the origin of replication correlates with the purine excess minimum. In the original coloring scheme the CDS(+) and CDS(–) orbits are colored according to COG functional categories. All other orbits have a uniform color.

the Microbial Genome Viewer (www.cmbi.ru.nl/MGV/) (33), the GenDB system (www.cebitec.uni-bielefeld.de/groups/brf/software/genodb_info/) (34), and GenomeViz (www.uniklinikum-giessen.de/genome/) (35).

3.2.10. Access Modes and Availability

Most of the options of dedicated genome browsers and of JPGV are available in the free access mode. If the user wants to work with user-defined lists, online



Fig. 6. Linear genome plot for the sequence range 150,000 to 250,000 of the *Mesoplasma florum* L1 genome. Genes on the + and - strands are shown together with the GC content. The original GenColors coloring is according to COG functional categories. The font sizes have been modified after importing the PDF file into Adobe Illustrator.

registration is required. For user-defined lists, different access rights can be set ranging from default usage by the creator alone to free access. Further access rights, for example for annotation purposes, can only be obtained from the GenColors administrators. More detailed information on GenColors is available on the website gencolors.fli-leibniz.de. Currently, SGB (sgb.fli-leibniz.de) and JPGV (jpgv.fli-leibniz.de) are freely accessible. The GenColors system is also available upon request from the authors for local installation.

3.3. Summary and Outlook

GenColors provides a seamless integration of new sequences generated in ongoing genome projects with sequences of finished genomes obtained from GenBank and offers, in particular, a number of genome comparison tools. This represents a very effective mode of making directly available the richness of database information to the process of genome annotation and to genome analysis. GenColors is designed to allow an easy setup of dedicated genome browsers for a group of related genomes and also includes tools for the generation of linear and circular genome plots.

During the GenColors development a number of related tools have become available. Examples are the microbial annotation system MaGe (www.genoscope.cns.fr/agc/mage) (36), MicrobesOnline (www.microbesonline.org) (37), BugView (www.gla.ac.uk/~dpl1n/BugView/) (38), and the integrated microbial system IMG (img.jgi.doe.gov) (39). Also, some of the GenColors features bear resemblance to the Artemis/ACT system (40). Note, however, that contrary to GenColors no database is included in Artemis. So, we consider Artemis a useful supplementary tool to GenColors. Further databases and software for the comparison of prokaryotic genomes are compiled in a recent review (41). A comparison of these tools to GenColors is beyond the scope of this article.

The GenColors system is under continuous development. Ongoing work is primarily aimed at making available genome comparison options in JPGV that are already operating in dedicated browsers, at the prediction of genomic islands of horizontally transferred genes (42) and at a detailed analysis of intergenic sequence regions. Upon finalization of the manuscript clickable whole-genome views and results of horizontal gene transfer predictions according to an analysis based on codon usage have been included (43).

In summary, GenColors offers a great variety of tools for exploration and analysis of prokaryotic genomes and can thus hopefully contribute to one of the basic goals of current bioinformatics, the conversion of information into knowledge.

4. Notes

1. The available coloring schemes in the alignment viewer are: C-beta branched, aliphatic, aromatic, charged, equal, hydrophobic, negatively charged, no color, polar, positively charged, small, stacking, unequal. Note, that the percent identity values for aligned protein sequences calculated by BLAST and needle are usually different because BLAST performs a local alignment but needle a global one.
2. The following quantities are calculated by the program Syn-SCAN: S_d (observed synonymous [syn] substitutions), p_s (proportion of observed syn substitutions [S_d/S]), N_d (observed nonsynonymous [nonsyn] substitutions), p_n (proportion of nonsyn substitutions [N_d/S]), S (potential syn substitutions), d_s (Jukes-Cantor correction for multiple hits of p_s), N (potential nonsyn substitutions), d_n (Jukes-Cantor correction for multiple hits of p_n), d_s/d_n (ratio of syn to nonsyn substitutions).
3. When analyzing genomic elements, the number of core genes is identical in all of the elements included in the study. However, in whole genomes consisting of more than one genomic element these numbers may be different because one and the same gene may have BBHs in more than one genomic element.

4. Syntenic gene groups and synteny groups are defined according to the following approach: number the genes of the both genomic elements to be compared sequentially according to their sequence start position. Assign coordinates $(m,0)$ and $(0,n)$ to non-BBH genes and (m,n) to BBH gene pairs, where m and n are the gene numbers in the two genomic elements. Generate a two-dimensional matrix or a plot with these data and search for clusters for which all BBHs are separated by five or less genes from the next BBH. For a specific cluster the genes of each genomic element form a syntenic gene group and the two gene groups together represent a synteny group.
5. Graphics files in PDF format can easily be modified (fonts, colors, annotations, ...) with software of the Adobe Creative Suite such as Adobe Illustrator or Adobe Photoshop.

Acknowledgments

The help of Kerstin Wagner in setting up and maintaining the SGB external link page as well as in icon design is gratefully acknowledged. We are also grateful to Andreas Petzold who has contributed code to GenColors. This work was supported by the grants 0312704E and 0313652D of the German Ministry for Education and Research.

References

- 1 Fleischmann, R. D., Adams, M. D., White, O., et al. (1995) Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science* **269**, 496–512.
- 2 Fraser, C. M., Gocayne, J. D., White, O., et al. (1995) The minimal gene complement of *Mycoplasma genitalium*. *Science* **270**, 397–403.
- 3 Bernal, A., Ear, U., and Kyrpides, N. (2001) Genomes OnLine Database (GOLD): a monitor of genome projects world-wide. *Nucleic Acids Res.* **29**, 126–127.
- 4 Thomson, N., Sebahia, M., Cerdano-Tarraga, A., Bentley, S., Crossman, L., and Parkhill, J. (2003) The value of comparison. *Nat. Rev. Microbiol.* **1**, 11–12.
- 5 Bentley, S. D. and Parkhill, J. (2004) Comparative genomic structure of prokaryotes. *Annu. Rev. Genet.* **38**, 771–792.
- 6 Fouts, D. E., Mongodin, E. F., Mandrell, R. E., et al. (2005) Major structural differences and novel potential virulence mechanisms from the genomes of multiple campylobacter species. *PLoS Biol.* **3**, e15.
- 7 Romualdi, A., Siddiqui, R., Glöckner, G., Lehmann, R., and Sühnel, J. (2005) GenColors: accelerated comparative analysis and annotation of prokaryotic genomes at various stages of completeness. *Bioinformatics* **21**, 3669–3671.
- 8 Stajich, J. E., Block, D., Boulez, K., et al. (2002) The Bioperl Toolkit: Perl modules for the life sciences. *Genome Res.* **12**, 1611–1618.
- 9 Rice, P., Longden, I., and Bleasby, A. (2000) EMBOSS: The European Molecular Biology Open Software Suite. *Trends Genet.* **16**, 276–277.

- 10 Wu, C. H., Apweiler, R., Bairoch, A., et al. (2006) The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Res.* **34**, D187–D191.
- 11 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 12 Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997) A genomic perspective on protein families. *Science* **278**, 631–637.
- 13 Glöckner, G., Lehmann, R., Romualdi, A., et al. (2004) Comparative analysis of the *Borrelia garinii* genome. *Nucleic Acids Res.* **32**, 6038–6046.
- 14 Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.
- 15 Gonzales, M. J., Dugan, J. M., and Shafer, R. W. (2002) Synonymous-non-synonymous mutation rates between sequences containing ambiguous nucleotides (Syn-SCAN). *Bioinformatics* **18**, 886–887.
- 16 Nei, M. and Gojobori, T. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.* **3**, 418–426.
- 17 Sharp, P. M. and Matassi, G. (1994) Codon usage and genome evolution. *Curr. Opin. Genet. Dev.* **4**, 851–860.
- 18 Supek, F. and Vlahovicek, K. (2005) Comparison of codon usage measures and their applicability in prediction of microbial gene expressivity. *BMC Bioinformatics* **6**, 182.
- 19 Passarge, E., Horsthemke, B., and Farber, R. A. (1999) Incorrect use of the term synteny. *Nat. Genet.* **23**, 387.
- 20 Clark, M. S. (1999) Comparative genomics: the key to understanding the Human Genome Project. *Bioessays* **21**, 121–130.
- 21 Birney, E., Andrews, D., Caccamo, M., et al. (2006) Ensembl 2006. *Nucleic Acids Res.* **34**, D556–D561.
- 22 Nascimento, A. L., Ko, A. I., Martins, E. A., et al. (2004) Comparative genomics of two *Leptospira interrogans* serovars reveals novel insights into physiology and pathogenesis. *J. Bacteriol.* **186**, 2164–2172.
- 23 Ewing, B. and Green, P. (1998) Base-calling of automated sequencer traces using Phred. II. Error probabilities. *Genome Res.* **8**, 186–194.
- 24 Mulder, N. J., Apweiler, R., Attwood, T. K., et al. (2005) InterPro, progress and status in 2005. *Nucleic Acids Res.* **33**, D201–D205.
- 25 Harris, M. A., Clark, J., Ireland, A., and Gene Ontology Consortium. (2004) The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res.* **32**, D258–D261.
- 26 Berman, H. M., Westbrook, J., Feng, Z., et al. (2000) The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242.

- 27 Hulo, N., Sigrist, C. J., Le Saux, V., et al. (2004) Recent improvements to the PROSITE database. *Nucleic Acids Res.* **32**, D134–D137.
- 28 Bonfield, J. K., Smith, K., and Staden, R. (1995) A new DNA sequence assembly program. *Nucleic Acids Res.* **23**, 4992–4999.
- 29 Blattner, F. R., Plunkett, G. 3rd, Bloch, C. A., et al. (1997) The complete genome sequence of *Escherichia coli* K-12. *Science* **277**, 1453–1474.
- 30 Freemann, J. M., Plasterer, T. N., Smith, T. F., and Mohr, S. C. (1998) Patterns of genome organization in bacteria. *Science* **279**, 1827a.
- 31 Gattiker, A., Michoud, K., Rivoire, C., et al. (2003) Automated annotation of microbial proteomes in SWISS-PROT. *Comput. Biol. Chem.* **27**, 49–58.
- 32 Degan, P. H., Lazarus, A. B., and Wernegreen, J. J. (2005) Genome sequence of *Blochmannia pennsylvanicus* indicates parallel evolutionary trends among bacterial mutualists of insects. *Genome Res.* **15**, 1023–1033.
- 33 Kerkhoven, R., van Enckevort, F. H., Boekhorst, J., Molenaar, D., and Siezen, R. J. (2004) Visualization for genomics: the Microbial Genome Viewer. *Bioinformatics* **20**, 1812–1814.
- 34 Meyer, F., Goesmann, A., McHardy, A. C., et al. (2003) GenDB: an open source genome annotation system for prokaryote genomes. *Nucleic Acids Res.* **31**, 2187–2195.
- 35 Ghai, R., Hain, T., and Chakraborty, T. (2004) GenomeViz: visualizing microbial genomes. *BMC Bioinformatics* **5**, 198.
- 36 Vallenet, D., Labarre, L., Rouy, Z., et al. (2006) MaGe: a microbial genome annotation system supported by synteny results. *Nucleic Acids Res.* **34**, 53–65.
- 37 Alm, E. J., Huang, K. H., Price, M. N., et al. (2005) The MicrobesOnline Web site for comparative genomics. *Genome Res.* **15**, 1015–1022.
- 38 Leader, D. P. (2004) BugView: a browser for comparing genomes. *Bioinformatics* **20**, 129–130.
- 39 Markowitz, V. M., Korzeniewski, F., Palaniappan, K., et al. (2006) The integrated microbial genomes (IMG) system. *Nucleic Acids Res.* **34**, D344–D348.
- 40 Berriman, M. and Rutherford, K. (2003) Viewing and annotating sequence data with Artemis. *Brief. Bioinformatics* **4**, 124–132.
- 41 Field, D., Feil, E. J., and Wilson, G. A. (2005) Databases and software for the comparison of prokaryotic genomes. *Microbiology* **51**, 2125–2132.
- 42 Gogarten, J. P. and Townsend, J. P. (2005) Horizontal gene transfer, genome innovation and evolution. *Nat. Rev. Microbiol.* **3**, 679–687.
- 43 Waack, S., Keller, O., Asper, R., et al. (2006) Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models. *BMC Bioinformatics* **7**, 142.

Comparative Microbial Genome Visualization Using GenomeViz

Rohit Ghai and Trinad Chakraborty

Summary

Recent years have brought a tremendous increase in the amount of sequence data from various bacterial genome sequencing projects, an increase that is projected to accelerate over the next years. Comparative genomics of microbial strains has provided us with unprecedented information to describe a bacterial species and examine for microbial diversity. This has allowed us to define core genomes based on genes commonly present in all strains of a species or genus and to identify dispensable regions in the genome harboring genus-, species-, and even strain-specific genes. Nevertheless, the task of organizing and summarizing the data to extract the most informative features remains a challenging yet critical endeavor. Visualization is an effective way of structuring and presenting such information effectively, in a concise and eloquent fashion. The large-scale views unveil commonalities and differences between the genomes that may shed light on their evolutionary relationships and define characteristics that are typical of pathogenicity or other ecological adaptations. We describe GenomeViz, a tool for comparative visualization of bacterial genomes that allows the user to actively create, modify and query a genome plot in a visually compact, user-friendly, and interactive manner.

Key Words: Genome visualization; circular genome plots; comparative genomics; horizontal gene transfer; whole genome alignments.

1. Introduction

Several circular genome visualization tools have been developed, and offer a wide variety of features. The Microbial Genome Viewer ([1](#)) is one such online tool. Users can choose from several genomes and create plots within the web

browser. It also offers a data upload facility to plot experimental data. However, the plot customization is tedious, and if a mistake is made it is not possible to undo and repeat without destroying the entire plot. Search functionality is limited and the plot is not interactive enough. Genomap (2) provides the functionality to create circular maps and offers a large number of customizable features, but little help in creating plots quickly and easily. Also, the plot interactivity is limited. BugView (3) also allows some comparative analysis, but is limited to only two genomes. Though the abilities in linear comparison are useful, the circular plots are static. GenomePlot (4) provides a user-friendly tab-delimited file format for easy modification by users, but the plot must be customized for each genome, and once again, no interaction is possible with the resulting plot. CGView (5) offers much functionality, which makes it easy to create and customize the plots and provides excellent hyperlinked circular plots. But the search ability is limited, and no markup is possible on the plot after it has been created.

GenomeViz (6) offers several advantages to the user. It uses a simple tab-delimited file format that can be readily modified by the user. It provides users with several premade files ready for beginning plotting immediately. Features like “tagging” provide the user with complete control over the colors of each gene. It also offers several different plotting methods for numerical data. Moreover, the plot is interactive (albeit with limited zooming ability), and it is easy to locate genes in one or all the genomes plotted, and extract data from either selected regions or parts of the plot. Creating the plot is itself an interactive process providing the user with complete control over the plot appearance. The resulting figures (Fig. 1) are publication quality. Some scripts are also provided to make the common tasks simpler for the user (*see* Notes 1, 2, 3)

There are two types of information that a visualization program must be capable of displaying; qualitative and quantitative. It is important to be able to visualize both qualitative and quantitative data from microbial genomes. Functional classifications (like Clusters of Orthologous Groups [COGs]) and identification of horizontally transferred genes are examples of qualitative data. They allow us to classify genes into different groups. Thus, it is informative to compare, for example, the distribution of potentially horizontally transferred genes between two related microbial genomes. Such a comparison can provide us with clues to regions that are more prone to insertion and deletion events in the coevolution of these two genomes. Quantitative data is simple numerical data, e.g., gene length, GC skew, GC content, conservation scores, gene expression intensity values, and so on. Quantitative data may be of two

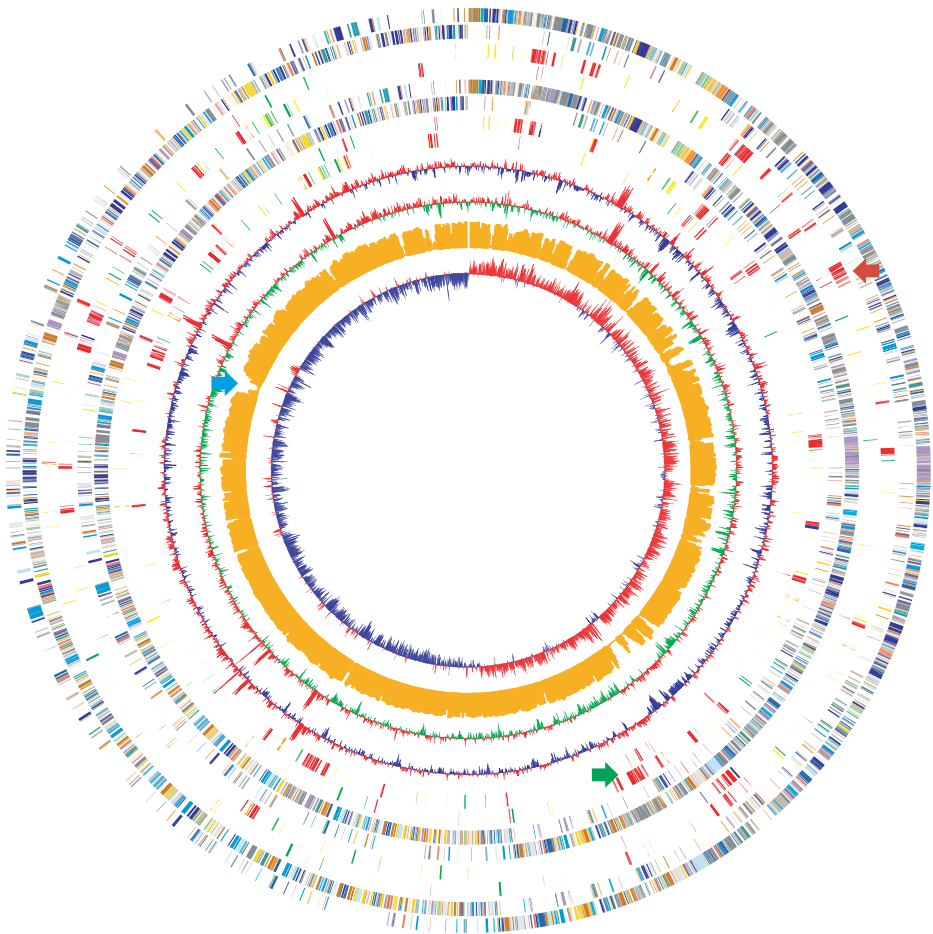


Fig. 1. The figure shows a typical GenomeViz plot. Shown in the figure is a comparison between the genomes of *Listeria monocytogenes* EGDe (pathogenic) and *Listeria innocua* (nonpathogenic). The outermost two circles are both strands of *L. monocytogenes* colored according to COG categories. The next two circles show the distribution of potential horizontally transferred genes in the *L. monocytogenes* genome as identified by the SIGI software (7). Shown next are both strands of *L. innocua* (again colored according to COG), followed by the horizontally transferred genes in this genome. The next two circles show GC-content plots for *L. monocytogenes* and *L. innocua*, respectively, followed by a whole genome alignment of both genomes computed using AVID. Last, the innermost circle shows the GC-skew plot of the *L. monocytogenes* genome. It is easy to identify visually the differences in the horizontally transferred genes in the two genomes, and correlate it with the GC-plots or the

types, gene-based (gene length, expression values) or window-based (GC skew, conservation scores). Gene-based quantitative data refers to a data where each gene is associated with a single value, e.g., gene length or fold change at one time point in a microarray experiment. Window-based quantitative data refers to values calculated for short, overlapping segments of the genome. GC content and GC skew for a genome are usually calculated in this manner.

2. GenomeViz Tags

GenomeViz uses the concept of “tags,” which may be applied to groups of genes for classification-type qualitative data. A tag is just a name given to a group of genes. It may be a short word, or a letter of the alphabet (e.g., “U” for genes with unknown function, or “CON” for genes conserved across a comparison of a few genomes). The genes of a genome may be divided into different groups and each group given its own “tag.” Tagging provides the user with the ability to change colors for entire groups easily and gain more control over the GenomeViz display (*see Note 5*) All the information on the groups and tags to be used in a particular plot must be written in a tag file. A tag file is a tab-delimited text file of at least two and at most three columns. It has the “tags” in the first column, their colors in the second, and their brief descriptions in the third column.

A small two-column tag file is shown next.

Transcription	RED
Translation	GREEN
OtherGenes	GREY

The first column is the tag column. In this example, it means that we have three types of groups (and so three tags) for the genes, “Transcription,”



Fig. 1. whole genome alignment. The red arrow indicates a group of genes identified as horizontally transferred in the *L. monocytogenes* genome but not in *L. innocua* and the green arrow shows genes identified in *L. innocua* but not in *L. monocytogenes*. Frequently, such regions are accompanied by deviations in GC content or gaps in the genome-alignment. Alignment gaps that may be indicative of regions of insertion/deletion in both genomes also can be easily seen, one such gap is marked with a blue arrow.

“Translation,” and “OtherGenes.” The second column simply states the color that should be used for coloring each group.

To change the color of the genes involved in “Translation,” simply change the text GREEN in the second column to say, BLUE. When the plot is reloaded, the new colors will be displayed.

However, a tag file may also have three columns, as shown next.

```
T orange transcription
R blue translation
M green cell motility
S violet signal transduction
- grey function unknown
```

The third column can be used to describe the tag if we wish. Its purpose is to provide a more informative description. It is recommended that numbers (0, 1, 2, 3...) not be used as tags. The character “-” can also be used as a tag. All these columns must be separated with a “single” tab character only.

When one has a large number of tags, then it is useful to have a short description of the tag. The tag file can be displayed within GenomeViz to read the descriptions anytime. A tag file with all the COG categories is provided with GenomeViz.

3. GenomeViz Map File

The file that contains the actual data to be plotted is called the map file. This has been designed to be a simple format that can be easily edited and modified by anyone manually or with a program.

A sample map file is shown next (first few lines from the genome of the hyperthermophilic archaeon *Aeropyrum pernix* genome).

```
1669695
APE0001 - - 213 938 hypothetical protein
APE0002 K - 938 1276 hypothetical protein
APE0004 R - 1260 2174 hypothetical protein
APE0006 - + 2261 2836 hypothetical protein
APE0007 - + 3896 5440 hypothetical protein
APE0009 P - 5774 6091 transport protein
```

The first line of the map file contains only a single column, and a single value: the total number of bases in the genome, in this case, 1,669,695. All other lines of the map file contain six tab-delimited columns. The six columns are described next.

1. A gene identifier or a name. National Center for Biotechnology Information (NCBI) frequently uses a “Locus Tag” feature to describe bacterial gene identifiers. For example, APE0001 is the locus tag for the first gene in the *A. permix* genome. The locus tag for each gene can be seen in the NCBI Gene database. There are some limitations to this identifier. First, it must be only a single word. Second, it must not be entirely a number, e.g., 1, 10, 124, are all invalid gene identifiers. Third, it must be unique for the genome the user is trying to plot. All identifiers for the genomes provided with GenomeViz follow these three basic rules.
2. The tag/value column. The second column contains the tag that has been applied to this gene to make it a part of a group of genes. In the example previously listed, four types of tags are visible, “K”, “R”, “P”, and “—”. The colors for these tags (and for others in the map file) must be described in the tag file. The second column contains tags in this example because this is an example of a qualitative data file. A map file, which contained the gene lengths for example, would have, in place of the tags, integer values for each gene.
3. The strand column. This column simply denotes the strand on which the gene lies. There can be only two values for this column, “+” or “—”. No other values are acceptable.
4. Gene start. This column contains the location of the start of the gene feature.
5. Gene end. This column contains the location of the end of the gene feature. Both the gene start and gene end must be valid integer values.
6. Description. The last column of the map file. It contains the description, annotation, name of the gene, and any other text information.

The only difference between a qualitative data map file and a quantitative data map file is the values in the second column. All other columns are identical for the same genome.

If there is any line in the map file that does not have six columns in the correct format, GenomeViz will show an error, point out the incorrect line number and the column, and stop the plotting. In such as case, one must identify the error, correct it, and redo the plot again. The map file format is easy to maintain and modify in simple text editors or spreadsheets, and the extensive format checking performed by GenomeViz before plotting helps identify and correct mistakes before they are incorporated in the plot. The map file alone is sufficient for plotting numerical data, but both the map and tag files are needed to plot classification-type data. The type of data, qualitative or quantitative, is automatically detected from the map file.

4. Plotting a Genome Circle

4.1. Types of Plots Available in GenomeViz

It is possible to plot data in several ways with GenomeViz. Given next is a list of methods available for plotting.

1. Plotting classification style data (qualitative).
 - a. Two circles (+ and – strand separately).
 - b. Single circle (both + and – strands as a single circle).
2. Plotting numerical data (quantitative).
 - a. Gradient style graph with two circles (+ and – strand separately).
 - b. Gradient style graph with single circle (+ and – strands as a single circle).
 - c. One-sided line graph (like a circular bar chart, useful for alignment data).
 - d. Two-sided line graph (useful for GC content and GC skew).

4.2. Plotting Classification-Style Data

Both the tag and map files are needed to create a classification-style plot in GenomeViz. Follow the following steps to create a classification style data plot in GenomeViz.

4.2.1. Loading a TAG File

1. Go to File in the Main menu.
2. Select Load Tag File, and choose for which genome to be loaded a TAG file for (Genome 1, 2, 3, ... 8). Choose “Genome 1.”
3. Browse to the location of a tag file (say the TAG file supplied with GenomeViz – tagfiles/COGs.tag).
4. Click Open. The tag file COGs.tag is now loaded and this is displayed in a small frame below the main menu. The loaded tags are also shown in the text display area. Now follow the steps next to load a map file and create the plot.

4.2.2. Loading a MAP file

1. Go to File in the Main menu.
2. Select “Load Map File 1.”
3. Choose “Draw Two Circles.”
4. Choose “Classification Style Graph.”
5. Browse to the location of a map file (e.g., the map file supplied with Genome Viz for *Escherichia coli* K12 in the samples/classification-data directory – *Escherichia coli*_K12.map).

6. Click Open.
7. The genome of *E. coli* K12 will be displayed (two circles for two strands) colored in the COG colors (as specified in the tag file) as Genome 1.

4.3. Plotting Numerical Data

No tag files are needed for plotting numerical data. Only a map file containing quantitative data is sufficient. Follow the following steps to load a map file containing numerical data to create a plot.

1. Go to File in the Main menu.
2. Select “Load Map File 1.”
3. Choose “Draw Two Circles.”
4. Choose “One Sided Line Graph.”
5. Choose “Blue.”
6. Browse to the location of a map file (e.g., the gc-content map file supplied with GenomeViz for *E. coli* K12 in the gc-content-mapfiles directory).

The GC content of the *E. coli* K12 genome in the map file will be displayed as a one-sided line plot colored in blue.

5. Plot Navigation and Highlighting

5.1. Using Mouse Over

In all plots, Mouse Over on any gene immediately displays all the information about the gene in the display areas just below the Main menu. The line number in the map file, the gene identifier, the tag/value, strand, gene start, gene end, and description all are displayed.

5.2. Selecting Genes

Clicking on any gene in the plot highlights it in a color called the “Selection Color.” The default Selection Color is yellow. The information on a selected gene is also displayed in a text display area on the right side of the drawing area. Right clicking on a gene deselects it.

5.3. Select COGs

One can select COG categories directly for each genome using this menu provided they are available in the map file. Thus, Select COGs→Select COGs in Genome 3→K-transcription, selects all genes classified in the category Transcription in the Map file for Genome 3. It is possible to select different categories in the same genome in different colors by simply changing the selection color before selecting the category. However, it is advisable to use

a neutral background tag file, e.g., COGsGrayScale.tag, to provide a better contrast for the categories of the user's choice. This tag file colors all COG categories in a neutral gray color. The user may also edit this tag file to reflect any other color as well.

5.4. Searching for Genes of Interest

The complete information in the map file can be searched using the Search option. All genomes may be queried independently of one another. Go to Search → Search Genome 1 (to search in the first genome). A Search window appears. Type in the term to search, and press “search” (*see Note 6*). After the search is completed, a pop-up window appears and lists how many results were found. These results can be examined in the text display area on the right hand side of the drawing area. The search results may also be saved to a text file. In addition, all the genes that matched the search pattern are highlighted in the GenomeViz plot in the “Search Color.” Several different searches (each with a different search color) can be run on the same genome or the plot. In this manner, the search and highlight functionality provides one with a rapid overview of distribution of search terms over the genome. A global search function is also available, i.e., all the plotted genomes may be searched at once for a single pattern. The results are displayed genome-wise in the text display area.

5.5. Removing a Genome Circle

If there has been an error in plotting a genome circle, this particular circle can be easily removed without affecting the rest of the plot. Navigate to Clear → Genome 1, to remove the outermost circle. Choose File → Clear All, to reset the entire plot.

5.6. Plot Summary

To have a quick overview of which files have been used to create each genome circle, one can go to Summary → Plot Details to have look at a table containing the names of all the tag files and map files being used for each genome circle in the plot.

5.7. Printing the Plot

It is possible to create publication quality plots with GenomeViz (*see Note 4*). Once the user is satisfied with the plot created and wants to finally print it, the user can go to File → Print. A print dialog box appears with several options. Give the dialog box time to complete its rendering of the print preview plot in the

small window. Choose the paper size and choose “Print to file” option. Provide a name for an output file, e.g., myplot.ps. GenomeViz creates postscript output files that can be easily read in by standard graphics programs, and converted to a PDF if desired.

5.8. The Mask Genome Menu

The search function provides highlighting genes based on a pattern match, and the tag file allows genes to be colored based on the group in which it belongs. To color genes on a numerical data plot, that do not share any common search pattern, it is not possible to color them using these options. However, individual genes of interest in both the classification-style plots and the quantitative data plots are searchable and can be colored by using the special mask genome menu. It is somewhat like a multiple search option, but with the facility of coloring each result in a specific color. It has a simple format, a two column tab-delimited format, as shown next. The first column is the gene of interest, and the second column specifies the color it should be displayed in.

```
Gene1  red
Gene2  blue
Gene3  yellow
```

The Gene1 will be red, Gene2 will be blue, and Gene3 will be yellow. No format checking is performed on the mask file. It must be ensured by the user that the format is correct, all gene identifiers used are present in the map file, and that the colors are valid Tk colors.

6. Implementation

6.1. Supported Platforms

GenomeViz has been tested to run successfully on Linux and Solaris operating systems (*see Note 7*). Unix systems that have ActiveTcl installed may also run GenomeViz but we have not tested this.

6.2. ActiveTcl

It is required that the user install ActiveTcl distributed by ActiveState (<http://tcl.activestate.com>) to run GenomeViz. It is recommended over any other existing Tcl installation that the user might have to run GenomeViz. Installing ActiveTcl will not interfere with the user existing Tcl installation and will have no effects on the user's Tcl programs, if the user has any.

6.3. Perl

The user will also need Perl to run the scripts that are distributed with GenomeViz (see **Note 8**). Perl is usually installed by default on Linux/Unix systems in the path `/usr/bin/perl`. The user can easily check this by typing the following command on the terminal.

```
which perl
```

The user may get

```
/usr/bin/perl
```

which means the user already has Perl installed, or the user may get something like

```
perl not found
```

which means the user does not have Perl and will need to install it.

If the user does need to install Perl, once again it is recommended that the user gets the ActivePerl distribution from ActiveState. It is easy to install and should not pose any difficulty.

7. Notes

1. Use the Perl programs `gc2viz` and `gcskew2viz` to compute window-based mapfiles for plotting in GenomeViz. They use only the nucleotide fasta file as input and create a mapfile that can be plotted in GenomeViz. The GC content map files supplied with GenomeViz contain only the GC content values of the actual genes themselves. The user can download whole genome nucleotide files for any sequenced bacteria genome from NCBI (<http://www.ncbi.nlm.nih.gov/genomes/lproks.cgi>). The nucleotide sequence files on the NCBI server have the ".fna" file extension.
2. A common application involves a list of genes that one would like to plot and visualize along with other data. The script "*tagit*" makes it simple for to create a file that can be plotted and visualized easily with GenomeViz. Suppose the user has a list of genes that the user is interested in. The user should provide the script with the file containing this gene list and the tag the user wants to attach to these genes. The user should also provide the map file to be used (currently GenomeViz provides 120 map files to choose from). The script creates a new map file, but with all the genes tagged with the designated custom tag the user provides.
3. Whole genome alignments provide us information on which regions of the genomes have been conserved and which have been subject to deletion and insertions. It is easy to get complete genome alignments of bacterial genomes using AVID. Avid also provides a simple format for the representation of such alignments. The script *avid2viz* can reformat genome alignment data from the AVID program to a map file format that can be plotted in GenomeViz. This map file can be used to visualize conservation data of genomes along with other data such as GC content, Basic Local Alignment Search Tool scores, and so on in GenomeViz.

4. Once a plot has been made, it should be saved to a postscript file. However, when the plot needs to be recreated, one needs to use the same input files once again. Use the Summary→Plot details to save the details of the files used to create the user's plot in such a case.
5. There are many different ways to specify the colors in the tag file. The colors in a tag file may be written by their name, e.g., Red, red, or RED are all acceptable. Hexadecimal codes are also allowed. Two color browsers are provided within GenomeViz that can help to select colors and obtain their standard names or hexadecimal codes.
6. The search box supports advanced pattern matching abilities provided by the Tcl/Tk regexp. For example, if the user wants to search for genes containing the pattern tRNA or rRNA, the user can type tRNA|rRNA, where the "|" character denotes OR. A link to a complete guide for regular expression pattern matching using Tcl can be found at the GenomeViz homepage.
7. GenomeViz and accompanying scripts and data can be download at the GenomeViz homepage (<http://www.uniklinikum-giessen.de/genome/>).
8. If the user can program in Perl, it is easy to modify the scripts provided with GenomeViz to create new programs that can compute parameters using a window-based approach, e.g., dinucleotide content, complexity, and so on.

Acknowledgments

The work reported herein is supported by grants from the Deutsche Forschungsgemeinschaft and the BMBF Network Program Pathogenomics to TC. RG is supported by the Graduate College of Biochemistry of Nucleoprotein Complexes (GK370), Justus Liebig University, Giessen, Germany.

References

- 1 Kerkhoven, R., van Enckevort, F. H., Boekhorst, J., Molenaar, D., and Siezen, R. J. (2004) Visualization for genomics: the Microbial Genome Viewer. *Bioinformatics* **20**, 1812–1814.
- 2 Sato, N. and Ehira, S. (2003) GenoMap, a circular genome data viewer. *Bioinformatics* **19**, 1583–1584.
- 3 Leader, D. P. (2004) BugView: a browser for comparing genomes. *Bioinformatics* **20**, 129–130.
- 4 Gibson, R. and Smith, D. R. (2003) Genome visualization made fast and simple. *Bioinformatics* **19**, 1449–1450.
- 5 Stothard, P. and Wishart, D. S. (2005) Circular genome visualization and exploration using CGView. *Bioinformatics* **21**, 537–539.
- 6 Ghai, R., Hain, T., and Chakraborty, T. (2004) GenomeViz: visualizing microbial genomes. *BMC Bioinformatics* **5**, 198.
- 7 Merkl, R. (2004) SIGI: score-based identification of genomic islands. *BMC Bioinformatics* **5**, 22.

BugView

A Tool for Genome Visualization and Comparison

David P. Leader

Summary

We describe BugView, a cross-platform application for presenting and comparing the genomes of bacteria or eukaryotes. We give particular emphasis to its use in comparing the genes of related bacterial genomes, and consider different methods of automating the preparation of genome comparison files, including a new web-based facility. Ways of using BugView to study and present the internal structure of genomes are also discussed. BugView/web, a Java applet for web deployment of BugView files, is presented for the first time.

Key Words: Genome; genome comparison; genome visualization; synteny; dynamic programming; Java applet.

1. Introduction

BugView is a desktop computer program, designed to allow users to visualize and compare pairs of bacterial genomes (*1*). It uses Genbank files, publicly available from the National Center for Biotechnology Information (NCBI) FTP site, as a source of genome data; and it incorporates comparison functions employing dynamic programming. The program is free and cross-platform: versions are available for Mac OS 8/9, Mac OS X, Windows 95 to Vista, and Unix/Linux.

BugView is not restricted to displaying bacterial genomes: it can handle introns, and so can also be used with eukaryotic genomes. Nor is there anything to prevent it being used to display and edit single genomes, either individually

or together in the same window. However, this chapter concentrates, for the most part, on describing how to use BugView with pairs of related bacterial genomes—its primary purpose. It describes how to download the required data files, how to create a special comparison file that stores the relationships between the genes of the two genomes, how to navigate and edit the displayed comparison, and how to export particular views of a genome comparison.

This is followed by a section that considers the visualization and presentation of genes within the genome, in particular the arrangement of genes belonging to similar functional categories.

A final section describes how to display one's own BugView comparisons on the web using a special version of the program (a Java applet).

2. Software and Data Files

2.1. BugView

At the time of writing the latest version of BugView is 1.3.4 (released October 2006), which supercedes all previous versions. In particular it allows parsing of .ptt files in the format introduced in 2006, and is recommended for all users.

2.1.1. Downloading and Installing BugView

1. Connect to <http://www.gla.ac.uk/~dpl1n/BugView/bvddownload.html> (or alternative, see **Note 1**).
2. The user should click on the link for the operating system of the user's computer, and the file will be downloaded.
3. The Manual should be downloaded, and download the Sample files on the same webpage.
4. The files are in different compressed formats for each platform, and, if they do not decompress automatically, may be compressed with the following utilities: Mac OS8/9 (.sit)—Stuffit, Mac OS X (.dmg)—double-click to open the disc image, Windows (.zip)—Winzip, Unix/Linux (.tar.Z)—uncompress, followed by tar xvf.
5. For Mac and Windows, just drag the uncompressed executable program to a location of the user's choice; for Unix/Linux the program is in the form of a file, BugView.jar, which should be placed in the same location (most conveniently one in the user's "path") as a supplied shell script bugview.sh. For Mac OS8/9 it is advisable to rebuild the desktop to ensure the application and files acquire the correct icons.
6. The Mac and Windows versions are launched by double-clicking the BugView icon; the Unix/Linux version is launched by running the shell script, bugview.sh.

2.1.2. Hardware Requirements

To run BugView a machine with a processor speed of at least 500 MHz—extremely modest by contemporary standards—is recommended, although the performance of the sequence-comparison functions within BugView is appreciably enhanced on machines with faster processors. The free RAM requirement is more difficult to quantify, but on older machines insufficient RAM can limit the size of genome file that can be loaded (*see Note 2*).

2.1.3. System Software Requirements

BugView is a Java program and requires an operating system-specific version of the “Java Virtual Machine” to run. The situation for different operating systems is summarized next.

1. Mac OS8/9. MRJ (Mac Runtime for Java) is part of the Mac OS 8 or OS 9 installation. The last standard version of this for classic Mac, MRJ 2.2.5, can be downloaded using the Software Update control panel.
2. Mac OS X. Apple’s version of the Java Virtual Machine is part of the Mac OS X installation. As of Mac OS X 10.4.6, the default version of Java is 1.5, although previous versions of the OS may have Java 1.3 or 1.4. Although Java 1.5 is not needed for the basic functionality of BugView, it is required to overcome one specific OS X “bug” (*see Note 3*).
3. Windows. Some versions of Windows shipped with Microsoft’s limited version of the Java Virtual Machine, which, nevertheless, should be adequate to run BugView. Later versions did not, in which case the latest version of Java for Windows can be downloaded from Sun Microsystems’ website (<http://java.sun.com/>).
4. Unix/Linux. A Java Virtual Machine is installed with Sun’s Solaris operating system, but may not come with other versions of Unix, in which case a version can be downloaded from Sun’s website (<http://java.sun.com/>).

2.2. Genbank Files

Bacterial genome files are available from NCBI. They are actually held on the NCBI FTP site at <ftp://ftp.ncbi.nih.gov/genomes/Bacteria/>. This gives an alphabetical listing of completed genomes with links to download pages for individual entries. Alternatively the files can be accessed via the website, currently from <http://www.ncbi.nlm.nih.gov/genomes/lproks.cgi>. Here, the alphabetical listing provides more information than that on the FTP site, but the route to the FTP download site is more complex: one should click on the “RefSeq” link for the bacterium of interest, and then on the resulting page click on “RefSeq FTP” (*see also Note 4*).

2.2.1. *gbk Files*

The file with the extension “.gbk” in the section of the FTP site for a particular bacterium contains the nucleic acid sequence of its genome, and annotation of genes and other features. This is the file that is required for viewing the genome in BugView. There may, in fact, be several files with different RefSeq numbers (identifiers starting with “NC_”), the largest being the bacterial genome and the other(s) being plasmids associated with it.

It is worth remarking that the RefSeq (Reference Sequence) number for a genome generally corresponds to the number on the “Accession” line of its documentation, and is often referred to as the Accession number. In this chapter, we shall use the term “RefSeq number” throughout for consistency.

2.2.2. *ptt Files*

Files with the extension “.ptt” contain tabular information on each annotated gene, with columns available for, but not always furnished with, the COG (Classification of Orthogonal Groups) number and category (2). Because the COG category can be imported into BugView it is worth downloading the relatively small .ptt files corresponding in RefSeq number to the .gbk files one has downloaded in **Subheading 2.2.1.**

2.2.3. *faa Files*

Files with the extension “.faa” contain the amino acid sequences (in FastA format) for all the annotated genes of a genome. If the Basic Local Alignment Search Tool (BLAST) is to be used to generate a genome comparison file (**Subheading 3.1.3.**), download the .faa files corresponding in RefSeq number to the .gbk files downloaded in **Subheading 2.2.1.**

2.3. *Optional Ancillary Software*

2.3.1. *Standalone BLAST*

Subheading 3.1.3.2. describes how to use a local installation of the program, BLAST (3), to generate a BugView genome comparison file. Standalone versions of BLAST for various platforms (but not for Mac OS8/9) can be downloaded from <http://www.ncbi.nlm.nih.gov/BLAST/download.shtml>. Instructions for installation are included in the download, but the nontechnical user will probably require assistance with the set up.

2.3.2. *gcfprep*

To use standalone BLAST to generate a BugView genome comparison file it is necessary to perform successive comparisons of protein sequences. A Perl script to automate this, *gcfprep*, can be downloaded from <http://www.gla.ac.uk/~dpl1n/BugView/bvdownload.html>. This script has only been tested on standalone BLAST running under Solaris, and may need modification to run on other platforms.

2.3.3. *BlastToGCF*

Subheading 3.1.3.3. describes how to use a Grid-enabled web-accessible version of the program, BLAST, to perform successive comparisons of the protein sequences encoded by the genes of two genomes. A small utility program, BlastToGCF, has been written to convert the output to a BugView genome comparison file. Versions for different platforms are available for download from <http://www.gla.ac.uk/~dpl1n/BugView/bvdownload.html>.

3. Using BugView

3.1. Genome Comparison

3.1.1. *An Introduction to the BugView Interface*

Figure 1 shows BugView after launching and loading files. Three regions of the interface can be distinguished: the menu bar, a control console consisting predominantly of named buttons, and the main genome display window, only the upper part of which can be seen in the figure. (This latter will be blank before any files have been loaded.) General operations can be accessed from menu items, and the Help menu (the position of which may differ from that in the illustration, depending upon the platform) gives access to systematic brief descriptions of the items in each of the menus. The controls in the console mainly operate on objects (genes, and so on) in the display window. If no object is selected—as will be the case initially—most of the controls will be dimmed, indicating that they are unavailable. The operation of these controls is described in **Subheadings 3.1.4.** and **3.1.5.** **Subheading 3.1.6.** considers some additional controls for “power users.”

3.1.2. *Processing Genbank Files*

1. The first time the user works on a genome in BugView the user must convert one or more .gbk file, downloaded as described in **Subheading 2.2.1.**, to BugView-format Data and Sequence files.

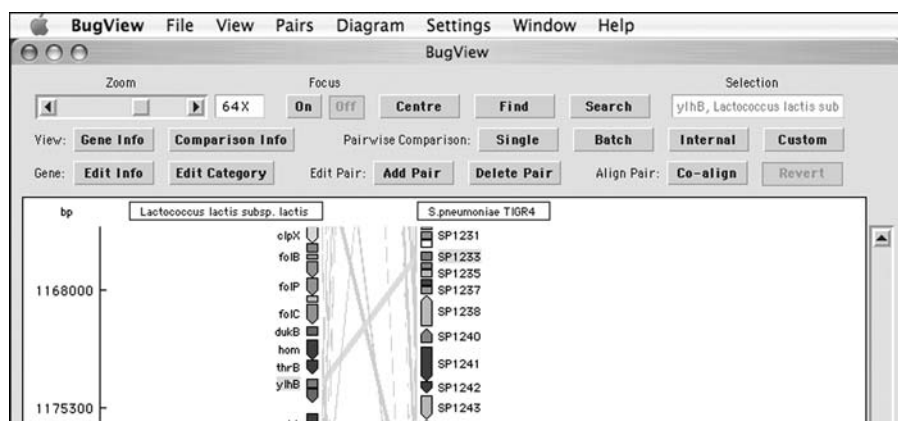


Fig. 1. General view of BugView. The console of control buttons can be seen with most in an active state. Above the console the menus are visible (here for Mac OS X—they will differ in detail for other platforms), and below a section of the display area, with part of the vertical scrollbar visible.

2. Choose “Convert Genbank File” from the File menu.
3. Wait while the conversion occurs, which may take 1–2 min (*see Note 5*).
4. When conversion is complete the user will receive a message containing the filenames of the Data and Sequence files generated. These are based on the RefSeq number of the genome, and are given the extensions “.gda” and “.seq,” respectively.
5. Now is a convenient time to import any COG information from the .ptt file downloaded as described in **Subheading 2.2.2**. Choose “Load COGs from .ptt File...” from the “Load COGs” submenu in the File menu. (This can be done subsequently, but if the menu choice is not available, *see Note 6*.)
6. A message will appear indicating how many COG categories have been assigned. If this is zero, it will reflect the absence of COG information in the .ptt file. If the .ptt file contained COG information, then those genes to which this information relates will have become colored according to a scheme to be found in the Help menu under “Category Colour Key.”
7. Choose “Save File” from the File menu to save the COG annotations.
8. In subsequent sessions with BugView the Data and Sequence files are used, and the Genbank file is not required. Edits the user makes (such as assigning COG categories) will be saved to the Data file. (This separation from the Sequence file—which cannot be edited in BugView—accelerates saving changes to the Data file, and avoids possible corruption of the Sequence file.)
9. Before performing a second conversion one is advised to unload loaded files. Choose “Unload All Files” from the File menu.

3.1.3. Creating a Comparison File

A BugView comparison file contains a user-defined list of gene pairs for two genomes. There are three ways in which it can be created.

3.1.3.1. CREATION WITHIN BUGVIEW

Because of the time involved in generating a large number of comparison pairs manually, the user would normally only create a comparison file from within BugView when the user just wished to compare a subset of genes within two genomes, or when automated creation (**Subheadings 3.1.3.2.** and **3.1.3.3.**) was not available.

1. Load the data (.gda) and sequence (.seq) files (**Subheading 3.1.2.**) for the two genomes to be compared by choosing “Open Genome File” from the File menu, followed by “Open Data File” or “Open Sequence File”, as appropriate, from the submenu.
2. Choose “New Comparison File” from the File menu.
3. The user will be prompted to name the file being generated, and the user will be able to navigate through the filespace to the directory where it will be saved. It is recommended that the file be saved to the same directory as the associated genome files, and that its name should reference the RefSeq numbers of these (*see Note 7*) and have the extension .gcf (e.g., NC_003112-NC_003116.gcf).
4. A message will appear informing the user that the comparison file has been created. Although the area between the genomes (where the comparison pairs will appear) is not yet occupied, the labels on the right-hand genome have been reoriented to the right (i.e., to the outside) so that they do not intrude upon this inner area.
5. Addition of pairs to the comparison file is described in **Subheading 3.1.4.2.** Whether or not one starts adding pairs immediately, it is worth generating a project file at this stage (*see Note 8*).

3.1.3.2. CREATION USING STANDALONE BLAST

It is assumed that the user has downloaded (**Subheading 2.3.1.**), installed, and tested standalone BLAST according to the NCBI documentation, and also downloaded gcfprep (**Subheading 2.3.2.**). It should be emphasized that the user is performing comparisons of the protein products of the annotated genes found in the .faa files (**Subheading 2.2.3.**), not comparisons of the nucleic acid sequences.

1. Run the formatdb program included in the BLAST download (*see also Note 9*) to generate databases of the genomes from the relevant .faa files. Check the “formatdb.log” file for possible problems at this stage.

2. Run the script `gcfprep` by typing its name and responding to the prompts. Comparison of two bacterial genomes might take up to 1 h, depending on the speed of the machine, but an indication of progress is given every 50 comparisons. The output `.gcf` file lists the gi numbers of all intergenome pairs with an *e*-value less than 0.05 (or as specified by the user if the alternative `gcfprepE` is used). A log file contains details of any proteins that were missed.
3. BugView has several features that allow filtering of comparison pairs on the basis of percentage identity, rather than *e*-value. To use these features, it is necessary for the percentage identities to be calculated within BugView. To do this, after the comparison file (and the cognate genome files) has been loaded into BugView (*see Note 10*), choose “Update Pair Scores” from the Pairs menu. After the update has been completed (*see Note 11*) remember to save.

3.1.3.3. CREATION USING GRIDBLAST

For those users who are not in a position to set up standalone BLAST, web access to a BLAST grid service has been provided by BRIDGES, a UK e-Science project. This had just come into operation at the time of writing, and it is possible that some details of use (particularly the url) may have changed by the time of publication. (Check the BugView website.) Before starting, ensure that the `.faa` file for at least one of the genomes to be compared is available.

1. Connect to <http://cassini.nesc.gla.ac.uk:9081/wps/portal> (*see Note 12*).
2. The user is required to register before being able to use this Grid service. There is a small link—“Sign up”—at the top right of the page for doing this.
3. After registration, click “Log in” at the extreme top right corner, which opens the login page. There, enter the User ID and Password in the appropriate fields, and click the “Log in” button.
4. On the page that appears, click the blue “Computational Resources” tab on the horizontal bar.
5. Next, click “GRIDBLAST Job Submission,” which loads the page for running BLAST genome comparisons.
6. In the first two fields, respectively, enter a job name and, if the user prefers not to wait while the job runs, an e-mail address for notification of completion.
7. Clear the contents of the third field and leave it empty. Instead of pasting the large genome `.faa` file here, upload it from the filespace at the “Select input file” option using the “Browse” button. Note the RefSeq number of this file, and the fact that it will subsequently be referred to as the “Query” genome.
8. Choose the second genome from the list on the pull-down menu. The names of the genomes, rather than their RefSeq numbers, are listed on the menu, check to reconcile these, referring to the Genbank website if necessary (*see Subheading 2.2.*). Make a note of this RefSeq number as that of the “Database” genome.

9. None of the default values of the pull-down menus is appropriate. Carefully select the following:

BLAST Program	blastp
<i>e</i> -value	0.1 or 0.01
word size	3
generate alignments	no
include gi numbers in output	yes
output format	txt
10. Click the button entitled “Submit Job.” It typically takes about 10 min for a comparison of genomes with 2000 genes to run, generating an output file of about 5 Mb in size.
11. The relevant information from this output file is converted to a BugView comparison file using a small utility, BlastToGCF (**Subheading 2.3.3.**). Launch this, choose “Load BLAST File” from the File menu, and locate and load the GridBLAST output file. After a short delay, the user should receive a message that the file has been read, with an invitation to view the list of protein pairs that has been generated. If all appears satisfactory at this stage, choose “Write gcf File” from the file menu. The user needs to enter the RefSeq numbers of the “Query” and “Database” Genomes (as in **Subheadings 7. and 8.**) and then save with a suitable name and .gcf extension. The resulting file will now typically be only 50 K in size.

3.1.4. Editing Comparison Pairs

Whether a comparison file has been generated automatically, as in **Subheadings 3.1.3.2. and 3.1.3.3.**, or from within BugView, it will generally be necessary to add or delete comparison pairs on the basis of visual inspection or scientific knowledge. To illustrate how this is done, we shall take as starting point the situation where an empty comparison file has been constructed (**Subheading 3.1.3.1.**).

3.1.4.1. LOCATING AND EDITING GENES OF INTEREST

When starting with an empty comparison file it is likely that there are specific genes the user wants to compare. These can be located by using the “Find” or “Search” facilities.

1. A “Find” dialogue can be evoked by clicking the eponymous button on the console, or by using the standard keyboard shortcut (command-F or control-F, depending on platform). A gene can be found by entering an ID (gi number), name, or product (see **Note 13**)—entering a gene name such as “trpS” might be a typical example. This example is likely to give a single “hit” on each genome, with the first hit being selected and its name highlighted. Using control-G or command-G the user can cycle through all the hits in the display window.

2. In some genomes, the gene names are unhelpfully designated as `cds_1`, and so on. If the gi number of a gene of interest is unknown, attempting to locate it on the basis of its product will be the best option. In this case, the “Search” facility (console button) is preferred. Thus, a term such as “polymerase” might bring up a list of all RNA and DNA polymerase subunits, and so on, allowing the user to choose the subunit of interest. As different product names in the list are selected, the corresponding genes are selected and their names highlighted in the display window (see **Note 14**).
3. To inspect a gene of interest, use the “Focus On” button in the console. This zooms the gene to the highest magnification at which it will fit into the display window. At this stage, the zoom factor can be decreased by using the console slider to see the context of the gene. Clicking on the “Gene Info” button in the console (or double-clicking on the gene) will open an information window for the gene. (The gi number may be of particular interest for forming a pair, see **Subheading 3.1.4.2**.) The user can change from the “Information” view to the “DNA sequence” and “protein translation” views by clicking on the appropriate buttons (see **Note 15**).
4. In cases where the genome is poorly annotated, the user may wish to add annotations or change the name or gene-product information. This can be done by selecting the gene and clicking the “Edit Info” button in the console (or transferring directly from the previous “Gene Info” window by clicking the “Edit” button). The gene category is edited separately by clicking the “Edit Category” button in the console, and choosing from the categories listed. Edits are included in the `.gda` file after saving from the File menu.

3.1.4.2. ADDING PAIRS

We shall consider two different situations in which one would be adding a new pair to the comparison strip. The first is where the user has identified the two genes from which the user wishes to create a pair.

1. Usually the pairwise alignment of the two genes will be checked. Select one gene by clicking on it, and then click the “Single” button on the console in the “Pairwise Comparison” group (**Fig. 1**). Paste the ID (gi number) of the second gene into the field marked “Query Gene ID” and click “Start.” Local and Global pairwise alignments will be performed, with the Local alignment being displayed. Generally the user’s scientific judgment will determine whether an alignment is significant or not. As a rough guide, our experience is that there is likely to be a significant similarity when the “Score” is greater than about 120 (see **Note 16**).
2. To make a Pair, click the “Make Pair” button that becomes active after the comparison has run. This brings up an “Add Pair” dialog with the gi numbers for the two genes already entered. (This dialog can also be invoked from the “Add Pair” button in the console after selecting one of the genes in the display window. In this case, the second gi number would have to be entered or pasted.) Click “OK” to create the pair.

3. The two genes and the comparison strip will be selected, and the “Co-align” button in the console will allow them to be viewed together in their respective genome contexts. This will often facilitate identifying other pairs when one is working on a gene cluster. The new pairs are included in the .gcf file after saving from the File menu.
4. The second situation to be considered is the location of a gene of interest on one genome, but searching by name or product does not identify a corresponding gene on the other genome.
5. Select the gene of interest in the display window, and click “Batch” in the “Pairwise Comparison” group of the console (**Fig. 1**). Click the “Start” button.
6. Typically 2000 comparisons will take no more than a few minutes (*see Note 17*). The three best matches will be displayed, and the user can choose from a pull-down menu, which (if any) of these to make pairs from, and then click the “Make Pair” button. Thereafter, proceed as in **Subheadings 3. and 4.**

3.1.4.3. DELETING PAIRS

The user may wish to delete some biologically spurious pairs from automatically aligned genomes.

1. Select the alignment pair to be deleted. This can be done by selecting either of the genes in the pair or, better (but generally more difficult), the strip between them.
2. Click the “Delete Pair” button in the console.
3. If user has selected the strip between pairs, or a gene that has no other pairs, a confirmation dialogue will appear. If the user has selected a gene that is a member of more than one pair, the user must choose from a list of pairs that have appeared. After the gi numbers of the pair, the percentage identity (local alignment) is displayed in parentheses to help distinguish between alignments of different quality.

3.1.5. Traversing and Reviewing Comparison Pairs

Generally after the user has generated a genome alignment automatically the user may wish to go through the genome, reviewing the comparison pairs that have been assigned, and considering genes that appear not to have counterparts in the other genome. Three approaches to this are described.

3.1.5.1. MANUAL TRAVERSAL

In manual traversal, start at the beginning of one genome and examine paired and unpaired genes, scrolling down. Although straightforward, an example of this procedure is described to illustrate some of the facilities in BugView.

1. Click on a gene near the “top” of the first genome (or start from a known position using “Find” or “Search”) and then click the “Focus On” button in the console

- (**Fig. 1**). This zooms the gene to the highest magnification at which it will fit into the display window (**Fig. 2A**).
2. “Click” on the scrollbar “up triangle” to scroll to the very start of the genome, even though it may not be evident that there is still “play” here (**Fig. 2B**).
 3. Select the first gene by clicking on it. If it is a member of a pair, the “Co-align” button on the console will become enabled, if not it will remain dimmed.
 4. For the first gene that is a member of a pair, click on the “Co-align” button on the console. **Figure 2C** shows a typical result of such an alignment for two strains of *Neisseria meningitidis* (see **Note 18**). The first group of genes on the genome of one strain align to a group starting at gene 248 on the second strain. (The genomes are, of course, circular, but the origin of replication is used as a reference point for the “start.”)
 5. Scroll down the genome. Using the scrollbar to do this can often be unsatisfactory for large genomes at high magnifications. In this case, it is better to use the keyboard “up” and “down” arrows (which scroll half a window at a time). For even finer adjustment, use the mouse pointer—if the mouse is scrolled within an area of the display window to the right of the genomes, the pointer changes to a “hand,” which can be used to scroll the window interactively by small amounts.
 6. As the user scrolls, insertions or deletions can cause the alignment pairs to diverge increasingly from horizontal. The user can realign, as in **step 4**, or, more conveniently, interactively using the mouse pointer with the “alt” key depressed. (Here, the cursor changes to a hand with the forefinger extended.)
 7. At the end of a homology block, select a gene in this region and zoom out using the slider on the console. Clicking the “Centre” button in the console will maintain

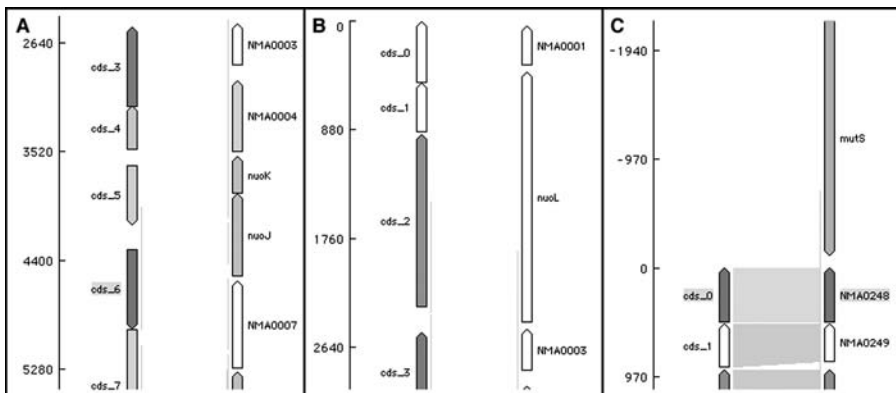


Fig. 2. Genome alignment in BugView. (A) Uppermost visible part of left genome after clicking and focusing. (B) The previous after clicking the top of the vertical scrollbar. (C) Region of first gene in the left-hand genome after coalignment with related gene in right-hand genome.

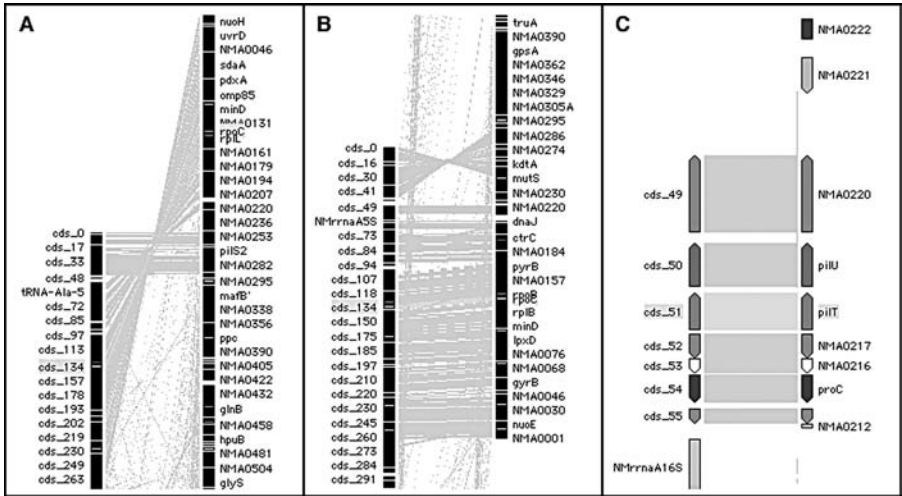


Fig. 3. Reversal of relative genome direction in BugView. (A) View of first two blocks of related genes in the genomes of two strains of *Neisseria meningitidis* showing the second block of genes with relatively reversed orientation below the first block of aligned genes with the same orientation. (B) The previous after reversing directions and coaligning. (C) The previous after focusing on the first genes in the second block and decreasing the magnification slightly.

the region of interest in the center of the display window as long as the zoom level is above one.

8. In the *N. meningitidis* comparison, a second block of genes in the first strain can be seen to be aligned to those at the “start” of the genome of the other strain, but in an inverted orientation—a very common situation (Fig. 3A). To review this second block, first, select a gene near the middle of it, then choose “Reverse Directions” from the View menu, click the “Co-align” button (this gives the alignment shown in Fig. 3B), and then the “Focus On” button. Scroll back to the start of the group and continue as before (Fig. 3C). If it is necessary to restore the original orientation of the genomes at any stage, this is done by choosing “Restore Directions” (which will have replaced “Reverse Directions”) from the View menu. (The default alignment can be restored by clicking the “Revert” button in the “Align Pair” group on the console.)

3.1.5.2. THE TRAVERSE FACILITIES

An alternative to manual traversal, or an adjunct to it, is to review separately the pairs and the unpaired genes using the traverse facilities. This is probably of most interest for examining the unpaired genes, especially in the case of different strains of the same bacterium.

1. To work with all the gene pairs, choose “Traverse Pairs” from the Pairs menu. In the dialogue box click the “load” button. A list of paired genes will appear in the window. The pairs can be traversed by scrolling or using the “up” and “down” arrow keys. A pair selected in this window will also be selected in the genome display window, and can be coaligned and centered without closing the traversal window.
2. To review all the unpaired genes, choose “Traverse Unpaired Genes” from the pairs menu. The names of the unpaired genes from both genomes will appear in the window. Traversal is as for pairs in **step 1**.

3.1.5.3. USING THE MATRIX VIEW

It can be difficult to keep track of a position in a genome while traversing gene pairs from genomes in which the gene order has diverged significantly. Using the Matrix view in conjunction with the display view in the main window can help in this respect.

1. Choose “Matrix Genome Comparison” from the Diagram menu. A dot-matrix comparison of the genomes will be displayed (*see Note 19*). A typical example, in which each dot represents a homologous pair, is presented in **Fig. 4**. Pairs with the same orientation follow a diagonal from top left to bottom right, whereas those with opposite orientation follow a diagonal from bottom left to top right (and are colored red for ease of identification).
2. The horizontal and vertical guideline tools can be used to mark the blocks of related genes (or gaps), numbering them for reference with the text tool (*see Note 20*). The annotated matrix can be printed or saved as a graphic file.
3. Having defined a particular region of alignment of the genomes, the user can transfer to that region on the main genome display window. This is done by enclosing the region in a small rectangle using the selection tool in the Matrix display (#1 in **Fig. 4**), and then clicking the “Transfer” button. In the main window, the region selected will be zoomed and, if possible (*see Note 21*), centered.

3.1.5.4. THE PAIR DISPLAY RANGE FACILITY

In **Fig. 4** it can be seen that there is a pull-down menu entitled “Identity Cutoff.” This allows the restriction of the display of the matrix comparison to pairs, the percentage identity of which is greater or equal to the number selected (40% in this case). A similar way of filtering the pairs to be displayed is available in the main window, and can be accessed by selecting “Set Pair Display Range” from the Pair menu. This is more sophisticated than the option in the Matrix view as it allows the user to set both upper and lower limits for display. The facility is useful for reviewing those automatically generated pairs that have a relatively low identity. The pairs listed in the Pair Traversal window—**Subheading 3.1.5.2**.—also reflect this selection.

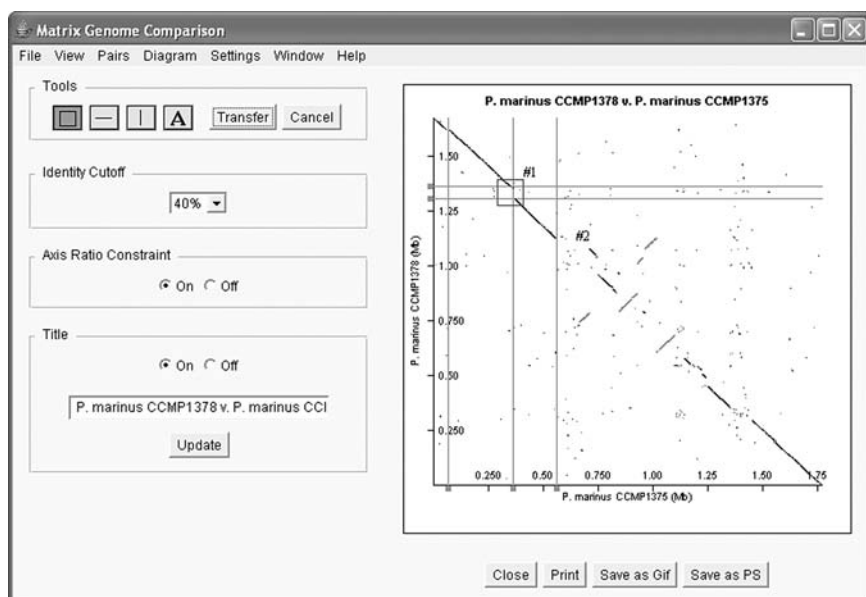


Fig. 4. View of the Matrix Genome Comparison window of BugView. The figure shows a comparison of two marine cyanobacterial genomes in which only pairs with at least 40% local identity are displayed. A couple of annotations have been made, and a few horizontal and vertical guidelines have been added. These latter can be moved by their square handles or deleted by “alt-clicking.” A rectangular selection has been made with the selection tool (highlighted), and clicking on the “Transfer” button would take the user to the corresponding region in the main display window.

3.1.6. Further Aspects of the BugView Interface

In **Subheading 3.1.1.**, we described the general features of the BugView interface, and in this and subsequent sections the description of controls focused mainly on visible features—the menus, the console buttons, and so on. Although these controls are likely to be the main ones employed by users familiarizing themselves with BugView, they do involve frequent—and ultimately tedious—mouse movement. For users who have become proficient with the basic operation of BugView there are some extra controls for more efficient working (in addition to keyboard equivalents for some of the menu items).

3.1.6.1. KEYBOARD CONTROL

It has already been mentioned (**Subheading 3.1.5.1.**) that the user can scroll using the “up” and “down” arrows of the keyboard. There are also keyboard

controls for zooming, focusing, centering, and several other functions. These are listed in the “Mouse and Keyboard control” item of the Help menu.

3.1.6.2. CONTEXT-SENSITIVE MENUS

At any time, pressing the right mouse button (control-pressing on the Macintosh platform) will invoke a pop-up menu, the contents of which are dependent on the position of the pointer. The menus available when the pointer is on a gene or a pair are of the most interest, where the options available are roughly equivalent to those that are available from the console if the gene or pair is selected. Their selection from the pop-up menu is obviously quicker than from the console, and is especially advantageous where an operation is being performed repeatedly on a set of genes or pairs.

3.2. *Internal Genome Structure*

The information in the previous section (**Subheading 3.1.**), describing the use of BugView for visualizing genome comparisons, is, in many cases, also applicable to surveying the genes in a single genome. However, not yet mentioned are the specific facilities BugView provides for visualizing groups of genes of similar function—genes in preset categories or those defined by the user. These are dealt with in this section.

3.2.1. *Predefined Gene Categories*

The predefined categories to which genes may be automatically assigned using a suitable .ptt file (**Subheading 2.2.2.**), or manually with the “Edit Category” control (**Subheading 3.1.4.1.**), have already been introduced. In fact, the categories in BugView extend the COG categorization—stable RNA genes have been added (and are assigned automatically when converting a Genbank file), and additional categories for virulence and for inactive genes are also included. The full list can be viewed by choosing “Category Colour Key” from the Help menu.

3.2.2. *Custom Gene Sets*

Although the functional categories available in BugView cannot be modified by the user, it is possible to create custom sets that can be used in certain visualizations. This is not entirely straightforward, so a hypothetical example is described.

1. Choose “Create Custom Set” from the Diagram menu.
2. Let us suppose the user wishes to visualize genes associated with the function of RNA polymerase in *N. meningitidis*. If more than one genome is loaded, choose

the genome of interest, and then enter the term “polymerase” as “Search String” and click the “Search” button.

3. The results include not only RNA polymerases, but DNA polymerases and a polyA polymerase. Select each of the latter and click “Remove”.
4. This decreases the list to eight entries, but does not include relevant terms such as “sigma” and “rho,” which may occur in the absence of the term “polymerase”. The list is extended by searching for these terms (or previously identified known gene products) and removing any duplicates.
5. Type a name for the set (e.g. “RNApol”) and click the button “Create Set”. It is important to realize that at this stage, the set is available for use in the current session, but must be saved to disc for use in subsequent sessions. This is done by selecting “Write Custom Set” from the Diagram menu, selecting the appropriate submenu, and giving the set a name such as “RNApol.set”. The visualization of custom sets is described in **Subheading 3.3.3**.
6. The set can be loaded in a subsequent session by choosing “Load Set” from the Diagram menu.

3.2.3. Finding Repeated Genes

It may be that the user is interested in genes present in multiple copies, rather than those related by a specific function (**Subheading 3.3.2**). If the user can identify a member of such a gene family, the user can perform pairwise alignments to search for other family members.

1. Select the member of the gene family and click the “Internal” button in the “Pairwise Comparison” group on the console (**Fig. 1**).
2. A dialogue box with a progress window will appear. Click “Start”.
3. When all the comparisons have been performed, those above a preset threshold will be listed. (The default is 100, but this can be altered by choosing “Internal Comparison Filter” in the Settings menu.) Of these, alignments for the best three (again customizable) will be displayed. Those of interest can be noted, their gene information edited, and a custom set constructed from them, as in **Subheading 3.3.2**.

3.2.4. Gene Category Displays

The genomic location of genes of different categories can be displayed in either horizontal or circular orientation.

3.2.4.1. CIRCULAR DISPLAY

This is obtained by choosing “Circular Diagram” from the Diagram menu. An example is illustrated in **Fig. 5**, showing a format that is frequently employed in publications and presentations. Up to four different gene categories can be

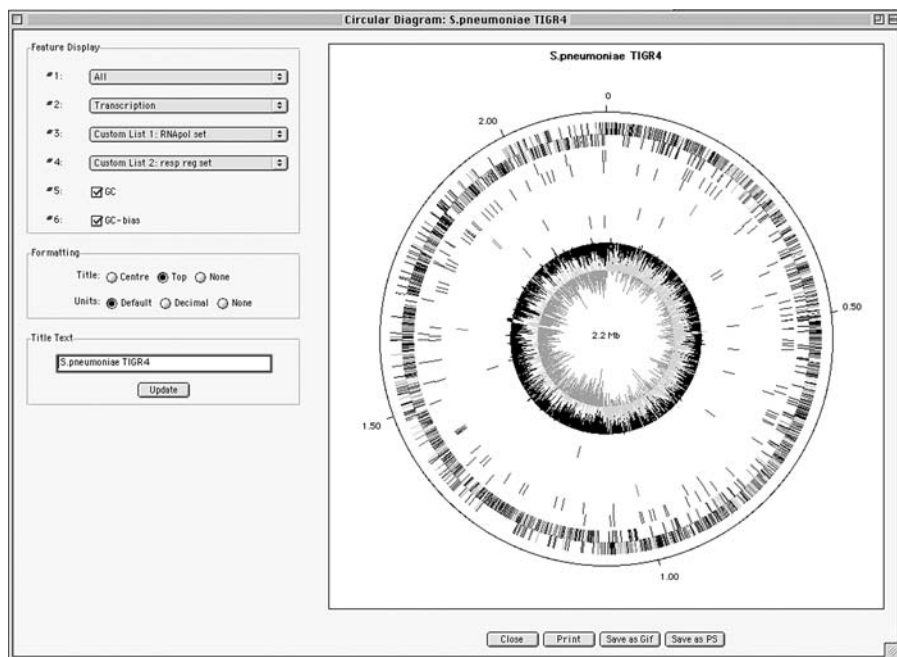


Fig. 5. View of the circular diagram window of BugView. The figure shows a display of different sets of genes of *Streptococcus pneumoniae* arranged in concentric circles. The outer circle shows all genes (their directionality indicated by whether they are outside and inside an imaginary central diameter), the second shows one of the preset COG categories (in this case Transcription), and the third and fourth show custom categories generated by the user within BugView (RNA polymerases and response regulators, respectively). Plots of GC-content and GC-bias are also displayed.

represented, including the custom sets of **Subheading 3.3.2.**, and the strand on which a gene resides is indicated by whether the gene (represented as a short line) is outside or inside the conceptual circle traced by the genome. GC-content and GC-bias can also be represented. The diagram (as is true for the contents of the main genome display window) can be printed or saved as either a gif graphic (suitable for web use or slide presentation) or a postscript file, which may be more suitable for publication (*see Note 22*).

3.2.4.2. LINEAR DISPLAY

The linear display is obtained by choosing “Linear Diagram” from the Diagram menu. It was introduced primarily as a means of viewing the whole of a genome at a scale that allowed individual genes to be distinguishable and

identifiable by color category. (The gene direction indication can be turned off in this view to make better use of the area available.) The names of individual genes are shown on “mouse over,” and clicking on an individual gene takes the user to that gene in the main window. Alternatively, the user can view up to three categories of gene together in this display, which may be useful in certain situations.

3.3. Web Deployment

BugView differs from the web-based Java applet, Der Browser (4), from which it was developed, by being a desktop application and having gene-comparison features that the applet lacked. However, after the original description of the BugView application (1), it was decided that it would be useful to provide an applet version—BugView/weB—to enable users to make web presentation of genome comparisons generated in the desktop application. This applet version is available from <http://www.gla.ac.uk/~dpl1n/BugView/bvapplet.html>, and is described for the first time.

3.3.1. The Scope of BugView/weB

For security reasons web applets have restrictions placed upon them, with the result that the scope of BugView/weB is more limited than that of the BugView application.

1. File read and write is not allowed. This means that the web author has to provide the files for the genomes and comparisons to be displayed, and the user is not able to edit pairs, print (printing the web page may not work), or save the graphic view (except using screen capture software). Instructions for referencing BugView files from the webpage are given in **Subheading 3.3.2**.
2. Menus are not allowed in applets. In the event, many of the menu items and some of the console button items are redundant in this context, so the console has been simplified (**Fig. 6**). The remaining functions on the console are for navigating the genomes and viewing information on genes and pairs. To this end, the controls for reversing directions and displaying GC-bias have been transferred from menu items to console buttons. (Although not mentioned previously, inclusion of GC-bias in the main display window is available by choosing “Display Other Features” in the View menu of the application.)
3. “Help” is available from the “?” button in the console, although as a pop-up web page. Users may therefore need to be warned to disable pop-up blocking if they wish to use the “Help” facility.
4. Context-sensitive menus are still available, but their contents differ from those in the desktop application. In the absence of menus, and because of the pressure of

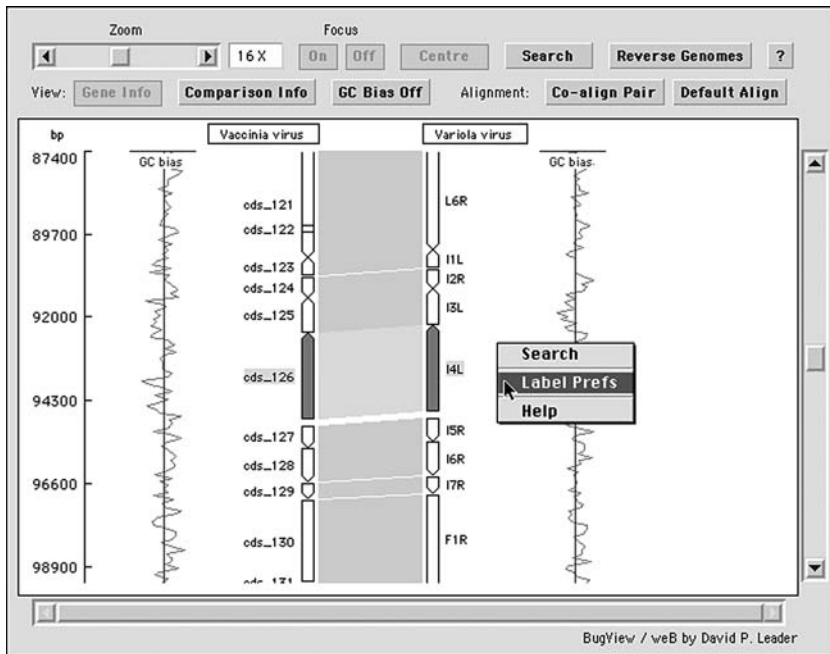


Fig. 6. View of BugView/web. A close-up view of part of a comparison of pox viruses is shown. The figure also illustrates a context-sensitive menu (invoked by right click—control-click on the Macintosh) and a parallel display of GC-bias. The view is of the applet portion of the webpage only. Comparison with Fig. 1 allows the user to see which console features have been removed and which menu features added to the console.

space on the console, changing label preferences is offered on the menu when one right/control clicks outside a gene or pair (Fig. 6).

5. The size of files that can be loaded in the applet appears to be limited to about 1.5 Mb. Thus, the user can only serve sequence files for small genomes such as the poxvirus genomes illustrated.
6. Although the main interest in BugView/web is for presenting the comparison of two genomes with multiple comparison pairs, it is also possible to present a series of individual genomes side-by-side.

3.3.2. File Organization and HTML Mark-Up

Figure 7 shows the organization of supporting files in relation to the webpage containing the applet. The BugView files are referenced within the <applet> tag as parameters, which can have values of “datan,” “seqn,” or “comparison.”

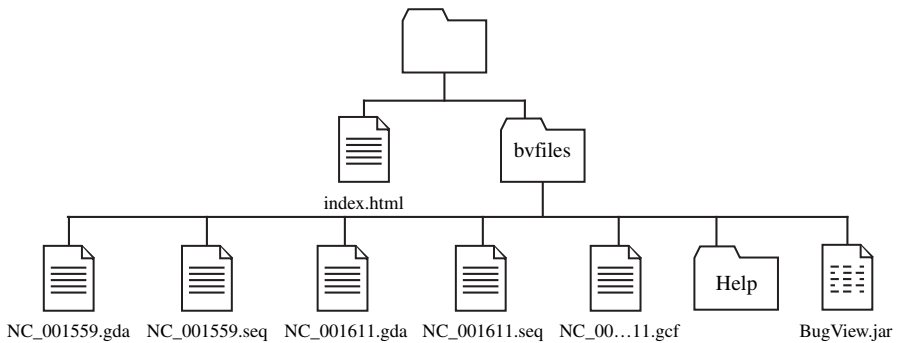


Fig. 7. Organization of BugView/web files on a web server. Directories are represented as folders, and files as text documents. The example listed in the text is illustrated. The top directory is not named, but might typically be a user's public_html directory on a Unix server. The index.html file (it need not have this name) is the webpage in which the applet is marked up as in the text. A directory, bvfiles, contains all the other files associated with the applet. These are the five files generated by BugView, the Help folder and contents (which are included in the applet download) and the applet itself, BugView.jar, which is an archive of all the compiled files of the BugView/web program. (If the name of the directory, bvfiles, is changed, the applet markup on the webpage must be changed accordingly.)

The “*n*” in “*datan*” and “*seqn*” represents an integer which must start at 1, followed by successive values—2, 3, and so on—without omissions. An example, being the markup for the comparison in [Fig. 6](#), is:

```

<APPLET codebase="bvfiles" code="BugView.class"
archive = "BugView.jar" width="640" height="500">
  <PARAM NAME="progressbar" VALUE="true">
  <PARAM NAME="data1" VALUE="bvfiles/NC_001559.gda">
  <PARAM NAME="sequence1" VALUE="bvfiles/NC_001559.seq">
  <PARAM NAME="data2" VALUE="bvfiles/NC_001611.gda">
  <PARAM NAME="sequence2" VALUE="bvfiles/NC_001611.seq">
  <PARAM NAME="comparison" VALUE="bvfiles/NC_001559-NC_
001611.gcf">
  You need Java to view this applet.
</APPLET>

```

where “bvfiles” is the name of the directory (folder) in which the BugView files are located. The width and height can, of course, be altered to suit individual circumstances (although a narrower width will not accommodate the

applet) and the “progressbar” parameter is optional (and the bar itself is not displayed with older versions of Java). Web pages with the applet marked-up in this manner should have a “Transitional” Document Type Definition (see **Note 23**).

4. Notes

1. The website <http://www.gla.ac.uk/~dpl1n/BugView/> is available from Glasgow University, where the author is currently employed. Should he move elsewhere, he will attempt to ensure that users are forwarded appropriately from this url. However, in any case, either the software or redirection to a new url will be found in the author’s private webspace at <http://www.q7design.demon.co.uk/BugView/>.
2. It would appear, for example, that at least 100Mb of free RAM is needed to convert a Genbank file for a 5-Mbp genome. BugView will inform the user if there is insufficient memory to perform a file conversion.
3. There is a bug in Java 1.4 for Mac OS X that prevents pasting into text areas and fields. Initially this version of Java was standard for Mac OS X 10.4. This bug has been fixed in Java 1.5, which can be installed using “Software Update” or from the developer section of Apple’s website (<http://devworld.apple.com/java/>).
4. This page can be reached by using the Entrez interface at the NCBI website to search for a particular genome (<http://www.ncbi.nlm.nih.gov/gquery/>).
5. The user can be deceived by the fact that the “wait cursor” disappears after the first file (the data file) has been created, even though creation of the second (the sequence file) is still occurring and takes much longer to complete. If the program has insufficient memory available to process the file, the user will receive an error message. In this case, it is worth quitting BugView, quitting all other unnecessary programs, and trying again. On Mac OS8/9 freed memory can remain fragmented after quitting applications, so it is advisable to restart the computer before retrying.
6. COG information can only be loaded when there is a single genome in the BugView window—if more than one genome has been loaded the menu item will appear dimmed. The reason for this is that the .ptt files contain no internal RefSeq numbers from which the program can determine to which genome they relate.
7. If the user is uncertain of the RefSeq numbers of the files loaded into BugView, they can be checked quickly by choosing “Genome and Pair Summaries” from the View menu.
8. A project file can be generated at any time that all five files for a genome comparison (two .gda files, two .seq files, and the .gcf file) are loaded into BugView. Choose “New Project File” from the file menu, and save to the same directory as the associated data, sequence, and comparison files with a name that references their RefSeq numbers and has the extension .prj (e.g., NC_003112-

NC_003116.prj). On subsequent occasions, all five files can be opened at once by choosing “Open Project” from the File menu.

9. The appropriate way to run formatdb in this case is:

```
formatdb -i 'RefSeqNo.faa' -p T -o
```

where ‘RefSeqNo.faa’ should be replaced by the actual name of the input file.

10. If the comparison file fails to load it could be because the .faa file used to create it was from a later release of the genome than that used for the data file. In this case, the comparison file might reference a gene not annotated in the earlier data file, a situation that versions of BugView before 1.3.3 could not handle. The remedy is to upgrade to v1.3.3 of BugView (or higher) in which the bug was fixed.
11. Depending on the speed of the desktop machine, it may take 1 h or so to update pair scores (which is why one is given a chance to change one’s mind or interrupt the process). One is advised to turn off screen-savers or auto-sleep settings before starting.
12. Because some institutions ban traffic from nonstandard ports like 9081, it is intended eventually to change the url to <http://cassini.nesc.gla.ac.uk/wps/portal>. Certain features of the site require a modern version of Java, but this is not required for the actual BLAST search. The choice of web browser should not be critical, but if one has problems with older browsers (e.g., Internet Explorer 5.1 for Macintosh) one is advised to try a more modern browser.
13. The initial default category for Find and Search is “product” but changes to reflect the most recent selection.
14. Having located a gene of interest in the Search facility and having dispelled the dialogue box, it is all too easy to click inadvertently in the display area and lose the selection. This can be avoided by making sure the mouse cursor is over the console.
15. If the “DNA” and “Protein” buttons are dimmed it will almost certainly be because the .gda file has been loaded without the .seq file.
16. The local alignment (which only displays “good” regions of similarity) is generally of more interest at this stage. The “Score” is relative, being greater as the length of the protein increases. Thus, for two alignments with 100% identity, that for a short protein will score less than that for a longer one. The percentage identity values are much cruder than the “Score” as they are based simply on the number of matches and mismatches—they do not allow for the fact that similar amino acids are likely to be conserved, or that the alignment of rare amino acids is more significant than that of common ones.
17. In a batch alignment, proteins will be skipped if they exceed the maximum size set in Preferences (this defaults to 1000 amino acids, but can be changed from the Settings menu). Such proteins will be listed in the output so that the user can repeat the comparison if he has available a machine with a sufficiently fast processor.

18. These have RefSeq numbers NC_003112 and NC_003116, for those who wish to reproduce this example.
19. It should be emphasized that this matrix is based on the user's preassigned comparison pairs—it is not generated from programmatic whole-genome comparison in BugView.
20. Select the horizontal or vertical tool and then click where the guideline should be positioned. The position of the guideline may be adjusted by dragging the square handle, and the guideline may be removed by alt-clicking the handle (the cursor should change from a cross-hair to an arrow first). Text can be edited in a relatively crude manner after it has been clicked.
21. Centering is not always possible when one gene is at one of the extremities of the genome in the display window. In such cases, it should be easy enough to identify the region corresponding to that selected in the Matrix view.
22. The user is cautioned that although the postscript format provides scalable vector graphics—enabling the user to obtain high quality text—the quality of the postscript line graphics generated from BugView is limited by the resolution of the screen display because arithmetical “rounding” occurs. Postscript graphics can be viewed in the free Ghostscript viewer for Unix and Windows or in Preview on Mac OS X, but are best opened and edited in a professional vector graphics application such as Adobe Illustrator.
23. It is conceivable that at some time in the future web browsers may no longer support the “Transitional” <applet> tag. In such an eventuality, revised markup instructions will be mounted on the BugView/weB webpage.

Acknowledgments

The author would like to thank Micha Bayer and Richard Sinnott of the BRIDGES project for setting up GridBLAST for use with BugView, and all those users who have stimulated improvements by providing feedback on the program.

References

- 1 Leader, D. P. (2004) BugView: a browser for comparing genomes. *Bioinformatics* **20**, 129–130.
- 2 Tatusov, R. L., Galperin, M. Y., Natale, D. A., and Koonin, E. V. (2000) The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.* **28**, 33–36.
- 3 Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (2000) Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410.
- 4 Grigoriev, A., Levin, A., and Lehrach, H. (1998) A distributed environment for physical map construction. *Bioinformatics* **14**, 252–258.

CGAS

A Comparative Genome Annotation System

Kwangmin Choi, Youngik Yang, and Sun Kim

Summary

Recent advances in genome sequencing technology and algorithms have made it possible to determine the sequence of a whole genome quickly in a cost-effective manner. As a result, there are more than 200 completely sequenced genomes. However, annotation of a genome is still a challenging task. One of the most effective methods to annotate a newly sequenced genome is to compare it with well-annotated and closely related genomes using computational tools and databases. Comparing genomes requires use of a number of computational tools and produces a large amount of output, which should be analyzed by genome annotators. Because of this difficulty, genome projects are mostly carried out at large genome sequencing centers. To alleviate the requirement for expert knowledge in computational tools and databases, we have developed a web-based genome annotation system, called CGAS (a comparative genome annotation system; <http://platcom.org/CGAS>). This chapter describes how to use CGAS and necessary background knowledge on the computational tools and resources. As an example, a *Bacillus subtilis* genome is considered as an unannotated target genome and compared with several reference genomes, including *Bacillus halodurans*, *Oceanobacillus iheyensis* HTE831, and *Bacillus cereus* group genomes (representative strain of *Bacillus. cereus*, *Bacillus anthracis*).

Key Words: Comparative genomics; genome annotation; Bidirectional Best Hit (BBH); sequence clustering; protein domain; genome context.

1. Introduction

The remarkable success of genomics in the past decades was achieved largely by technological advances in DNA sequencing as well as genomic

information process. Biologists currently have full access to the whole genome sequences of several hundreds of microbial organisms as well as a handful of eukaryotic species, including yeast (*Saccharomyces cerevisiae*, *Schizosaccharomyces pombe*), nematode (*Caenorhabditis elegans*), fruitfly (*Drosophila melanogaster*), thalecress (*Arabidopsis thaliana*), rice (*Oryza sativa japonica*), rat (*Rattus norvegicus*), house mouse (*Mus musculus*), and human (*Homo sapiens*). One of the most effective methods to annotate a newly sequenced genome is to compare it with well-annotated and closely related genomes using comparative genomics tools. Comparing genomes requires the use of a number of computational tools and produces a large amount of output data, which should be analyzed by human annotators. Thus, it is very challenging for biologists, even for bioinformaticians, to annotate a newly sequenced genome by combining computational tools and databases. For this reason, genome projects are mostly carried out at large genome sequencing centers. It is expected that new high-throughput sequencing technologies will help biologists in sequencing a genome easily and there is a pressing need to improve the computational environment for genome annotation. We have been developing a system for computational comparative genomics, PLATCOM (<http://platcom.org/platcom>). To alleviate the requirement for expert knowledge in computational tools and databases for genome annotation, we have developed a web-based genome annotation system, called CGAS (a comparative genome annotation system; <http://platcom.org/CGAS>) on top of PLATCOM.

CGAS is a system where users can upload a newly sequenced genome and annotate it in comparison with several genomes of their choice. Comparing functional sequences (e.g., gene) of different species is a powerful method for interpreting genomic information, because the evolution rate of functional sequences tends to be much slower than nonfunctional ones. By comparing the genome sequences at different evolutionary distances, biologists can computationally detect conserved coding and noncoding regions, and also identify unique sequences for a given species. CGAS offers a web-based computational procedure for genome in six core steps: (1) open reading frame (ORF) identification from a newly sequenced whole genome, (2) six-frame translation of ORF into amino acid sequences, (3) sequence similarity search against databases, (4) protein family assignment based on sequence similarity. For more refined decision, (5) motif or functional domain search and (6) genome context analysis (i.e., gene neighborhood search) can be used for additional steps.

In this chapter, we describe steps for protein function annotation of a newly sequenced genome in detail and also discuss tools for each step. Each step

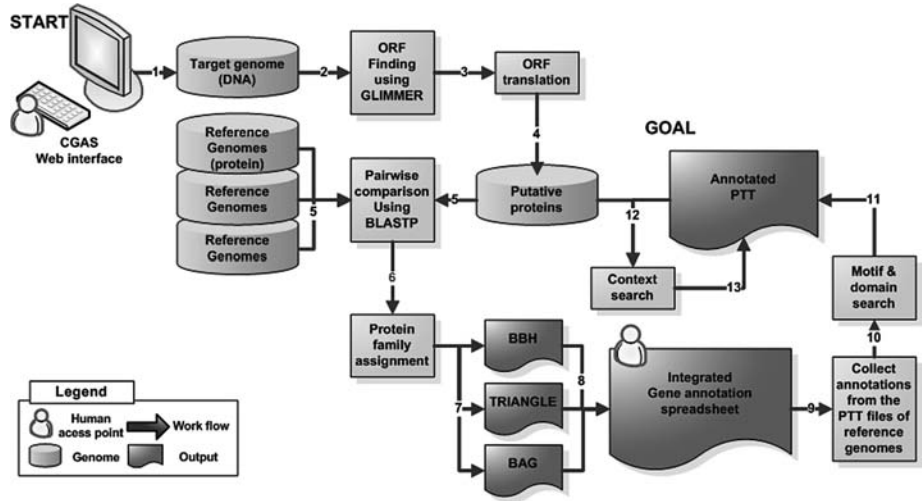


Fig. 1. Overview of a comparative genome annotation system (CGAS).

can be done automatically or semiautomatically, but it is worth reminding that gene prediction is literally a “prediction,” which requires experimental data to prove it and manual correction steps are often required. A roadmap to protein function annotation is briefly described in [Fig. 1](#).

2. Materials

The goal is to annotate genes in a target genome in comparison with reference genomes. In this section, we used a target genome and nine reference genomes as an example annotation task using CGAS.

1. Files that a user needs to maintain. The annotation of several thousands to tens of thousands genes cannot be finished in a day, so a user needs to save and upload files so that the annotation task can resume later. Let us denote xxx be a name for the target genome.
 - a. xxx.fna: a DNA sequence of the target genome.
 - b. xxx.faa: a set of ORFs in FASTA format that are predicted from xxx.fna. Any gene prediction algorithm can be used for predicting ORFs from xxx.fna. In addition to ORFs, we need information on the positions of ORFs on xxx.fna, which are specified in a file, xxx.ptt (*see item 1c*). We recommend to use our web server where Glimmer3 is set up to generate automatically both xxx.fna and xxx.ptt (*see item 1c*).

- c. xxx.ptt: information on the gene in the target genome. It is in the same format as the ptt files at National Center for Biotechnology Information (NCBI). Initially, it only contains gene identifiers and their positions on xxx.fna. The annotation information that the user will type in will be stored in this file.
 - d. xxx.tbl: this file stores information on gene matches between the target genome and the selected reference genomes. The user will need to perform this task “only once,” but this file should be stored locally and uploaded each time the annotation task resumes. The gene match information includes BBH (bidirectional best hits), TRIANGLE (triangles of BBHs), and BAG (the result of BAG clustering of all proteins in the target genome and the selected reference genomes [1]).
2. Target genome: *Bacillus subtilis* (NC_000964).
 - a. A user can download the whole genome sequence file (NC_000964.fna) from NCBI ftp site (<ftp://ftp.ncbi.nih.gov/genomes/>). This file is an input for ORF identification and translation step. Sample FAA and PTT files of *B. subtilis* with annotation information removed are provided at CGAS webpage (<http://platcom.org/CGAS>).
3. Reference genomes: a user needs to select reference genomes that the target genome will be annotated in comparison with. In this example, we selected nine genomes from the same phylogenetic group (bacillaceae), which *B. subtilis* belongs to and groups close to bacillaceae (listeriaceae, staphylococcus). CGAS provides a web interface to select these genomes, so all the user has to do is to select them on the web. In this chapter, four genomes very close to *B. subtilis* are used as reference genomes: (taxonomical hierarchy: Bacteria > Firmicutes > Bacillales > Bacillaceae).
 - a. *Bacillus halodurans* (NC_002570).
 - b. *Bacillus cereus* ATCC 10987 (NC_003909).
 - c. *Bacillus anthracis* strain Ames (NC_003997).
 - d. *Oceanobacillus iheyensis* HTE831 (NC_004193).
4. ORF finding and six-frame translation of the target genome.
 - a. Archaeal and bacterial ORFs typically consist of uninterrupted stretches of DNA between a start codon (usually ATG, but some genes use GTG, TTG, or CTG) and a stop codon (usually TAA, TGA, or TAG), but some bacteria (e.g., mycoplasmas) have only two stop codons. In multicellular eukaryotes, most genes are interrupted by introns and this makes eukaryotic ORF prediction far more complicated than prokaryotic cases. Most of ORF finders also perform six-frame translation by default.
 - b. There are many algorithms and software tools for gene identification, but this chapter only uses GLIMMER available at its webpage (<http://www.cbcb.umd.edu/software/glimmer> [2]). The package consists of two main programs: the first program to run is the training program, build-icm. This

program takes as an input set of sequences and builds and generates the interpolated Markov models. These sequences can be complete genes or just partial ORFs. The second program is GLIMMER, which uses the interpolated Markov models to identify putative genes in an entire genome.

5. Generating xxx.tbl file from xxx.faa.

- a. CGAS provides an interactive spreadsheet for users to save, upload, and manage annotation data using simple data type. CGAS generates BBH, TRIANGLE, and BAG results based on all-to-all protein pairwise comparisons between target and reference genomes, and the result is stored in a file xxx.tbl.
- b. The BBH method is a widely used homology based method to define a computational counterpart concept to orthology, which generally results in a single gene in one genome being predicted to be the ortholog of a single gene in the other genome. The BBH method has been used in various function prediction studies, such as the construction of a conserved coexpression network and the prediction of regulatory motifs. PLATCOM system maintains a BBH database based on Pairwise Comparison Database (3).
- c. The TRIANGLE method expands BBH concept for three genomes, i.e., triangles formed by BBHs, by using the BBH database in PLATCOM system.
- d. BAG (1) is a sequence clustering program and available online (<http://platcom.informatics.indiana.edu/CLASSEQ>).

6. Motif and structural domain search tools.

- a. Analysis of structural and sequence pattern information of a protein is complementary to similarity-based analysis and helps predict protein structure, cellular localization, or a protein family. Furthermore, identification of certain structural features of proteins, such as signal peptides, transmembrane segments, or coiled-coil domains, may provide some functional clues even in the absence of detectable homologs by similarity-based analysis.
- b. PROSITE (4) is a database of protein families and domains. Proteins or protein domains that belong to a particular family generally share functional attributes and it is generally assumed that they are derived from a common ancestor. Among protein sequences that belong to the same family, some regions are often better conserved than others during evolution. These regions are generally important for the function of a protein or for the maintenance of its three-dimensional structure. PROSITE v19.0 currently contains 1639 patterns and profiles specific for protein families or domains. Each of these signatures comes with documentation providing background information on the structure and function of these proteins. PROSITE is also available at <http://www.expasy.org/prosite/>.
- c. PRORULE (5) provides additional information about PROSITE profiles, the position of structurally or functionally critical amino acids and the condition

on the proteins to maintain their biological role. PRORULE is also available at <http://www.expasy.org/prosite>.

- d. SCOP (6) provides a detailed and comprehensive description of the structural and evolutionary relationships between all proteins whose structure is known, including all Protein Data Bank entries. The SCOP classification of proteins has been constructed manually by visual inspection and comparison of structures. It provides assistance of tools to make the task manageable and help provide generality. SCOP is available at <http://scop.mrc-lmb.cam.ac.uk/scop>.

7. Genomic context analysis tools.

- a. When multiple genomes are compared, the context of gene matches can increase the accuracy of similarity-based sequence annotation (7–9). Such contexts include phylogenetic profiles of protein families, domain fusion events in multidomain proteins, gene neighborhood and synteny, and expression regulation patterns (e.g., operon). A very simple rationale based on genome context analysis is that genes whose products are involved in closely related functions (e.g., subunits of a multisubunit enzyme or components of functional coupling) should all have similar, if not identical, phylogenetic patterns and their expression should be regulated coordinately because the selective pressure probably comes from the necessity to synchronize their expression. This method has been successful in characterizing gene functions even when experimentally characterized homologs to the gene do not present. CGAS provides three different tools for contextual analysis.
- b. OperonViz (<http://platcom.org/platcom>) is a tool for generating graphical visualization of gene neighborhoods and synteny across selected genomes and allow users to navigate the gene neighborhood across genomes. Protein family information may come from COG classification or *de novo* protein family classification by BAG clustering. Gene clusters are identified by requiring a intergenic distance (200 bp by default). OperonViz is useful to identify conserved synteny, horizontal gene transfers, functional coupling, and functional hitchhiking.
- c. MCGS (<http://platcom.org/platcom>) is a tool that predicts a set of physically clustered gene sets using a novel hybrid gene team model (10,11). To study cooccurrences of functionally related genes in multiple genomes, MCGS employs a novel hybrid pattern model that combines the set and the sequential pattern models, i.e., gene clusters with or without physical proximity constraint.
- d. ComPath (<http://platcom.org/compath>) is a metabolic pathway analysis tool using comparative genomics approach, where a selected metabolic pathway in multiple genomes can be compared with various sequence analysis tools. ComPath is based on KEGG database (12) and EC convention (<http://www.expasy.org/enzyme>) and provides various functions to reconstruct metabolic pathways using sequence similarity-, domain-, structure-, and genomic context-based methods.

3. Methods

1. Prepare unannotated target genome sequence in FASTA format.
2. Identify ORF using GLIMMER (Step-1 in CGAS system).
 - a. We strongly recommend that a user perform all the steps in this section at our website automatically. Visit CGAS web server (<http://platcom.org/CGAS>), select “Step-1,” and upload the genome sequence file. The input file should be in a FASTA format. If a user uses the web server, go to **Subheading 3.3.** directly.
 - b. In the case that a user wants to run a gene-finding program on a local computer, the user should follow the following steps. A user can run GLIMMER or other gene finding programs to generate both faa file and ptt file, two standard format data files, i.e., NCBI’s protein sequence file (*.faa) and protein translation table (*.ptt). In this chapter, we explain how to use GLIMMER.
 - c. Go to GLIMMER webpage (<http://www.cbc.umd.edu/software/glimmer>) and download the most updated version of GLIMMER package. Install the package in a local machine. We are using GLIMMER 3.01 Beta version for this chapter.
 - d. Run long-orfs that takes an unannotated target genome sequence file (in FASTA format) as an input and outputs a list of all “potential genes.” Now the user has the gene location information that will be used for genome context analysis in **Subheading 3.6.**

```
long-orfs-no_header -t 1.0 genome genome.longorfs.
```
 - e. Run extract to extract the long ORF sequences from the output at the previous step.

```
extract-nostop $genome genome.longorfs > genome.train.
```
 - f. Run build-icm to build a collection of interpolated Markov models from the training data in glimmer.seq.

```
build-icm -r genome.icm < genome.train.
```
 - g. Run glimmer3. A list of ORFs with their scores and a collection of Markov models (glimmer.icm) will be generated. A set of the putative genes will be output as well.

```
glimmer3 [option] genome genome.icm genome.
```
 - h. GLIMMER automatically performs six-frame translation and reports best translations in two result files (xxx.detail, xxx.predict). These outputs contain gene position and strand information (**Fig. 2**). A PTT-format file should be prepared for further context sequence analysis.
3. Inspect the predicted ORFs manually if needed.
 - a. The correctness of ORF identification may need to be manually checked for genomes with a high GC content (e.g., *Halobacterium salinarum*) because start codon prediction turned out to be highly error-prone. Several methods can be used for inspection: (1) sequence homology search against protein databases,

ID	Frame	----- Start -----			Stop	--- Length ---		Scores -----									
		of	Orf	of Gene		of	Orf	of Gene	Raw	InFrame	F1	F2	F3	R1	R2	R3	NC
0001	+2	17	20	139	120	117	-6.46	0	96	0	-	0	-	-	3	1.00	
	+2	140	242	361	219	117	-2.14	0	54	0	-	45	-	-	0	1.00	
	-1	435	351	148	285	201	5.44	26	73	0	-	26	-	-	0	1.00	
	+2	668	668	790	120	120	0.96	0	99	0	-	-	-	-	0	1.00	
	-3	899	839	717	180	120	-1.73	0	55	-	-	-	-	-	0	43 1.00	
	-1	936	933	808	126	123	-0.04	6	27	-	-	6	-	-	-	66 1.00	
	-3	1124	1106	918	204	186	-2.03	0	99	-	-	-	-	-	0	0 1.00	
	+1	4	4	1398	1392	1392	5.93	99	99	-	-	-	-	-	-	0 1.00	
	-2	1750	1720	1457	291	261	0.14	12	-	-	-	-	12	-	87	1.00	
	-2	1957	1945	1751	204	192	-1.42	0	-	-	69	-	0	-	10	1.00	
0002	-3	2078	2063	1908	168	153	-2.76	0	-	-	41	-	-	-	0	58 1.00	
	-2	2308	2293	2174	132	117	0.73	0	-	-	93	-	0	-	5	1.00	
	+3	1542	1641	2756	1212	1113	5.40	99	-	-	99	-	-	-	0	1.00	
	-3	2807	2729	2616	189	111	-4.52	0	0	-	99	-	-	-	0	0 1.00	
	-3	2948	2927	2808	138	117	-1.90	0	99	0	-	-	-	-	0	0 1.00	
	+2	2789	2816	2977	186	159	0.18	0	99	0	-	-	-	-	0	0 1.00	
	-2	2980	2947	2792	186	153	2.00	0	99	0	-	-	-	-	0	0 1.00	
	-1	3228	3174	2887	339	285	4.52	0	99	-	-	0	-	-	0	0 1.00	
	-2	3352	3304	3086	264	216	0.54	0	99	-	-	-	-	-	0	0 1.00	
	+2	3362	3386	3502	138	114	-8.96	0	99	0	-	-	-	-	0	0 1.00	
0003	+1	2605	2761	3834	1227	1071	8.07	99	99	-	-	-	-	-	-	0 1.00	
	-1	3891	3828	3679	210	147	-6.05	0	99	-	-	-	-	-	0	0 1.00	
	-1	4233	4152	4009	222	141	-2.76	0	-	59	-	-	-	-	40	1.00	
0004	+2	3719	3785	4264	543	477	3.22	99	-	99	-	-	-	-	-	0 1.00	
	-3	4292	4259	4119	171	138	-0.02	0	-	97	-	-	-	-	0	1 1.00	
	+3	4242	4320	4442	198	120	-4.55	0	-	-	0	-	-	-	-	99 1.00	
	+3	4671	4677	4808	135	129	-1.90	0	0	99	0	-	-	-	0	0 1.00	
	-1	5016	4845	4717	297	126	-0.44	0	0	99	-	0	0	-	-	0 1.00	

Fig. 2. The final output of GLIMMER3.

(2) amino acid composition analysis (e.g., codon bias, oligonucleotide composition), (3) existence of typical ribosome-binding sites (i.e., Shine-Dalgarno) and promoter followed by the ORF, and (4) integration of genomic and experimental proteomic data.

- b. You may compare the GLIMMER result with those from other ORF finding programs including CRITICA (http://geta.life.uiuc.edu/~gary/GJO_programs.html), GeneMark (<http://opal.biology.gatech.edu/GeneMark/>), [13] NCBI ORF finder (<http://www.ncbi.nlm.nih.gov/gorf/gorf.html>), or the translation tool on the ExPASy server (<http://www.expasy.org/tools/dna.html>).
- 4. Search BBH and TRIANGLE and perform sequence clustering using BAG (Step-2 in CGAS system).
 - a. This step generates a xxx.tbl file by using three computational methods, BBH, TRIANGLE, and BAG clustering.
 - b. If you already have both xxx.faa and xxx.ptt files from the previous step, visit CGAS (<http://platcom.org/CGAS/>), and select “Step-2.”
 - c. Upload the unannotated sequence file, xxx.faa (i.e., target genome) prepared in previous section. This file has to be in the FASTA format.
 - d. Select two or more reference genomes from the genome list. If the phylogeny of target genome is already known, the selected genomes should be close relatives of the target genome.
 - e. Annotate protein functions based on BBH, TRIANGLE, and BAG results between target and reference proteins by using spreadsheet (see Subheading 3.5.).

- 5. Protein function annotation (Step-3 in CGAS system).
 - a. If the user already has xxx.faa, xxx.ptt, and xxx.tbl files from the previous steps, click “Step-3,” button in the CGAS frontpage. This will generate an annotation spreadsheet.
 - b. The “Summary” button displays the summary of annotations for matching genes by BBH, TRIANGLE, and BAG (see Fig. 3).
 - c. The “Analysis” button allows interactive sequence analysis. Use this feature to confirm the function of a target gene (see Fig. 3).
 - d. The user can cut-and-paste from the output by clicking “Summary” button or from the webpage by following links (many links on our webpage will lead to the original source information, e.g., sites at NCBI or Swissprot).
- 6. Sequence analysis for the gene being annotated and its matching genes.
 - a. A user can perform various sequence analysis, including the multiple sequence alignment, Gibbs motif search, and so on. A list of analysis tasks will

16077075

Annotate here

DNA gyrase, A subunit

Update

FASTA

E-value	Z-score	Utility commands
e-10	100	Summary Analysis

Bidirectional Best Hit (BBH)

NC_002570	NC_003909	NC_003997	NC_004193	Utility commands
15612570	42779087	30260201	23097462	Summary Analysis

TRIANGLE

NC_002570	NC_003909	NC_003997	NC_004193	Utility commands
15612570	42779087	-	-	Summary Analysis
15612570	-	30260201	-	Summary Analysis
15612570	-	-	23097462	Summary Analysis
-	42779087	30260201	-	Summary Analysis
-	42779087	-	23097462	Summary Analysis
-	-	30260201	23097462	Summary Analysis

BAG

NC_002570	NC_003909	NC_003997	NC_004193	Utility commands
15612570	42782666	30260201	23099149	Summary Analysis
15614702	42779087	30263544	23097462	

Fig. 3. Gene annotation spreadsheet in comparative genome annotation system (CGAS) system.

change over time. **Figure 4** is a snapshot of the current sequence analysis page.

- 7. Motif and structure analysis.
 - a. Use motif and domain analysis to refine the annotation of genes.
 - b. Perform PROSITE and/or PRORULE search (*see Subheading 2.6.*).
 - c. Perform SCOP search (*see Subheading 2.6.*).
- 8. Synteny and gene neighborhood analysis.
 - a. MCGS (*see Subheading 2.7.* and **Fig. 5**).
 - b. OperonViz (*see Subheading 2.7.* and **Fig. 6** for an example).
- 9. Metabolic pathway analysis. Users can also compare the ORFs in terms of metabolic pathways using ComPath (<http://platcom.org/compah>).
 - a. Choose a metabolic pathway.
 - b. Choose a set of reference genomes. The same set of reference genomes as for CGAS analysis can be chosen or a different set of genomes can be chosen—not

Sequence & Genomic Context Analysis

Sequence Analysis Tools

You have chosen the following sequences. Unselect the ones you do not wish to have.

	Genome	Entrez	CDD	Uniprot	Pfam	Prosite
☒	/tmp/Target_19958.faa	16077072	oasis_pfam_16077072	RECF_BACSU	PF02463	PS00617 PS00618
☒	NC_002570	15612567	oasis_pfam_15612567	RECF_BACHD	PF02463	PS00617 PS00618
☒	NC_003909	42779085	oasis_pfam_42779085	RECF_BACC1	PF02463	PS00617 PS00618
☒	NC_003997	30260199	oasis_pfam_30260199	RECF_BACAN	PF02463	PS00617 PS00618
☒	NC_004193	23097459	oasis_pfam_23097459	RECF_OCEIH	PF02463	PS00617 PS00618

Select Sequence Analysis Method

- 1. ☐ BLASTP
- 2. ☐ FASTA
- 3. ☐ HMMer
- 4. ☐ CLUSTALW
- 5. ☐ PHYLIP
- 6. ☐ ATV
- 7. ☐ BAG using FASTA with Zscore of
- 8. ☐ BAG using BLATP with Zscore of
- 9. ☐ Gibbs Sampler with motif length of

Fig. 4. Sequence analysis page in comparative genome annotation system (CGAS) system.

- all genomes are available at ComPath because it uses the genome information at KEGG.
- Then upload a query sequence file, i.e., xxx.faa file for the target genome. Then the screenshot will look like one next (Fig. 7).
 - Each enzyme in the target genome ("Your_Seq" column) can be computationally predicted by checking the "Your_Seq" column and the corresponding enzyme, say 1.2.1.19, and then choosing the computational methods for prediction, SCOP and SUPERFAMILY search, HMM, HMM-common shared region predicted by BAG, and contextual search.
 - The predicted genes in the target genome will appear in the cell, which was initially empty. Note the colors of genes predicted show which computational method predicted the genes.
 - The user can perform computational tests on the predicted genes using sequence analysis tools. As of now, phylogentic analysis, Gibbs motif sampler, and BAG clustering are provided. Amino acid sequences can be extracted for further analysis.

Gene Team Tools (MCGS)

- orthologous: ☒ COG ☐ BAG ☐ Triangle
- disregard the strand direction ☐
- use linear model for genomes ☐ (default: graph model)
- allowing distance between two genes
- minimum number of genes
- minimum number of known family
- the number of strong association (Type 1)
- the number of weak association (Type 3)
- output: association type ☒
positions of strong association ☒
positions of weak association ☒
- and aligning (target) genome of
- and aligning (target) gene of

The following will be used for reference genomes

- NC_002570 = NC_002570=Bacillus halodurans, complete genome.
- NC_003909 = NC_003909=Bacillus cereus ATCC 10987, complete genome.
- NC_003997 = NC_003997=Bacillus anthracis str. Ames, complete genome.
- NC_004193 = NC_004193=Oceanobacillus ihayensis HTE831, complete genome.

Run MCGS!

Fig. 5. Parameter setting page for MCGS.

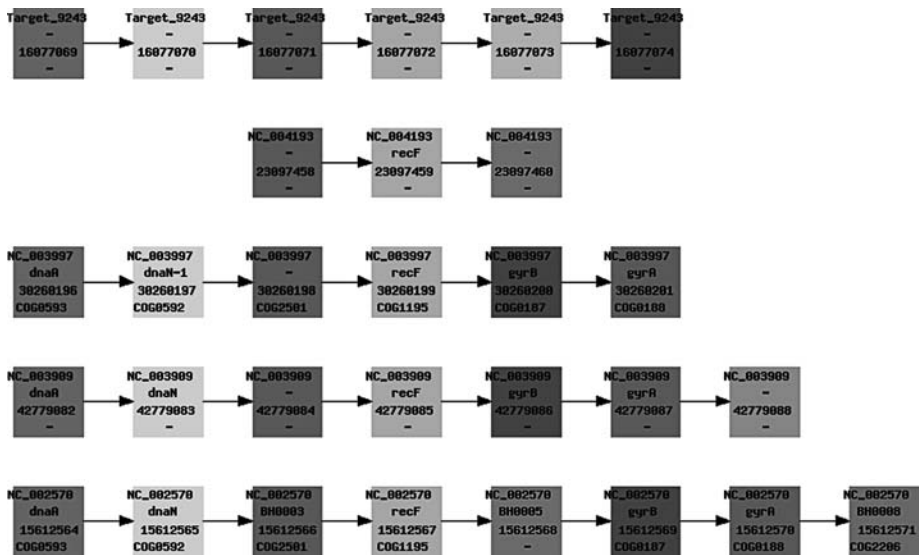


Fig. 6. Reconstruction of gene neighborhood using OperonViz. Each color represents a protein family. Users can navigate the gene neighborhood structure on each chromosome by clicking any square.

IV. Genome-EC table

- 1. Genomes to be searched can be limited by checking checkbox(es) on the top row. If not, ComPath uses all genomes in this table
- 2. Please make sure that an EC number is set from the third column.

Merge	All or None	Select One	☐ bsu	☐ bha	☐ bca	☐ baa	☐ oih	☒ Your_Seq
☐	allnone	1.1.1.1	☒ BG11902☒ BG11941☒ BG13553	☒ BH1829	☒ BCE0198☒ BCE0742☒ BCE0877☒ BCE1099☒ BCE2252☒ BCE2296☒ BCE2604☒ BCE2677☒ BCE3145☒ BCE3521☒ BCE4453	☒ BA_2770☒ BA_3097☒ BA_3165☒ BA_3633☒ BA_4058☒ BA_5039	☒ OB1378☒ OB2738☒ OB2801	
☐	allnone	1.1.1.2	☐ BG12582				☐ OB0788	
☐	allnone	1.1.1.27	☐ BG19003	☐ BH3937	☐ BCE2007☐ BCE5032☐ BCE5135	☐ BA_0106☐ BA_2427	☐ OB3279	

Fig. 7. Interactive spreadsheet of ComPath.

4. Conclusion

We illustrated how a newly sequenced genome can be annotated using CGAS. The functionality of CGAS depends largely on that of the PLATCOM system, which provides a reconfigurable platform for comparing genomes. Thus, we expect to add more functionalities to CGAS as PLATCOM evolves.

Acknowledgments

This research was partially by NSF CAREER Award DBI-0237901 INGEN (Indiana Genomics Initiatives), and AVIDD (Analysis and Visualization of Instrument-Driven Data) Linux cluster.

References

- 1 Kim, S. and Lee, J. (2007) BAG: A Graph Theoretic Sequence Clustering Algorithm. *International Journal of Data Mining and Bioinformatics*. **1**(2), 178–200.
- 2 Delcher, A. L., Harmon, D., Kasif, S., White, O., and Salzberg, S. L. (1999) Improved microbial gene identification with GLIMMER. *Nucl. Acids Res.* **27**, 4636–4641.
- 3 Choi, K., Ma, Y., Choi, J.-H., and Kim, S. (2005) PLATCOM: a platform for computational comparative genomics. *Bioinformatics* **21**, 2514–2516.
- 4 Hulo, N., Bairoch, A., Bulliard, V., et al. (2006) The PROSITE database. *Nucl. Acids Res.* **34**, D227–D230.
- 5 Sigrist, C. J. A., De Castro, E., Langendijk-Genevaux, P. S., Le Saux, V., Bairoch, A., and Hulo, N. (2005) ProRule: a new database containing functional and structural information on PROSITE profiles. *Bioinformatics* **21**, 4060–4066.
- 6 Andreeva, A., Howorth, D., Brenner, S. E., Hubbard, T. J. P., Chothia, C., and Murzin, A. G. (2004) SCOP database in 2004: refinements integrate structure and sequence family data. *Nucl. Acids Res.* **32**, D226–D229.
- 7 Doerks, T., von Mering, C., and Bork, P. (2004) Functional clues for hypothetical proteins based on genomic context analysis in prokaryotes. *Nucl. Acids Res.* **32**, 6321–6326.
- 8 Wolf, Y. I., Rogozin, I. B., Kondrashov, A. S., and Koonin, E. V. (2001) Genome alignment, evolution of prokaryotic genome organization, and prediction of gene function using genomic context. *Genome Res.* **11**, 356–372.
- 9 Bork, P., Jensen, L. J., von Mering, C., Ramani, A. K., Lee, I., and Marcotte, E. M. (2004) Protein interaction networks from yeast to human. *Curr. Opin. Struct. Biol.* **14**, 292–299.
- 10 Kim, S., Choi, J. -H., and Yang, J. (2005) Gene teams with relaxed proximity constraint. *Proc. IEEE Comput. Syst. Bioinform. Conf.* 44–55.

- 11 Kim, S., Choi, J. -H., Saple, A., and Yang, J. (2006) A hybrid gene team model and its application to genome analysis. *J. Bioinform. Comput. Biol.* **4**, 171–196
- 12 Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., and Hattori, M. (2004) The KEGG resource for deciphering the genome. *Nucl. Acids Res.* **32**, D277–D280.
- 13 Lukashin, A. V. and Borodovsky, M. (1998) GeneMark.hmm: new solutions for gene finding. *Nucl. Acids Res.* **26**, 1107–1115.

BLAST QuickStart

Example-Driven Web-Based BLAST Tutorial

David Wheeler and Medha Bhagwat

Summary

The Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between protein or nucleotide sequences. The program compares nucleotide or protein sequences to sequence in a database and calculates the statistical significance of the matches. This chapter first provides an introduction to BLAST and then describes the practical application of different BLAST programs based on the BLAST Quick Start mini-course (www.ncbi.nlm.nih.gov/Class/minicourses). In each example, emphasis is placed on practical step-by-step procedures, although relevant theory is also given where it affects the choice of BLAST program, parameters, and database.

Key Words: NCBI; BLAST; mini-courses; MegaBLAST; human genome.

1. Introduction

BLAST is an acronym for Basic Local Alignment Search Tool and refers to a suite of programs used to generate alignments between a nucleotide or protein sequence, referred to as a “query” and nucleotide or protein sequences within a database, referred to as “subject” sequences. The original BLAST program used a protein “query” sequence to scan a protein sequence database. A version operating on nucleotide “query” sequences and a nucleotide sequence database soon followed. The introduction of an intermediate layer in which nucleotide sequences are translated into their corresponding protein sequences according to a specified genetic code allows cross-comparisons between nucleotide and

protein sequences. Specialized variants of BLAST allow fast searches of nucleotide databases with very large query sequences, or the generation of alignments between a single pair of sequences. Both the standalone and web version of BLAST are available from the National Center for Biotechnology Information (www.ncbi.nlm.nih.gov). The web version provides searches of the complete genomes of *Homo sapiens* as well as those of many model organisms, including mouse, rat, fruit fly, and *Arabidopsis thaliana*, allowing BLAST alignments to be seen in a full genomic context (1).

1.1. Query and Database Sequence Formats

BLAST “query” sequences are given as character strings of single letter nucleotide or amino acid codes, preceded by a definition line, beginning with a “>” symbol and containing identifiers and descriptive information. This format is known as FASTA. BLAST databases are constructed from concatenated FASTA formatted sequences using a program called “formatdb” that produces a mixture of binary- and ascii-encoded files containing the sequences and indexing information used during the BLAST search.

1.2. Scoring of Alignments and Substitution Matrices

A BLAST alignment consists of a pair of sequences, in which every letter in one sequence is paired with, or “aligned to,” exactly one letter or a gap in the other. The alignment score is computed by assigning a value to each aligned pair of letters and then summing these values over the length of the alignment. For protein sequence alignments, scores for every possible amino acid letter pair are given in a “substitution matrix” where likely substitutions have positive values and unlikely substitutions have negative values. By default, BLAST uses the “blosum62” matrix, a member of the most commonly used series of substitution matrix (2), however, several members of the PAM (3) series are also available. For nucleotide alignments, BLAST uses a reward of +2 for aligned pairs of identical letters and a penalty of −3 for each nonidentical aligned pair. The creation of a gap in an alignment results in a negative “gap-creation” penalty, with each extension of a preexisting gap incurring a lesser penalty. For a detailed treatment of the theory of alignment scoring (see ref. 4).

1.3. Overview of the Algorithm

BLAST begins a search by indexing all character strings of a certain length within the “query” by their starting position in the query. The length of the string to index, called the “wordsize” is configurable by the user. The allowable range for the “wordsize” varies according to the BLAST program used; typical values are 3 for protein-to-protein sequence searches and 11 for nucleotide

to nucleotide searches. BLAST then scans the database looking for matches between the “words” indexed in the “query” and strings found within the database sequences. For nucleotide-to-nucleotide searches, these matches must be exact; for protein-to-protein searches, the score of the match as determined using a substitution matrix, must exceed a specified threshold. When a word match is found, two nearby words in the case of protein searches, BLAST attempts to extend both forward and backward from the match to produce an alignment. BLAST will continue this extension as long as the alignment score continues to increase or until it drops by a critical amount owing to the negative scores given by mismatches. This critical amount is known as the “dropoff.” The methods BLAST uses to initiate refine alignments are given more fully in refs. 5 and 6.

1.4. Statistical Significance

The alignments found by BLAST during a search are scored, as previously described, and assigned a statistical value, called the “Expect Value.” The “Expect Value” is the number of times that an alignment as good or better than that found by BLAST would be expected to occur by chance, given the size of the database searched. An “Expect Value” threshold, set by the user, determines which alignments will be reported. A higher “Expect Value” threshold is less stringent and the BLAST default of “10” is designed to ensure that no biologically significant alignment is missed. However, “Expect Values” in the range of 0.001 to 0.0000001 are commonly used to restrict the alignments shown to those of high quality.

2. Course and Website

This BLAST Quickstart chapter illustrates the use of the principal BLAST programs to solve problems that arise in the analysis of protein and nucleotide sequences. Each section provides a succinct description of a protocol with two problems that serve as practical examples. Relevant theory is given when it affects the selection of a search strategy or search parameter, however, the emphasis is on the procedure itself. The sections follow closely the structure of the BLAST QuickStart Mini-Course found at www.ncbi.nlm.nih.gov/Class/minicourses. The BLAST QuickStart is one of 10 2-h format Mini-Courses offered by NCBI on campus at the National Institutes of Health and at locations around the country to over 4000 students a year. The courses use a paired problems approach in which the first of two similar problems or problem sets is solved by the instructor during the first hour on a computer linked to a projection system, while the students watch; in the second hour, the students tackle the second problem, or set of problems at their

own computers. These courses have been effective as practical introductions to bioinformatics procedures. To get the most from the sections next, it will be necessary to navigate to the URL previously listed and click on the “BLAST Quickstart” link to reach the online exercises, although the liberal collection of screen shots will allow the reader follow along for the most part without web access.

3. Nucleotide BLAST

Nucleotide BLAST refers to the use of a member of the BLAST suite of programs, such as “blastn” to search with a nucleotide “query” against a database of nucleotide “subject” sequences.

3.1. Available Nucleotide-Level Searches

There are two members of the BLAST suite of programs that are designed to make nucleotide-to-nucleotide alignments. The first is the original BLAST nucleotide search program known as “blastn.” The “blastn” program is a general purpose nucleotide search and alignment program that is sensitive and can be used to align tRNA or rRNA sequences as well as mRNA or genomic DNA sequences containing a mix of coding and noncoding regions. A more recently developed nucleotide-level BLAST program called MegaBLAST (7) is about 10 times faster than “blastn” but is designed to align sequences that are nearly identical, differing by only a few percent from one another. MegaBLAST allows the rapid mapping of a transcript onto a typical 3 billion base mammalian genome in seconds, and is useful for processing large batches of sequences. A refinement of MegaBLAST, known as discontinuous MegaBLAST, uses a discontinuous template to define an initial “word” in which characters in some positions, such as those in the wobble base position of codons, need not match. Discontinuous MegaBLAST allows rapid cross-species mappings involving coding regions in cases where species differences in codon usage would prevent alignments using the original MegaBLAST program.

3.2. Examples of Nucleotide BLAST Searches

3.2.1. Problem 1

Click on the link indicated by “P” next to the “Nucleotide-nucleotide BLAST (blastn)” to access the problem. This problem demonstrates how to use BLAST to find human sequences in GenBank that can be amplified with a particular primer pair. Access the nucleotide–nucleotide BLAST page (by clicking on the Nucleotide–nucleotide BLAST link). Paste both the forward and reverse

primers into the BLAST input box. Insert a string of about 30N's after the first primer sequence to separate the two sequences to be found in separate, not overlapping alignments. Limit your search to human sequences by selecting “*Homo sapiens*” from the “All organisms” pull down menu under the Options for advanced blasting and click the BLAST! link. Retrieve results by clicking on the “Format” button. Look for two hits to the same database sequence.

In this result, shown in Fig. 1, there are 13 GenBank entries that align to both the forward and reverse primers at different locations (indicated by thick bars) with a gap in between (indicated by a thin gray bar). There are two GenBank entries that align only to the reverse primer. One alignment of the primer pair to the GenBank entry L78833.1 is shown in Fig. 2. The forward primer aligns to the sequence L78833.1 on the forward strand (as indicated by

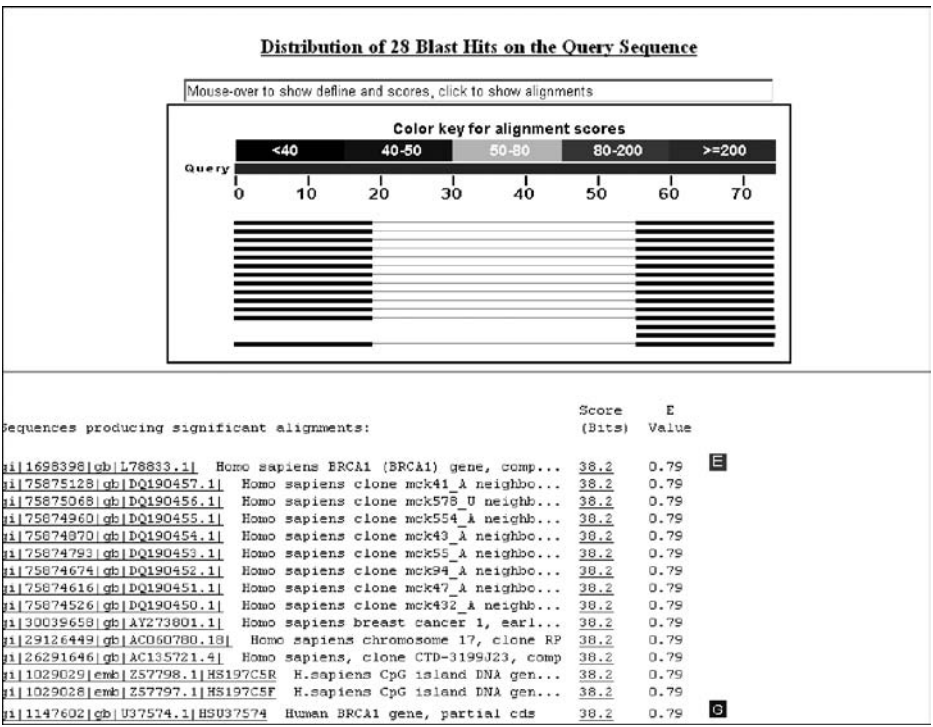


Fig. 1. Graphical overview of primer hits from the nucleotide–nucleotide search problem 1. The result shows that 13 GenBank entries align to the forward and reverse primers at different locations (indicated by thick bars) with a gap in between (indicated by a thin gray bar). There are two GenBank entries that align only to the reverse primer.

```

> gi|1698396|gb|L78833.1| Homo sapiens BRCA1 (BRCA1) gene, complete cds; ribosomal protein
L21-like protein (rpl21) pseudogene, complete sequence;
Rho7 (Rho7) and VatI (VatI) genes, complete cds; and unknown
(ifp35) gene, exons 1 through 3 and partial cds
Length=117143

Score = 36.2 bits (19), Expect = 0.79
Identities = 19/19 (100%), Gaps = 0/19 (0%)
Strand=Plus/Plus

Query 1      GTACCTTGATTTCGTATTC 19
          |||||
Sbjc 3252 GTACCTTGATTTCGTATTC 3270

Score = 38.2 bits (19), Expect = 0.79
Identities = 19/19 (100%), Gaps = 0/19 (0%)
Strand=Plus/Minus

Query 56     GACTCTACTACCTTTACCC 74
          |||||
Sbjc 3475 GACTCTACTACCTTTACCC 3457

```

Fig. 2. Alignment view of one of the primer hits to the GenBank entry L78833.1. The forward primer (query nucleotides 1..19) aligns to the sequence L78833.1 on the forward strand (indicated by Strand Plus/Plus) at nucleotides 3252..3270. The reverse primer (query nucleotides 56..74) aligns to the reverse strand (indicated by Strand Plus/Minus) at nucleotides 3475..3457. The Strand information is highlighted by rectangles and the nucleotide locations of L78833.1 are highlighted by ovals.

Strand Plus/Plus) at nucleotides 3252..3270. The reverse primer aligns to the reverse strand (as indicated by Strand Plus/Minus) at nucleotides 3475..3457. Thus, the two primers will amplify the sequence from nucleotides 3252..3475 of the entry L78833.1. Retrieve the entry L78833.1 in Entrez, by clicking on it. The annotation shows that the amplified region covers the Exon 1a and the upstream sequence of the *BRCA1* gene. Refer to the **Note 1** for the multiple hits. You may perform similar search against the human genome BLAST database (see **Note 2**).

3.2.2. Problem 2

Click on the link indicated by “H” next to the Nucleotide–nucleotide BLAST (blastn) to access the problem. This problem describes how to obtain single-nucleotide polymorphism (SNP) information in similar sequences in the database. Hermankova et al. (8) studied the HIV-1 drug resistance profiles in children and adults receiving combination drug therapy. To identify the SNPs in the HIV-1 isolates from these patients, or other similar sequences in the database, use the sequence from one of the patients given next and run a nucleotide–nucleotide BLAST search as described in the problem previously listed. Format the results using the “Flat Query with Identities” option from the “Alignment View” pull down menu under the “Format” options (see **Note 3**).

Query	5	CCTCAATCACTCTTTGGCAACGACCCCTCGTCACAAATAAGATAGGGGGGCAACTAAAG	64
23380210	1		60
23380202	1		60
15150145	1		60
7638172	1		60
7638170	1		60
7638168	1		60
23380208	1		60
23380206	1	G.....	60
23380204	1		60
23380200	1		60
15150149	1		60
15150147	1	G.....	60
51703160	1		60
51703042	1		60
44887180	1	G.....	60
13738955	1		60
7682537	19	G.....	78
51012122	1	G.....	60
6019233	1	G.....	60
37220926	183		242
63080064	1		60
9943154	1	A.....	60
9935201	1		60
6446433	19	G.....A.....	78
3096582	1806	G.....	1865

Fig. 3. Query-anchored alignment view for finding single-nucleotide polymorphism (SNP) from the nucleotide–nucleotide search problem 2. Nucleotides in the database sequences identical to the query HIV-1 sequence are indicated by dots and SNPs are indicated by the one letter nucleotide code. The A/G SNP at the query nucleotide 10 is highlighted.

Identify the SNP observed at alignment position 6 (query nucleotide number 10) in Fig. 3. There is an A/G SNP in many of the database sequences.

4. Protein BLAST

Protein-to-protein sequence searches are performed using the original member of the BLAST suite of programs, known as “blastp.”

4.1. Available Protein-Level Searches

The default wordsize for a blastp search is three; the default substitution matrix is the blosum62 matrix. Changing the wordsize from three to two increases the sensitivity of the search. Using a different substitution matrix can also have an effect on search sensitivity. During a “blastp” search, low-complexity regions of the query sequence are filtered to reduce the construction of spurious alignments and enhance search speed (see Note 4).

4.2. Examples of Protein BLAST Searches

4.2.1. Problem 1

Click on the link indicated by “P” next to “Protein–protein BLAST (blastp)” to access the problem. It describes how to use blastp to determine the type of protein. For this purpose, we will choose the database containing the curated and annotated protein sequences, such as RefSeq or Swissprot. Use the query sequence provided in the problem. This sequence was generated by translating a 5 *exon* gene from *Drosophila*. To determine the nature of this protein, run a blastp search. Access the “Protein–protein BLAST (blastp)” page by clicking on the link, paste in the query sequence, select the Swissprot database from the “Choose database” pull down menu and click on the BLAST! link. For each protein–protein search, the query is also searched against the Conserved Domain Database (*see Note 5*). Retrieve results by clicking on the “Format” button. The protein is similar to a number of aspartate amino transferases.

4.2.2. Problem 2

Click on the link indicated by “H” next to the “Protein–protein BLAST (blastp)” to access a similar problem to determine the type of protein. Use the query sequence provided in the problem. This sequence was generated by translating a 4 *exon* gene from *Drosophila*. To determine the nature of this protein, run a blastp search against the Swissprot database as described in **Subheading 2**. The protein is similar to a number of phosphoglucomutases.

5. Translated BLAST

Translated BLAST searches use a genetic code to translate either the “query,” database “subjects,” or both, into protein sequences, which are then aligned as in “blastp.” The translations are performed in the three forward as well as the three reverse reading frames so that no possible translation is missed.

5.1. Available Translated Searches

There are three varieties of translated BLAST search; “tblastn,” “blastx,” and “tblastx.” In the first variant, “tblastn,” a protein sequence query is compared to the six-frame translations of the sequences in a nucleotide database. In the second variant, “blastx,” a nucleotide sequence query is translated in six reading frames, and the resulting six-protein sequences are compared, in turn, to those in a protein sequence database. In the third variant, “tblastx,” both the “query” and database “subject” nucleotide sequences are translated in six reading frames, after which 36 (6×6) protein “blastp” comparisons are made. Protein sequences

are better conserved than their corresponding nucleotide sequences. Because the translated searches make their comparisons at the level of protein sequences, they are more sensitive than direct nucleotide sequence searches. A common use of the “tblastn” and “blastx” programs is to help annotate coding regions on a nucleotide sequence; they are also useful in detecting frame-shifts in these coding regions. The “tblastx” program provides a sensitive way to compare transcripts to genomic sequences without the knowledge of any protein translation, however, it is very computationally intensive. MegaBLAST can often achieve sufficient sensitivity at a much greater speed in searches between the sequences of closely related species and is preferred for batch analysis of short transcript sequences such as expressed sequence tags.

gi 11490591 gb AA070892.1	envelope glycoprotein [Human immunodeficiency virus 1]	190	2e-47
gi 1145173 gb AAA97852.1	envelope glycoprotein [Human immunodeficiency virus 1]	190	2e-47
gi 40850458 gb AAR95932.1	envelope glycoprotein [Human immunodeficiency virus 1]	186	4e-46
gi 40850456 gb AAR95931.1	envelope glycoprotein [Human immunodeficiency virus 1]	184	1e-45

Alignments			
> gi 18538741 gb AAL71647.1 envelope glycoprotein [Human immunodeficiency virus 1]			
gi 18538703 gb AAL71628.1 envelope glycoprotein [Human immunodeficiency virus 1]			
Length=201			
Score = 232 bits (591), Expect = 7e-60			
Identities = 110/112 (98%), Positives = 110/112 (98%), Gaps = 1/112 (0%)			
Frame = +1			
Query	268	TIAFNQSSGGDPEIVMHSFNCGGEFFYCNTTQLFNSTWPTNK-KSTNKTGTITLPCRIKQ	444
Sbjct	90	TIAFNQSSGGDPEIVMHSFNCGGEFFYCNTTQLFNSTWPTN KSTNKTGTITLPCRIKQ	149
Query	445	IINRUQEVGKAMYAPPIKQIRCSNITGIFLTRDGGNASDETETFRPGGGN	600
Sbjct	150	IINRUQEVGKAMYAPPIKQIRCSNITGIFLTRDGGNASDETETFRPGGGN	201
Score = 181 bits (460), Expect = 1e-44			
Identities = 89/89 (100%), Positives = 89/89 (100%), Gaps = 0/89 (0%)			
Frame = +2			
Query	2	EEDIVIRSENFNNAKTIIVQLKESIKINCTRPNNNTRKSIPIATGGAIYATSDIIGDIR	181
Sbjct	1	EEDIVIRSENFNNAKTIIVQLKESIKINCTRPNNNTRKSIPIATGGAIYATSDIIGDIR	60
Query	182	QAHCNLSRDQDNTLSQLVTKLREQFQNK	268
Sbjct	61	QAHCNLSRDQDNTLSQLVTKLREQFQNK	89
> gi 40850479 gb AAR95942.1 envelope glycoprotein [Human immunodeficiency virus 1]			
gi 18538655 gb AAL71604.1 envelope glycoprotein [Human immunodeficiency virus 1]			
gi 18538613 gb AAL71583.1 envelope glycoprotein [Human immunodeficiency virus 1]			
Length=201			

Fig. 4. Detecting frame shifts using a translated search: Blastx problem 1. The query, when translated in reading frame 2 (highlighted by a rectangle) up to nucleotide 268, is similar to only the first 89 amino acids of the database protein AAL71647.1. The translation of the query needs to be shifted to reading frame 1 (highlighted by an oval) to find similarity to the rest of the protein sequence.

5.2. Examples of Translated BLAST Searches

5.2.1. Problem 1

Click on the link indicated by “P” next to the “Translated query vs protein database (blastx)” to access the problem. This problem describes how to identify a frame shift in a nucleotide sequence by comparing its translated amino acid sequence to a similar protein in the database. Access the Blastx page by

> gi 19568959 gb AAL91985.1 envelope glycoprotein [Human immunodeficiency virus type 1]	
Length=514	
Score = 561 bits (1445), Expect = 3e-156	
Identities = 301/306 (98%), Positives = 304/306 (99%), Gaps = 0/306 (0%)	
Frame = +2	
Query 566	DIVFIDNDRNDSTSYRLSCNTSVITQACPQVSEFPIPIHYCAPAGFAILEKNNKTFSGT 745
Sbjct 185	DIVFIDNDRNDSTSYRLSCNTSVITQACPQVSEFPIPIHYCAPAGFAILEKNNKTFSGT 244
Query 746	GPCTNVSTVQCTHGIRPIVSTQLLNGSLAEIGIVIRYENITDNAKSIIQLNETVQINC 925
Sbjct 245	GPCTNVSTVQCTHGIRPIVSTQLLNGSLAEIGIVIRYENITDNAKSIIQLNETVQINC 304
Query 926	TRPNNNTRKSIPIGPGRAFYATGDIIGDIRKAYCNISGAKWNNTLKRIAYLKEQFPNKT 1105
Sbjct 305	TRPNNNTRKSIPIGPGRAFYATGDIIGDIRKAYCNISGAKWNNTLKRIAYLKEQFPNKT 364
Query 1106	IVFKPSSGGDPEIVMHSFNCRGEFFYCNTTKLFDSSWDXXXXXXXXXXXXXXXXXIIILPC 1285
Sbjct 365	IVFKPSSGGDPEIVMHSFNCRGEFFYCNTTKLFDSSWDNTLNKTUNNTWNKNSIIILPC 424
Query 1286	RIKQIINMWQEVGKANYAPPIEGPLYCLSMITGLILTRXXXXXXXXXXXXXXXXXFRPGGG 1465
Sbjct 425	RIKQIINMWQEVGKANYAPPIEGPLYCLSMITGLILTRDGGNETDGNNT G ETRFPGGG 484
Query 1466	NHRDNW 1483
	+NHRDNW
Sbjct 485	DHRDNW 490
Score = 315 bits (806), Expect = 4e-84	
Identities = 165/188 (87%), Positives = 173/188 (92%), Gaps = 4/188 (2%)	
Frame = +1	
Query 1	MRVKEKYQHLWRUGTHLLGLLMIRSAADQLUVTVYVGVPVWKEATTTPFCASDAKAYDTE 180
Sbjct 1	MRVKEKYQHLWRUGTHLLGLLMI SAADQLUVTVYVGVPVWKEATTTPFCASDAKAYDTE 60
Query 181	VHNVVATHACVPTDPNPQEVVLANVAENFNNDNNHVEQHEDIISLUDQSLKFCVKLT 360
Sbjct 61	VHNVVATHACVPTDPNPQEVVLAV ENFNNDNN+HVEQHEDIISLUDQSLKFCVKLT 120
Query 361	LCVTLNCTDKXXXXXXXXXXXXXXXXXXXXXXXXXGEIKNCFSFNITTAIRDVQR 540
Sbjct 121	LCVTLNCTD+ANNKTTTPNNT TT T +5+ + GEIKNCFSFNITTAIRDVQR 176
Query 541	GYALFSLK 564
	GYALF+KL
Sbjct 177	GYALENKL 184

Fig. 5. Detecting frame shifts using a translated search: blastx problem 2. The translation of the query sequence 1..564 in reading frame 1 (highlighted by a rectangle) is similar to the first 184 amino acids of the database protein AAL91985.1. The translation of the query needs to be shifted to reading frame 2 (highlighted by an oval) to find similarity to the rest of the protein sequence.

clicking on the link “Translated query vs protein database (blastx),” paste the nucleotide sequence provided in the problem in the query box and run the Blast search. The translation of the query sequence is similar to the sequences of envelope glycoproteins in the database. Compared to the similar proteins in the results, there appears to be a frame shift around nucleotide 268 as seen in [Fig. 4](#). The query when translated in reading frame 2 (as indicated by a rectangle) up to nucleotide 268 is similar to only the first 89 amino acids of the database protein AAL71647.1. The translation of the query needs to be shifted to reading frame 1 (as indicated by an oval) to find similarity to the rest of the protein sequence. To discover the nucleotide difference around 268, refer to **Note 6**

5.2.2. Problem 2

Click on the link indicated by “P” next to “Translated query vs protein database (blastx)” to access the problem. Paste in the sequence provided in the problem and run the blastx search to obtain a result similar to that shown in [Fig. 5](#). The translation of the query sequence 1..564 in reading frame 1 (as indicated by an oval) to find similarity to the rest of the protein sequence. There is a frame shift in the query nucleotide around 564. To find out the nucleotide difference around 564, refer to **Note 7**.

6. Genome BLAST

Genome BLAST refers to the application of any of the BLAST search programs to the complete genomic sequence of an organism or the transcript and protein sequences derived from its annotation.

6.1. Available Genome-Wide Searches

Genome BLAST services are available at NCBI for a variety of organisms including human, mouse, rat, fruit fly, and many others in a growing list. At a minimum, MegaBLAST and “blastn” searches against the complete genome are supported. These are usually offered in conjunction with “tblastn” searches against the genome, “blastp” and “blastx” searches against the proteins annotated on the genome and MegaBLAST, “blastn” and “tblastn” searches against collections of transcript sequences that have been mapped to the genome. Hits to the genome are displayed graphically within NCBI’s MapViewer to show their genomic context.

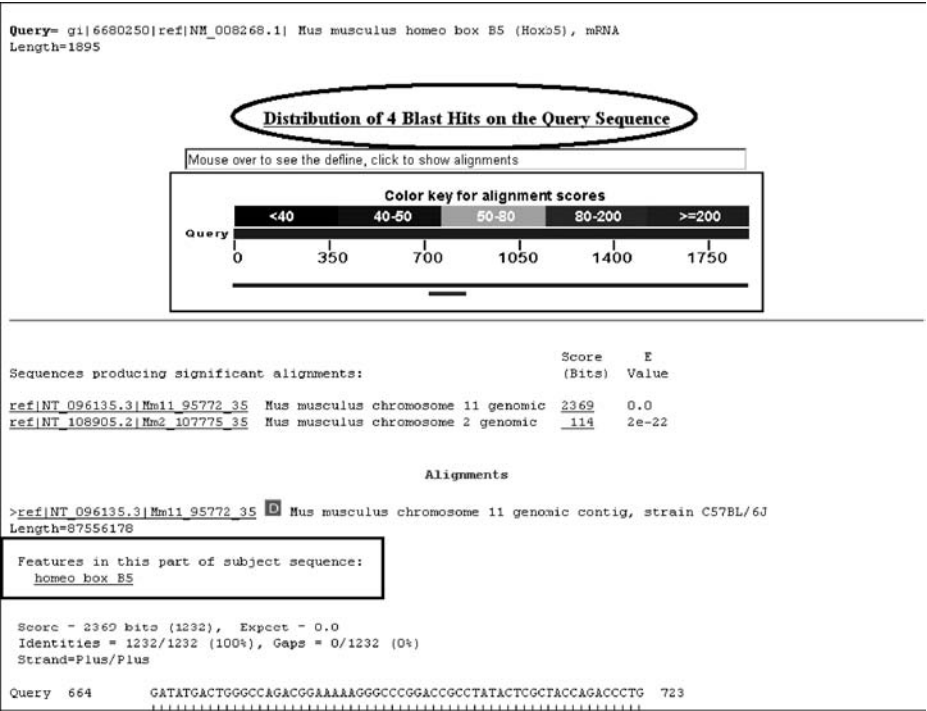
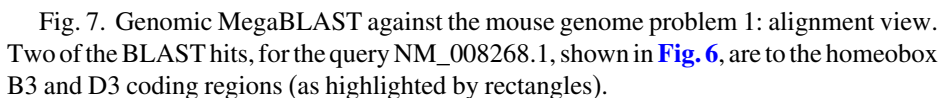


Fig. 6. Genomic MegaBLAST against the mouse genome problem 1: graphical overview. The mRNA query NM_008268.1 gets four hits to the mouse genome as highlighted by an oval. A part of the alignment view of the hit to the homeobox B5 coding region is also displayed (as highlighted by a rectangle).

6.2. Examples of Genome-Wide BLAST Searches

6.2.1. Problem 1

Click on the link indicated by “P” next to mouse genome BLAST to access the problem. This problem describes how to use mouse genome blast to identify the Hoxb homologues encoded by the mouse genome assembly sequence. As described in **Subheading 5.1.**, translated searches or protein–protein searches are more sensitive for identifying similarity in the coding regions than the nucleotide–nucleotide searches. Within the translated or protein–protein searches, tblastn will be more appropriate than blastx or blastp for this problem. Both latter programs will use protein databases consisting of



We will demonstrate the sensitivity of tblastn as compared to the nucleotide–nucleotide search to identify a similarity to a coding region by running two searches: (1) MegaBLAST the query mRNA sequence, NM_008268, against the mouse genomic sequence and (2) tblastn the query protein sequence, NP_032294, against the mouse genomic sequence. Access the mouse genome BLAST page, by clicking on the “mouse” link under the Genomes panel. For the first search, paste the accession number NM_008268 into the query box, accept the default MegaBLAST option, and select the “genome (reference only)” as the database. The results, shown in **Figs. 6** and **7**, contain only four hits, two to the two *Hoxb5*

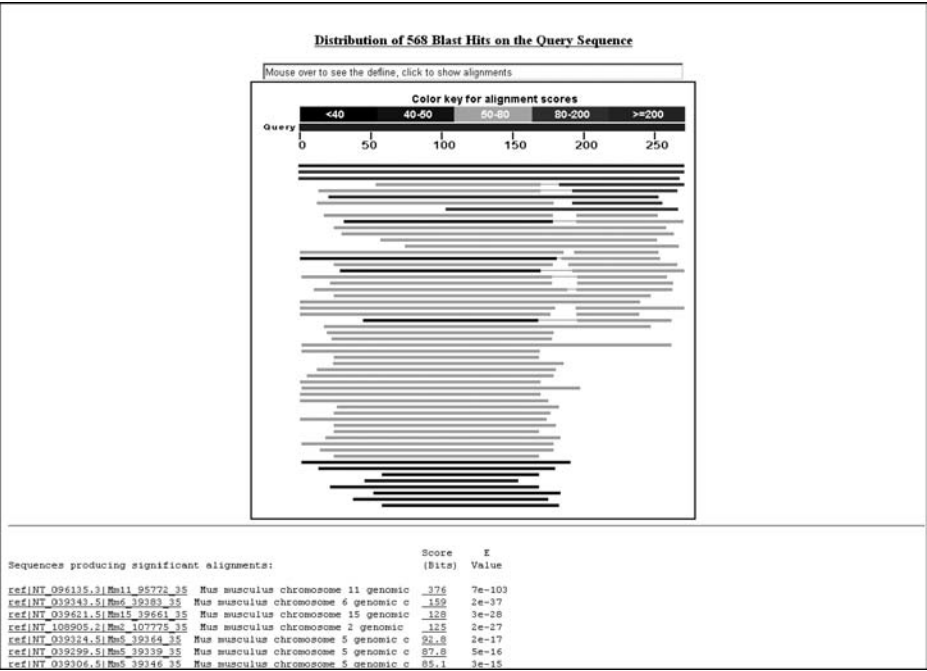


Fig. 8. Genomic tblastn against the mouse genome problem 1: graphical overview. The protein query, NP_032294, on performing tblastn against the mouse genome gets several more hits than the MegaBLAST of the corresponding mRNA against the mouse genome as shown in Fig. 6.

coding exons and one each to the *Hoxb3* and *Hoxd3* genes. Pay attention to the “Refer to Features in this part of subject sequence.” Three of these hits, two to the *Hoxb5* and one to the *Hoxb3* genes, are on the Contig NT_096135.3 placed on chromosome 11. For the second search, paste the protein accession number NP_032294 into the mouse genome search page, select “genome (reference only)” as the database and tblastn as the program. The result should appear similar to that shown in Fig. 8. This search gives several more hits than the earlier MegaBLAST search. Pay attention to the “Refer to Features in this part of subject sequence.” There is a complete hit to the homeobox B5 protein, shown in Fig. 9, and to the homeodomains of the other members of the homeobox B family, seen in Fig. 10 (corresponding to the amino acids 195..253 in the query), such as B6, B4, B3, B2, B13, and so on, on chromosome 11, homeobox



Fig. 9. Genomic tblastn against the mouse genome problem 1: alignment view. The protein query, NP_032294, on performing tblastn against the mouse genome, aligns completely to the two coding exons of the homeobox *B5* gene annotated on the mouse genome contig NT_096135.3 on chromosome 11.

A family members on chromosome 6, and homeobox C family members on chromosome 15 (refer to **Note 8** for the locations of conserved domain).

6.2.2. Problem 2

Click on the link indicated by “H” next to mouse genome BLAST to access the problem. This problem describes how to use mouse genome blast to identify the protocadherin β homologues encoded by the mouse genomic sequence. As described in **Subheading 2.**, tblastn will be useful for identifying unannotated homologues also. Access the mouse genome BLAST page, by clicking on the “mouse” link under Genomic BLAST. Paste the accession number for protocadherin β 1 protein, AAK26059, in the query box, select

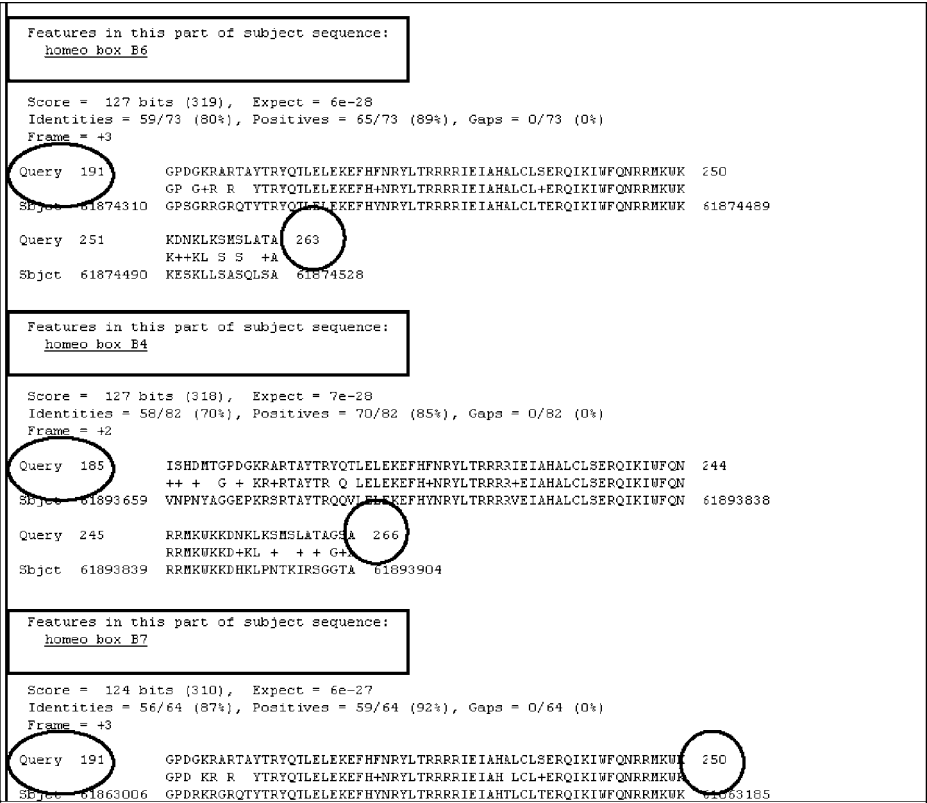


Fig. 10. Genomic tblastn against mouse genome problem 1: alignment view (continued). Some of the hits in the region of the conserved homeodomain (amino acid residues 195..253) of the query protein NP_032294 to other members of the homeobox protein family are displayed. The names of the proteins are highlighted by rectangles and the query amino acid numbers are highlighted by ovals.

“genome(reference only)” as the database and tblastn as the program. The result contains a complete hit to the protocadherin β 1 protein and to other members of the protocadherin β and γ subfamily A on chromosome 18 (see Note 8 for the locations of conserved domains).

7. BLAST2Sequences

BLAST2Sequences is used to compare two sequences, protein or nucleotide, using any one of the principal BLAST variants, “blastp,” “blastn,” “tblastn,” “blastx,” “tblastx,” or MegaBLAST.

7.1. Comparisons Between Two Sequences

The output of BLAST2Sequences consists of a set of the traditional pairwise alignments generated by the principal BLAST programs it uses, supplemented with a dot plot representation of these alignments. The dot plot is useful for highlighting deletions and duplications of segments between two sequences. The translated variants of BLAST2Sequences are useful for the detection of exons.

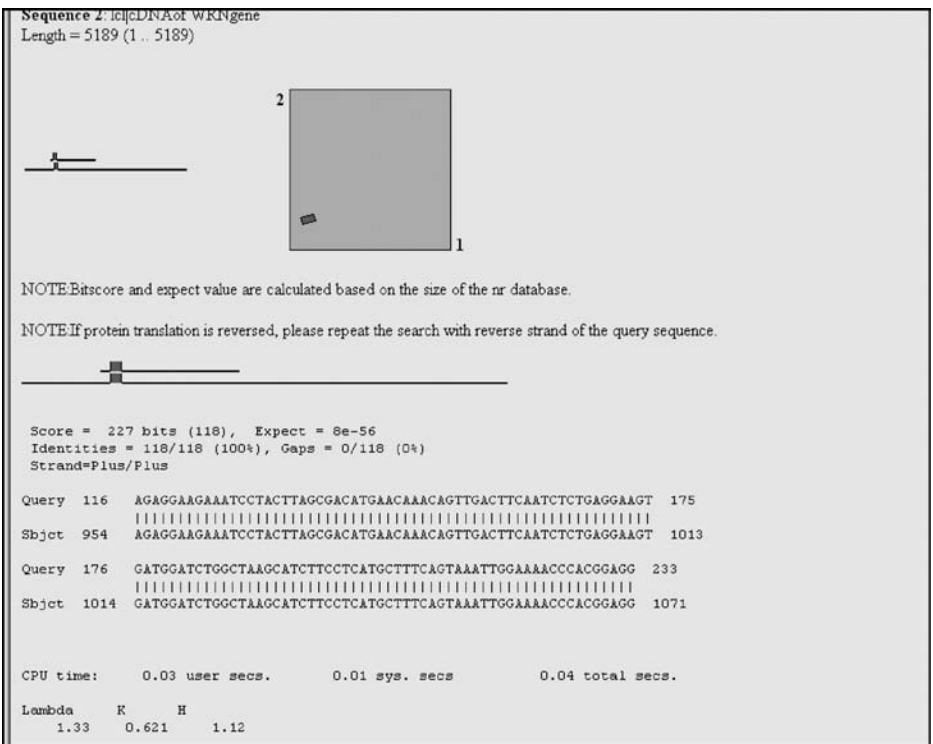


Fig. 11. BLAST2Sequences problem 1 output: detecting an exon on unannotated genomic sequence. The query (genomic sequence) nucleotides 116..233 are similar to “Sbjct” (cDNA sequence) nucleotides 954..1071. Compare the cDNA coordinates to the exon coordinates provided in the problem on the Mini-Course web page. The cDNA coordinates 954..1071 correspond to the exon number 8. Thus, the provided genomic DNA contains exon 8 of the *WRN* gene.

7.2. Examples of Two Sequence Comparisons

7.2.1. Problem 1

Click on the link indicated by “P” next to “Align two sequences (bl2seq).” This problem describes the comparison of two nucleotide sequences. The problem provides a genomic sequence and an mRNA (cDNA) sequence. The genomic sequence is a piece from a GenBank HTG record that contains part of the Werner’s syndrome gene *WRN*. This Gene contains 35 exons. The figure in the problem on the BLAST QuickStart website shows the mapping of exons to the cDNA coordinates. We will use BLAST2Sequences to determine which exon, if any, is contained in the supplied HTG sequence by comparing it against the *WRN* gene cDNA sequence. Access the BLAST2Sequences page by clicking on the link “Align two sequences (bl2seq).” Paste the HTG sequence in the top box under “Sequence1” and the cDNA sequence in the bottom box under

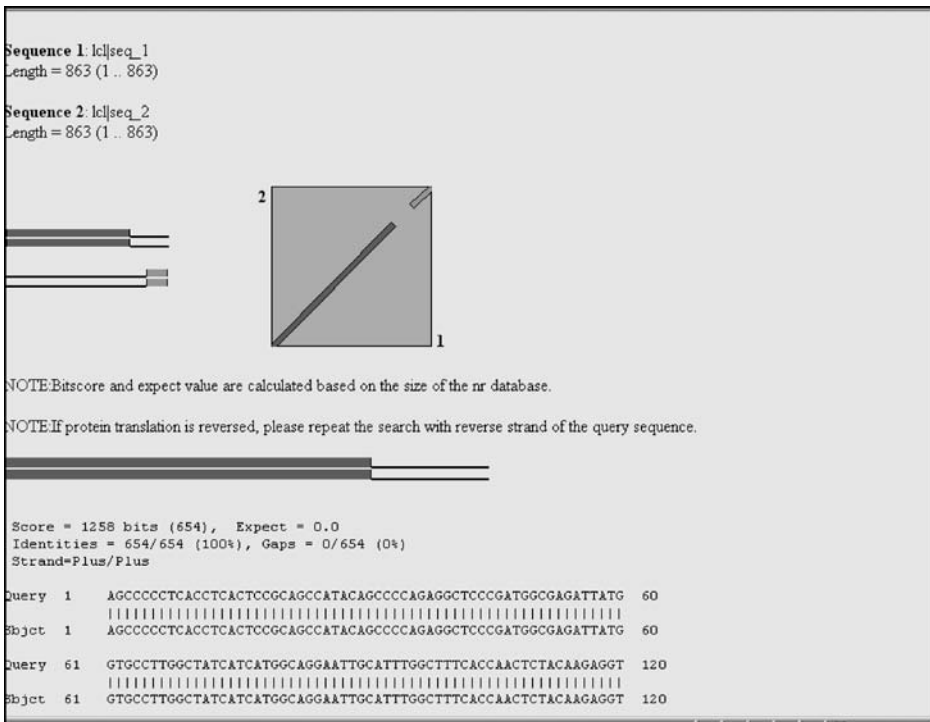


Fig. 12. BLAST2Sequences problem 2 output: the mystery of the missing piece. The alignment of the query sequence to itself is broken into two parts.

“Sequence2.” Click on the “Align” button to obtain the output of Fig. 11. The query (genomic sequence) nucleotides 116..233 are similar to “Sbjct” (cDNA sequence) nucleotides 954..1071. Compare the cDNA coordinates to the exon coordinates provided in the problem. The cDNA coordinates 954..1071 correspond to the exon number 8. Thus, the provided genomic DNA contains exon 8 of the WRN gene.

7.2.2. Problem 2

Click on the link indicated by “H” next to “Align two sequences (bl2seq).” This problem describes the importance of one of the BLAST parameters. The problem gives one DNA sequence. Paste the sequence in both the Sequence 1 and Sequence 2 windows in the BLAST2Sequences page, and click on Align to reach a display similar to that of Fig. 12. Why is the alignment broken into two parts? The sequence between the nucleotides 655..752 is missing in

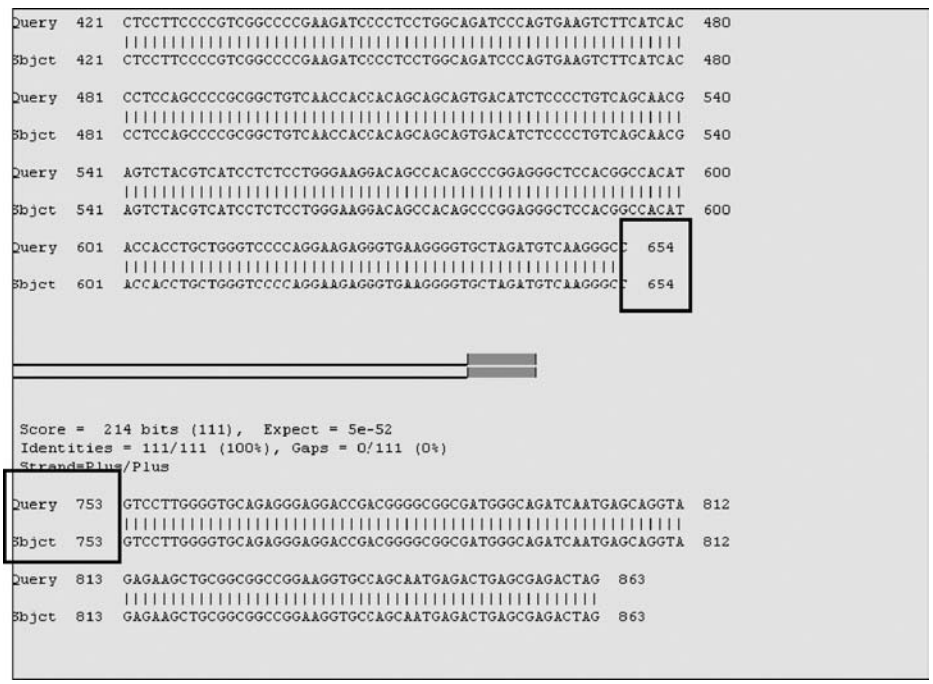


Fig. 13. BLAST2Sequences problem 2: alignment view at the junction. The sequence between the nucleotides 655..752 is missing in the alignment of the sequence to itself. The nucleotides at the junction of the two alignments are highlighted by rectangles.

the alignment of the sequence to itself. This is because of the default Low complexity filter option. Unclick the “Filter” option and perform the search again. Now the query sequence aligns completely to itself (as it should). BLAST masked the nucleotides 655..752, missing in the Alignment View of [Fig. 13](#), as it is a low complexity region (see [Note 4](#)).

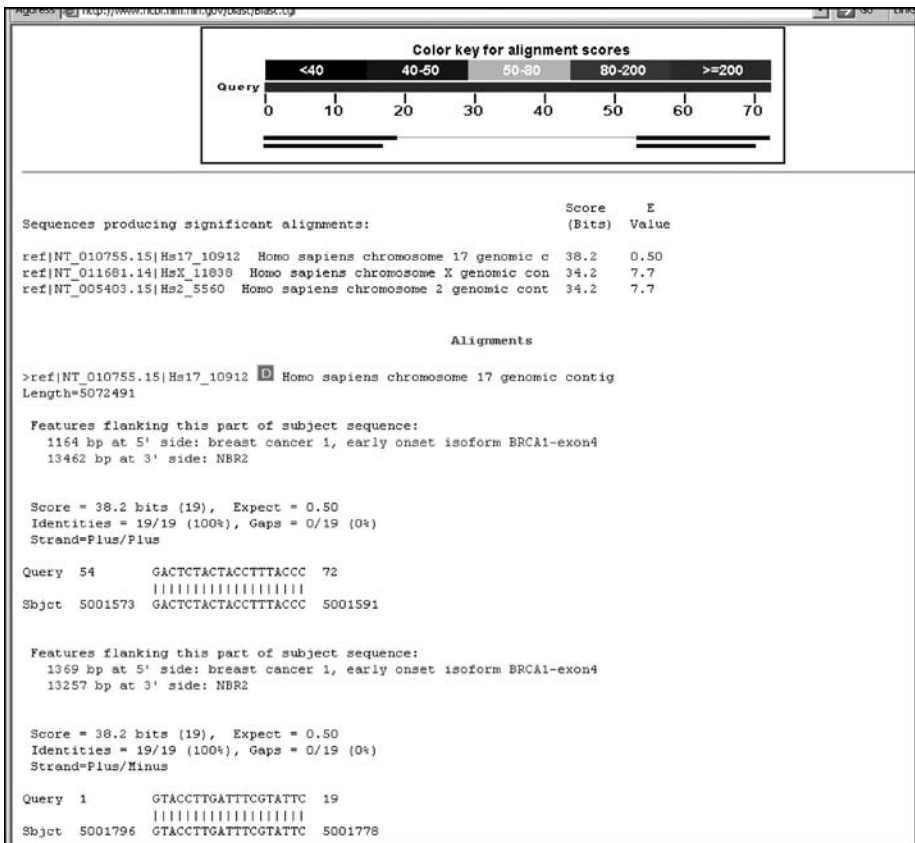


Fig. 14. Graphical overview of primer hits from the nucleotide–nucleotide search problem 1 on the human genome. The primer pair finds only one hit on chromosome 17 aligning to two separate regions shown by thick bars with a gap in between shown by a thin gray line. The parts of the forward and reverse primers also align to two different regions of the genome (as indicated by two separate primers not joined by a thin gray line) on chromosome X and 2.

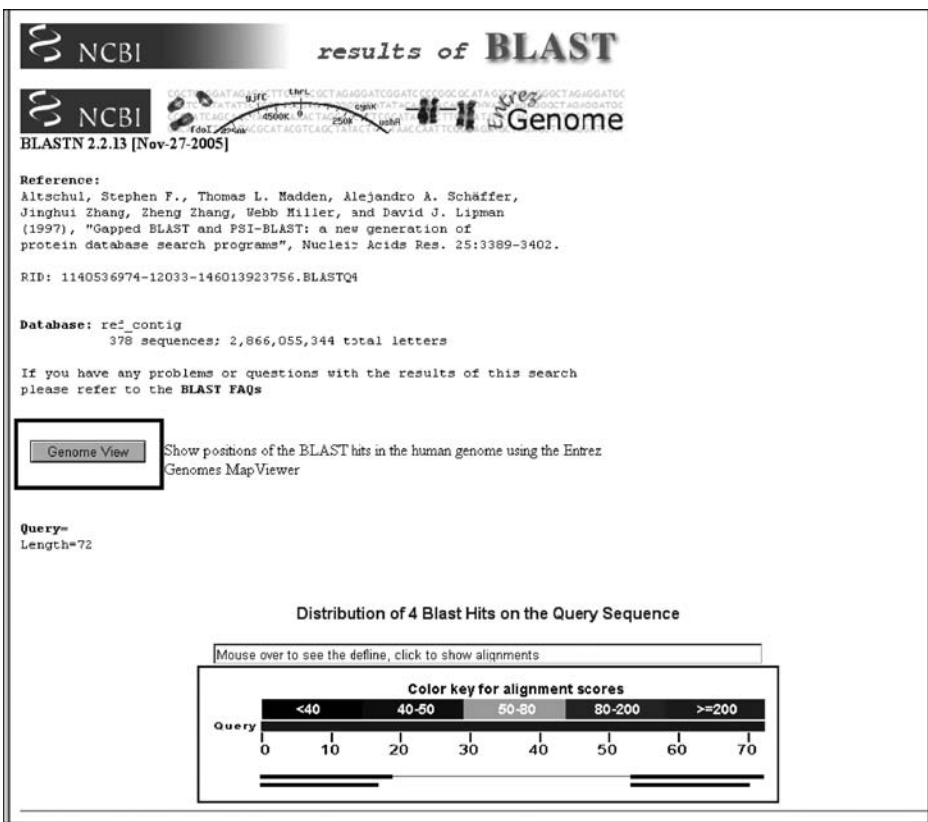


Fig. 15. Graphical overview of primer hits from the nucleotide–nucleotide search problem 1 on the human genome. Press the “Genome View” button highlighted by a rectangle to see hits on the human chromosomes.

8. Notes

1. GenBank and nr. The remaining 12 hits of the primer pair to the database sequences may represent the potential for amplification of different regions of the human genome. Alternatively, the result may stem from the redundant nature of GenBank. The default “nr” database used in this problem includes nucleotide sequences from the International Nucleotide Sequence Database Collaboration, which comprises the DNA DataBank of Japan, the European Molecular Biology Laboratory, and GenBank at NCBI (9,10). It is redundant in nature as each laboratory can submit the nucleotide sequence that they sequenced even if an identical sequence already exists in the database. The nucleotide “nr” BLAST database is not made nonredundant as opposed to the protein “nr” BLAST database,

which is made nonredundant by clustering identical proteins in one group. Thus, further analysis of the annotation on each of the entries is necessary to determine whether the primer pair will amplify a unique region of the human genomic DNA in the *BRCA1* gene. The user may perform the same search against the human genome BLAST database, which is nonredundant (refer to Genomic Blast (<http://www.ncbi.nlm.nih.gov/genome/seq/BlastGen/BlastGen.cgi?taxid=9606>) and **Note 2**). Which parameters will you need to change (*see Note 2*)?

2. Primers and human genome BLAST. If a primer pair is used (with some N's in between) as a query on the human genome BLAST page (NCBI home page-BLAST-Genomes-Human), the user will need to use the *blastn* as the program and increase the *e*-value to, say, 10. Select the "genome (reference only)" as the database. The primer pair finds only one hit on chromosome 17, shown in **Fig. 14**, aligning to two nearby regions (joined by a thin gray line). The parts of the forward and reverse primers also align to two different regions of the genome (as indicated by two separate hits not joined by a thin gray line) on chromosome X and 2. You may view the hit on the human genome by clicking on the Genome View button at the top and accessing the Map Viewer (**Fig. 15**).
3. **Alignment views.** BLAST offers a variety of alignment views. For example, the pairwise option shows an alignment of the query to one database sequence at a time. There are other options such as, Query-anchored with identities, Query-anchored without identities, Flat Query-anchored with identities, and Flat Query-anchored

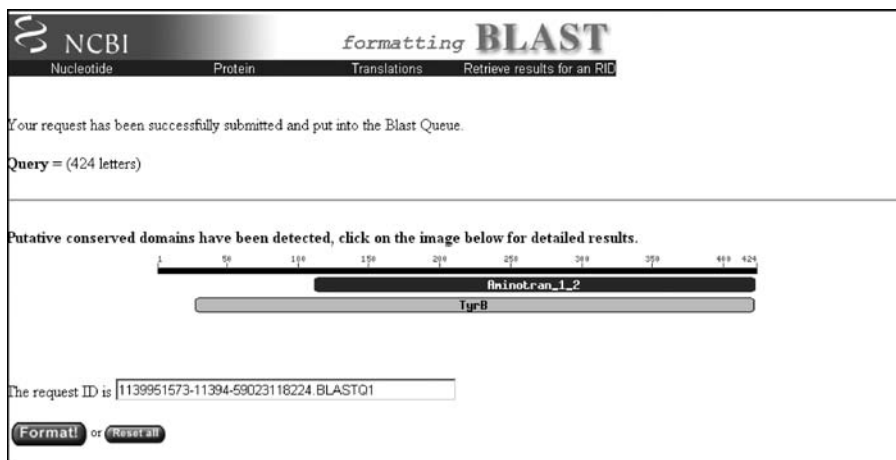


Fig. 16. Automatic conserved domain search: graphical overview. The query protein in the protein-protein BLAST problem 1 contains an amino transferase 1_2 conserved domain indicated by the red bar below the query line.

without identities. These options show the multiple alignment-like view of the query with the database sequences. They differ in the way identities and gaps are displayed. The option, Flat Query-anchored with identities, is useful to identify the conserved regions (indicated by the dots) in the database sequences with respect to the query and the SNPs (indicated by the one letter code).

- 4. Low-complexity sequence. The phrase “low-complexity sequence” refers to stretches of nucleotide or protein sequence that are repetitive or simple in composition (11). Extreme examples include runs of As in a nucleotide sequence such as the poly-A tails of eukaryotic mRNAs, or the poly-proline tracts found in some

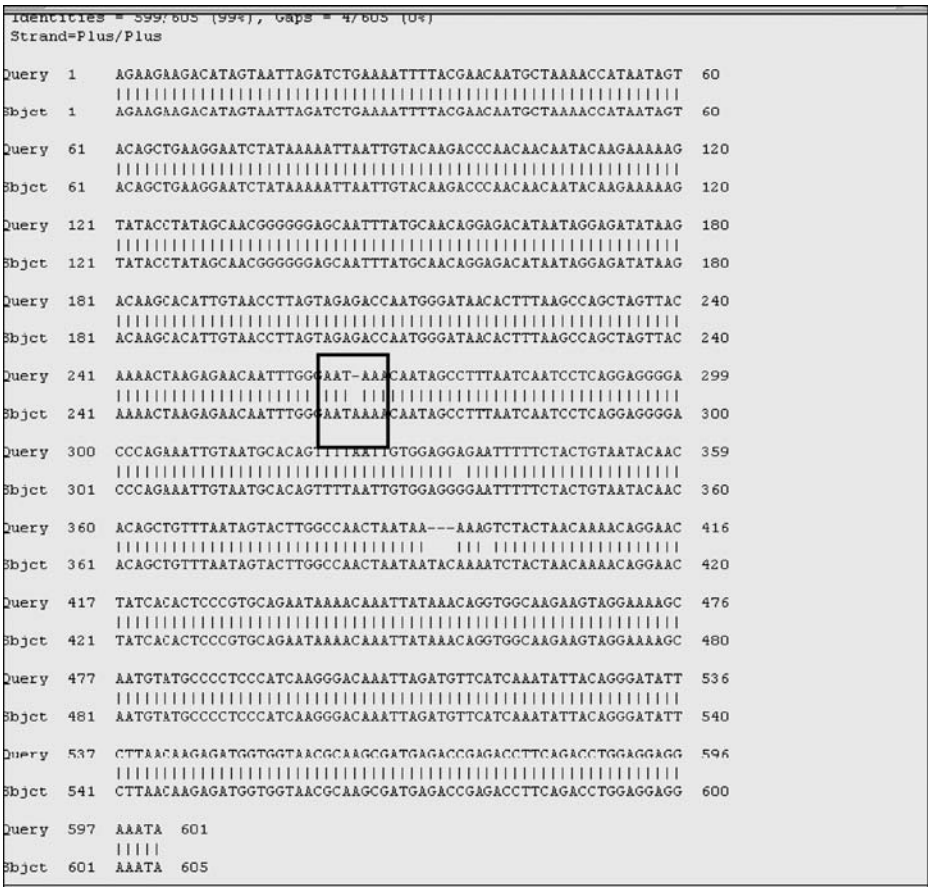


Fig. 17. Identification of a frame shift from BLASTX problem 1. The region of the query nucleotide lacks an “A,” corresponding to the nucleotide 266 in AY077250.1, causing a frame shift that is highlighted by a rectangle.

proteins, but the runs need not be limited to repeats of a single base or amino acid. BLAST detects and filters these runs in the “query” by default because they often lead to false starts when BLAST initiates alignments from word hits; beginning an alignment in the poly-a tail of an mRNA is not very likely to lead to a meaningful alignment between related mRNA sequences. This filtering can be turned off on the web using a checkbox, however, the resulting searches will take much longer because BLAST will have to process a great number of false starts. The results returned may also include a larger than usual number of questionable

Query	1	ATGAGAGTGAAGGAGAAAATATCAGCACTTGTGGAGATGGGGCACCATGCTCCTTGGGTTG	60
Sbjct	1	ATGAGAGTGAAGGAGAAAATATCAGCACTTGTGGAGATGGGGCACCATGCTCCTTGGGTTG	60
Query	61	TTGATGATCCGTAGTGTCTGCAGACCAATTGTGGGTCACAGTCTATTATGGGGTACCTGTG	120
Sbjct	61	TTGATGATCTGTAGTGTCTGCAGACCAATTGTGGGTCACAGTCTATTATGGGGTACCTGTG	120
Query	121	TGGAAGAAGCAACCACCCTCCATTTTGTGCATCAGATGCTAAAGCATATGATACAGAG	180
Sbjct	121	TGGAAGAAGCAACCACCCTCTATTTTGTGCATCAGATGCTAAAGCATATGATACAGAG	180
Query	181	GTACATAATGTTTGGGCCACACACGCCTGTGTACCCACAGACCCCAACCCACAAGAAGTA	240
Sbjct	181	GTACATAATGTTTGGGCCACACATGCCTGTGTACCCACAGACCCCTAACCCACAAGAAGTA	240
Query	241	GTATTGGCAAAATGTGGCAGAAAATTTTAAATGTGGGATAATAACATGGTAGAAGCAGATG	300
Sbjct	241	GTATTGGCAAAATGTGACAGAAAATTTTAAATGTGGGATAATGACATGGTAGAAGCAGATG	300
Query	301	CATGAGGATATAATCAGTTTATGGGATCAAAGCCTAAAGCCATGTGTTAAATTAACCCCA	360
Sbjct	301	CATGAGGATATAATCAGTTTATGGGATCAAAGCCTAAAGCCATGTGTTAAATTAACCCCA	360
Query	361	CTCTGTGTTACTTTAACTGCACTGATAAGATTAATAATACCAAACTACTCCTAATAAT	420
Sbjct	361	CTCTGTGTTACTTTAACTGCACTGATGAGGTTAATAATACCAAACTACTCCTAATAAT	420
Query	421	ACCACTACTACTCCCTTACCCTGTTACTCTACTAGTAATAGCAGCATGACAGAGAGAA	480
Sbjct	421	ACTCAAACCTAC---CACTACCCTGTTACTCTACTAGTA---GCAGCATGACAGAGAGAA	474
Query	481	GGAGAAATAAAAACTGCTCTTTCAATATCACCACAGCCATAAGAGATAAGGTGCAGAGA	540
Sbjct	475	-----ATAAAAACTGCTCTTTCAATATCACCACAGCCATAAGAGATAAGGTGCAGAAA	528
Query	541	GGATATGCATTGTTTAGTAACCTTTGATATAGTACCAATAGATAATGATAGAAATGATAG	600
Sbjct	529	GGCTATGCATTGTTTAAATAAAGCTT-GATATAGTACCAATAGATAATGATAGGAATGATAG	587
Query	601	TACCAGCTATAGGTTGCTAAGTTGTAACACCTCAGTCATTACACAAGCGTGTCCAAAGGT	660
Sbjct	588	TACCAGCTATAGGTTGCTAAGTTGTAACACCTCAGTCATTACACAGGCGTGTCCAAAGGT	647

Fig. 18. Identification of a frame shift from BLASTX problem 2. The region of the query sequence containing an extra “T,” compared to AF482979.1, at position 565 is highlighted by a rectangle.

- alignments. Nucleotide sequences are filtered using a program called Dust (12); protein sequences are filtered with SEG (13).
- Automatic CDD search. When a protein–protein BLAST search is ran, the query protein sequence is also searched against the conserved domains database. The presence of a conserved domain in the protein is reported on the page with the request ID before you format the page. For example, for the blastp problem 1, the query protein contains an amino transferase 1_2 conserved domain indicated by the red bar below the query line seen in Fig. 16. Click on the red bar to access the conserved domain database and determine the amino acid positions of the domain.
 - Single nucleotide differences in blastx Problem 1. To discover the nucleotide difference in the BLASTX Problem 1, we will compare the query nucleotide sequence to the nucleotide sequence on which the protein AAL71647.1 is

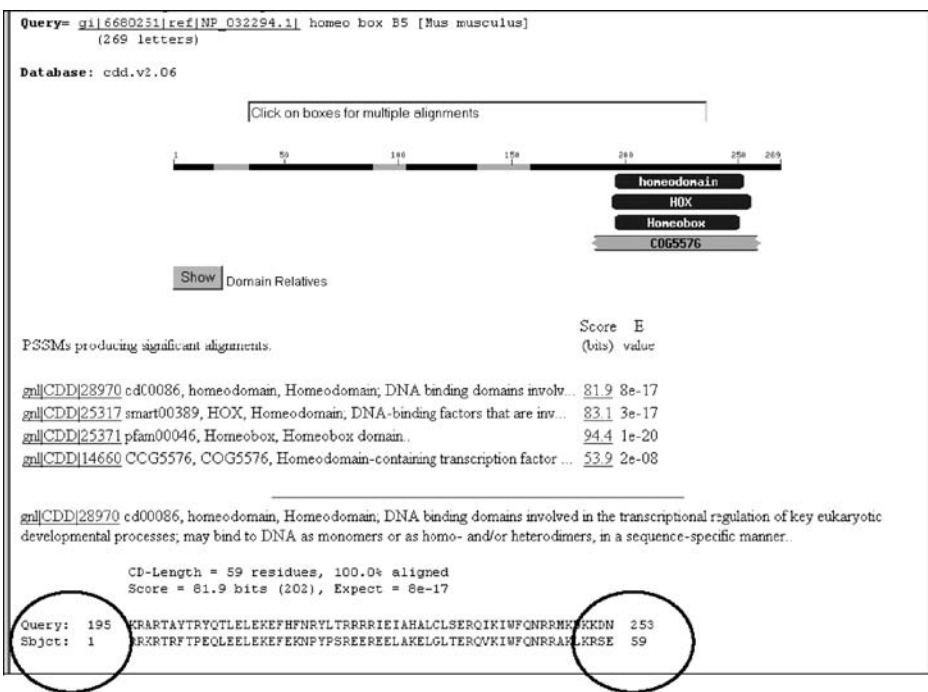


Fig. 19. Conserved domain search results. The query protein in the Genome BLAST Problem 1, NP_032294, contains a homeodomain between amino acid 195..253, highlighted by ovals. Perform this search for the protein accession number NP_032294 from the Genomes Problem 1 to reach the view shown in this figure. The query protein contains a homeodomain between amino acids 195..253.

annotated. Click on the accession number AAL71647.1. The protein is annotated on the nucleotide entry AY077250.1 as shown in “DBSOURCE.” From the BLAST mini-course page, click on the link “Align two sequences (bl2seq)” in the panel labeled “Special.” Paste the query nucleotide sequence from the problem in the box for Sequence 1 and the accession number, AY077250, in the second box. Unclick the filter box (*see Note 4*) and click the “Align” button to create the output shown in **Fig. 17**. The query nucleotide lacks an “A” corresponding to the nucleotide 266 in AY077250.1 causing a frame shift. There are other differences between the two nucleotide sequences (such as a nucleotide substitution or deletion of three nucleotides), which do not cause a frame shift.

7. Single nucleotide differences in blastx Problem 2. To discover the nucleotide difference in the blastx problem 2, click on the accession number AAL91985.1. The protein is annotated on the nucleotide entry AF482979.1 as shown in “DBSOURCE.” From the BLAST mini-course page, click on the link “Align two sequences (bl2seq)” in the panel labeled “Special.” Paste the query nucleotide sequence from the problem in the box for Sequence 1 and the accession number, AF482979, in the second box. Unclick the filter box and click on the “Align” button to produce the alignment of **Fig. 18**. The query nucleotide sequence contains an extra “T” at nucleotide 565.
8. Manual CDD search. A protein query can be also manually searched against the conserved domain database. The option is provided under the Protein panel at the “Search the conserved domain database (rpsblast)” link. Perform this search for the protein accession number NP_032294 from the Genomes problem 1 (**Fig. 19**).

References

- 1 Madden, T. L. and McGinnis, S. (2004) Blast: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic Acids Res.* **32**, W20–W25.
- 2 Henikoff, S. and Henikoff, J. G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* **89**, 10,915–10,919.
- 3 (1978) Atlas of Protein Sequence and Structure, chapter Matrices for detecting distant relationships. *Natl. Biomed. Res. Found.* Washington, DC.
- 4 Altschul, S. F. and Gish, W. (1996) Local alignment statistics. *Methods Enzymol.* **266**, 460–480.
- 5 Madden, T. L., Schaffer, A. A., Zhang, J., et al. (1997) Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 6 Madden, T. The blast sequence analysis tool, in *The NCBI Handbook*.
- 7 Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. (2000) A greedy algorithm for aligning dna sequences. *J. Comput. Biol.* **7**, 203–214.
- 8 Hermankova, M., Ray, S. C., Ruff, C., et al. (2001) Hiv- 1 drug resistance profiles in children and adults with viral load of <50 copies/ml receiving combination therapy. *JAMA* **286**, 196–207.

- 9 Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. (2006) Genbank. *Nucleic Acids Res.* **34**, 16–20.
- 10 Mizrachi, I. Genbank, in *The NCBI Handbook*.
- 11 Wootton, J. C. and Federhen, S. (1996) Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.* **266**, 554–571 <http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=handbook.chapter.ch16>.
- 12 Tatusov, R. and Lipman, D.J., Dust. Unpublished data.
- 13 Wootton, J. C. and Federhen, S. (1993) Statistics of local complexity in amino acid sequences and sequence databases. *Computers and Chemistry*, Elsevier Science, Amsterdam, The Netherlands.

PSI-BLAST Tutorial

Medha Bhagwat and L. Aravind

Summary

PSI-BLAST (Position-Specific Iterative Basic Local Alignment Search Tool) derives a position-specific scoring matrix (PSSM) or profile from the multiple sequence alignment of sequences detected above a given score threshold using protein–protein BLAST. This PSSM is used to further search the database for new matches, and is updated for subsequent iterations with these newly detected sequences. Thus, PSI-BLAST provides a means of detecting distant relationships between proteins. In this chapter, we discuss practical aspects of using PSI-BLAST and provide a tutorial on how to uncover distant relationships between proteins and use them to reach biologically meaningful conclusions.

Key Words: PSI-BLAST; BLAST; distant sequence similarity; PSSM; profile; structural relationships.

1. Introduction

BLAST (Basic Local Alignment Search Tool) is a sequence similarity search method, in which a query protein or nucleotide sequence is compared to nucleotide or protein sequences in a target database to identify regions of local alignment and report those alignments that score above a given score threshold ([\[1\]](#); and Chapter 9). Position-Specific Iterative (PSI)-BLAST is a protein sequence profile search method that builds off the alignments generated by a run of the BLASTp program. The first iteration of a PSI-BLAST search is identical to a run of BLASTp program ([\[1\]](#)). It then generates a multiple alignment of the highest scoring pairs of the BLASTp run above a certain preset score or *e*-value threshold and calculates a profile or a position-specific score matrix

(PSSM) from the multiple alignment. The PSSM captures the conservation pattern in alignment and stores it as a matrix of scores for each position in the alignment—highly conserved positions receive high scores and weakly conserved positions receive scores near zero. This profile is used in place of the original substitution matrix for a further search of the database to detect sequences that match the conservation pattern specified by the PSSM. The newly detected sequences from this second round of the search, which are above the specified score (*e*-value) threshold are again added to alignment the profile is refined for another round of searching. This process is iteratively continued until desired or until convergence, i.e., the state where no new sequences are detected above the defined threshold. The iterative profile generation process makes PSI-BLAST far more capable of detecting distant sequence similarities than a single query alone in BLASTp, because it combines the underlying conservation information from a range of related sequence into a single score matrix. In the evolution, three-dimensional (3D) structures of proteins may be conserved even after considerable erosion of their sequence similarity. PSI-BLAST has been demonstrated to be useful in detecting such relationships via sequence searches, which were previously only detected through direct comparison of the 3D structures (1,2). In this chapter, we discuss practical aspects of using PSI-BLAST and provide a tutorial on how to uncover distant relationships between proteins and use them to reach biological meaningful conclusions.

PSI-BLAST is most conveniently used on the internet with the help of the graphical user interface provided by the PSI-BLAST search page on National Center for Biotechnology Information (NCBI) website (<http://www.ncbi.nlm.nih.gov/BLAST/>). The PSI-BLAST page may be customized by the user in terms of automated or semiautomated or “two-page formatting” and other parameters modified as desired. This page can then be saved as permanent internet bookmark for repeated use on future occasions. As a rule of the thumb, beginners are advised to use the profile-inclusion threshold of expect (*e*)-value = 0.005 for their analysis (see **Note 1**). However, a user familiar with globular domains and compositional bias may use the inclusion threshold of 0.01 for inclusion in the profile, if a sequence does not have any major compositionally biased segments (see **Subheading 4** and **ref. 3** for further details on compositional bias). A pair of protein sequences can either be homologous (sharing a common evolutionary ancestor) or nonhomologous (evolutionarily unrelated). It should be noted that PSI-BLAST does not offer a direct binary decision on whether two sequences are related or not. However, the *e*-value obtained for a PSI-BLAST alignment can be used as a guide for this purpose. As a heuristic it may be assumed that any compositionally unbiased query, encompassing a globular domain in a protein, giving a hit with

e -value = <0.01 is likely to be an indication of a homologous relationship. However, a user must carefully evaluate such alignments case-by-case because there can occasionally be false-positives. A user may set the number of alignments and hits view as at least 1000 if searching the nonredundant (nr) database of NCBI, because of the large number hits obtained due to the current size of the database. PSI-BLAST may also be downloaded and run as a standalone program for Windows or UNIX-type operating systems. However, in this case the various parameters need to be specified using the set of command-line flags for the program. An advantage of using the standalone version is the ability to use alignments as queries to generate a starting PSSM, or saving and reusing the profile generated by a run of PSI-BLAST.

In the tutorial below the first example demonstrates how the structural and functional similarities between the *Escherichia coli* DNA polymerase III β -subunit and eukaryotic proliferating cell nuclear antigen (PCNA) can be identified and investigated using the PSI-BLAST program. In the next example, we demonstrate the strength of PSI-BLAST in exploring the function of an uncharacterized protein by means of identifying its 3D structural template. Emphasis here is chiefly on the practical steps involved, although when required, some of the relevant theoretical background is also provided.

2. Problem 1

2.1. Background

Cellular DNA polymerase enzymes tend to dissociate from DNA after adding a few nucleotides and require an accessory factor to tether them to DNA while elongating the growing DNA chain (4). In eukaryotes and archaea, this function is performed by the protein called PCNA, whereas in prokaryotes such as *E. coli* the same function is performed by the β -subunit of DNA polymerase encoded by the *dnaN* gene. When the crystal structure of *E. coli* DNA polymerase III β -subunit was solved, it was found to be ring shaped [5]; Fig. 1). The β -subunit forms a ring around the DNA and holds the polymerase on DNA, hence, it is also called β -clamp. It was predicted that PCNA proteins will also possess a similar ring-shaped structure (5). The crystal structure of PCNA confirms that it is similar to that of *E. coli* β -subunit [6,7]; Fig. 1). They all appear to have a sixfold symmetry; however, *E. coli* DNA Pol-III β -subunit is a dimer, whereas PCNA is a trimer. Each monomer of the *E. coli* protein contains three homologous domains, whereas each monomer of PCNA proteins consists of two homologous domains (Fig. 1). Each domain contains an identical fold consisting of two α -helices and eight β -sheets (nine in PCNA). The proteins are negatively charged, however, two α -helices are positively charged apparently nonspecifically clamp around DNA. The sequences of β -subunit proteins are

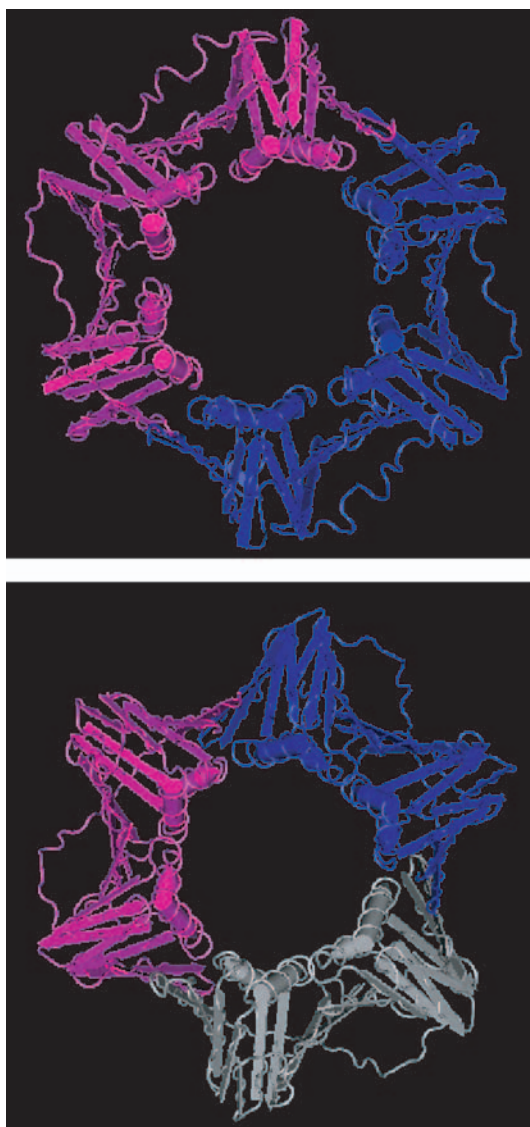


Fig. 1. The crystal structures of *Escherichia coli* DNA polymerase III β -subunit and human proliferating cell nuclear antigen (PCNA): The crystal structure of *E. coli* DNA polymerase III β -subunit (PDB accession number 2POL) is on the top and that of human PCNA (PDB accession number 1AXC) is at the bottom. Each monomer in the structures is identified by a different color. The entry 1AXC contains the structure of human PCNA complexed with C-terminal region of p21(WAF1/CIP1). The structure of only human PCNA without the p21 protein is displayed here.

well-conserved in prokaryotes and that of PCNA in eukaryotes, but despite performing similar functions, and having some sequence conservation based on their 3D structure (5), the conventional BLASTp program detects no sequence similarity between these proteins. This distant sequence similarity, however, can be detected by PSI-BLAST as demonstrated in Problem 1. When we use human PCNA (accession number NP_002583) as the query and nr as the database, *E. coli* β -subunit is retrieved in the fifth iteration.

2.2. Practical Steps

1. Access PSI-BLAST from the BLAST page <http://www.ncbi.nlm.nih.gov/BLAST/>.
2. Paste the accession number NP_002583 or gi number 4505641 in the query box (see Note 2).
3. Use the default parameters (except the number of alignments and descriptions) such as “nr” as the database (see Note 3), *e*-value 10 and the statistical significance threshold to include a sequence for generating the PSSM for the next iteration as 0.005. Change the maximum number of alignments and descriptions, 1000, from the respective pull down menus to retrieve possibly all statistically significant hits.
4. Format to get the results. The results are retrieved in another web page. The hits are divided into two sections. The hits with better statistical significance than the *e*-value threshold, 0.005, are listed first. Those with *e*-values worse than threshold, but have an *e*-value better than that selected on the query page, 10, are listed further down the page. Hits with *e*-values better than the threshold are used in forming the profile that will be used in subsequent PSI-BLAST iterations (see Note 4). It will be observed that most of the hits are to eukaryotic PCNA.
5. Click on the “taxonomy reports” to get a list of hits and their organisms. These are mostly eukaryotes and archaea.
6. Click the “Run 2nd iteration” button and then the “Format” link on the BLAST page (see Note 5).
7. Repeat steps 5 and 6 until the desired results or convergence (see Notes 6 and 7). *E. coli* DNA polymerase III β -subunit protein encoded by the *dnaN* gene is retrieved in the fifth iteration and the sequence alignment that is obtained is shown in Fig. 2.

An examination of the sequence alignment of PCNA and the DNA pol-III β -subunit reveals a conservation pattern that is chiefly comprised of two types of residues: (1) The hydrophobic residues that are distributed throughout the length of the alignment and (2) polar (principally charged) residues distributed sporadically in the alignment. A comparison of the conserved positions using the structures of PCNA and the β -subunit as a guide illustrate that the conserved hydrophobic positions are those that are required for forming the hydrophobic core that folds into the interior of the protein domain and stabilizes via hydrophobic interaction. Thus, they are the critical determinants of the common fold assumed by PCNA

```

Query   5      RLVQGSILKKVLEALKDLINEACWDISSSGVNLQSMDSHVSQLTLRSEGFDTYRCDR 64
          RL++ +      + ++ +N ++      + + D +++ + +
Sbjct  137    RLIEATQFSMAHQDVRYYLNGMLFETEGEELRTVATDGHRLAVCSMPIGQSL-----P 189

Query   65      NLAMGVNLTSMKILKCAGNEDIITLRAEDNADTLALVFEAPNQEKVSD---YEMKLMDL 121
          ++ V  + ++++      + + L + + N      + KL+D
Sbjct  190    SHSVIVPRKGVIELMRML-----DGGDNPLRVQIGSNNIRAHVGDFIFTSKLVDG 239

Query  122     DVEQL-GIPEQEYSCVVKMPSGEFARICRDLSHIGDA---VVISCAKDGVKFSASGELG 176
          + +      ++      + + + +      V + +++ +K +A+
Sbjct  240    RFPDYRRVLPKNPDKHLEAGCDLLKQAFARAAILSNEKFRGVRLYVSENQLKITANNPEQ 299

Query  177     NGNIKLSQTSNVDKEEEAVTIEMNEPVQLTFALRYLNFFTKATPLSSTVTLSMSADVPLV 236
          + EE      + + + F + Y+      A      V + + D
Sbjct  300    E-----EAEEILDVTYSGAEMEIGFNVSYVLDVLNALK-CENVRMML-TDSVSS 346

Query  237     VEYKIADMGHLKYYLAP 253
          V+ + A      Y + P
Sbjct  347    VQIEDAASQSAAYVMP 363

```

Fig. 2. Alignment of human proliferating cell nuclear antigen (PCNA) and *Escherichia coli* DNA polymerase III β -subunit. The sequence alignment obtained in the fifth iteration of Position Specific Iterative Basic Local Alignment Search Tool using query human PCNA protein NP_002583.1 to the “sbjct” *E. coli* DNA polymerase III β -subunit NP_418156.1 is displayed.

and the β -subunit. Likewise it is seen, that the polar residues localize to the solvent or ligand-exposed surfaces of the molecule. In particular, the positively charged positions localize to the interiors of the ring structure that interact with DNA, whereas the negatively charged positions localize to the exterior surface of the ring. Thus, despite the vast sequence divergence seen between PCNA and the DNA pol-III β -subunit, we observe that not only is their relationship detected using PSI-BLAST, but also subtle patterns are picked up in the alignment that have relevance for the shared folding and functional properties of these proteins.

3. Problem 2

In this problem, we will demonstrate the strength of PSI-BLAST to assign function to an uncharacterized protein and obtain its structural template. Retrieve an entry with accession number, BAE56987, in the Entrez Protein database (<http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Protein>). The *Aspergillus oryzae* protein is currently identified as an unnamed protein. Perform the PSI-BLAST search as described in **Subheading 2**. There are a number of hits to other unnamed or hypothetical proteins and a couple of histone acetyl transferases. The “G” buttons next to the hits link to the Entrez Gene report for the genes (*see Note 2*). The third iteration results include a number of hits to histone acetyl transferases. One of them is a hit to an experimentally

determined crystal structure, 1VHS ($e \sim 10^{-3}$), from the Protein Data Bank (8,9), and this is indicated in the search results by a red “S” button next to it. The sequence alignment of the query protein to 1VHS is shown in Fig. 3.

Examination of the alignment generated by the PSI-BLAST with a protein of known structure can now be used to explore the structural and biochemical properties of the uncharacterized protein. One can visualize the 3D structure of 1VHS in the aligned region and, thus, obtain a structural template for exploring the query protein using 3D structure visualization programs. For example Cn3D, a helper application for the web browsers provided by NCBI can be used directly for this purpose (<http://www.ncbi.nlm.nih.gov/Structure/CN3D/cn3d.shtml>). Click the “S” button next to the 1VHS description, then on the arrow indicating the alignment. First, by using the alignment in conjunction with the known structure (1VHS), one can show that the conserved region detected in the uncharacterized protein is likely to adopt an $\alpha + \beta$ -fold similar to 1VHS. This is not only supported by the statistically significant e -value for the relationship, but also by the conservation of several hydrophobic residues (as in Problem 1) that are likely to assume a key role in stabilizing a hydrophobic core congruent to what is observed in the structure 1VHS. Second, it may also be noticed that the proteins detected in the previous search share a key Q/RxxGxG/A motif that is found in a loop between a strand and helix. This motif is found in a large number of proteins of the GNAT supefamily and required for binding coenzyme A (10). The conservation between the uncharacterized *Aspergillus* protein and 1VHS of the above motif and the positions associated with active site required to bind CoA and transfer it to amino groups suggests that the

```

Query  53  LSYLSQQLNSEIEKGDYAMIDPIPYDEFRHYWFSHFGA---IMLLGDIKNTQDVKLMDR  109
          L  +   NS I      A  +P+  ++ R  WFS          + +  D      +
Sbjct  13  LEAVVAIYNSTIASRXVTADTEPVTPE-D-RXEWFSGHTESRPLYVAEDENGNAVAWISFE  71

Query  110 TGGANWSKLCLGSFTVRPNYPGRSSHICNSMFLVTDASRNRGVGRLMGEGYLEWAPKLV  169
          +F  RP Y      +          +A R +GVG  + +  L  AP L
Sbjct  72  -----TFYGRPAY-----NKTAEVSIYIDEACRGKGVGSYLLQEALRIAPNLGI  115

Query  170  TN  171
          +
Sbjct  116  RS  117

```

Fig. 3. Sequence alignment of an uncharacterized protein to phosphinothricin N-acetyltransferase. The sequence alignment obtained in the third iteration of Position Specific Iterative Basic Local Alignment Search Tool of the query uncharacterized protein BAE56987.1 to the “sbjct” phosphinothricin N-acetyltransferase 1VHS is displayed. The Q/RxxGxG/A motif found in a large number of coenzyme A-binding proteins is highlighted by a box.

former protein is likely to function as a CoA-dependent amino-group acetyl-transferase enzyme (11,12). Similar analysis using PSI-BLAST has been used to assign functions to a number of uncharacterized proteins including yeast SPT10 as a histone acetyl transferase (10).

4. Caveats to Remember While Using PSI-BLAST

There are several key caveats that need to be kept in mind while using PSI-BLAST for obtaining scientifically correct results. The first of these is the effect of compositional bias in the query sequence. Compositional bias is defined as the presence of low entropy, or low information content in a protein sequence. Typically, such a sequence may be marked by enrichment of the sequence in particular amino acids, homopolymeric stretches of a particular amino acid or presence of short-range repetitive structures such as coiled-coils or short β -helices. Such sequences as a rule assume nonglobular structures and can artificially result in high-scoring alignments with other similarly biased sequences in the database. Such relationships are typically neither biologically nor evolutionarily significant and can often mask true relationships, by preventing detection of a more subtle globular domain. For this purpose, the internet version of PSI-BLAST contains certain corrective measures: (1) filtering out of the low-complexity using the SEG program (see **Note 8**) and (2) using composition-based statistical correction in PSI-BLAST (13). These options are available in pull-down menu and by default composition based correction is kept on. Users are strongly advised to use these measures especially if they are searching with sequences of certain eukaryotic organisms, such as *Plasmodium* or *Dictyostelium*, whose proteins are particularly enriched in low-complexity sequence. Another caveat to keep in mind is that different queries belonging to same family or superfamily of proteins can perform differently in searches against the same database in terms of retrieving other members of that family or superfamily. Hence, it is advised that a user run PSI-BLAST from different starting points and compare the hits generated in the different searches. This acts as both a consistency check, which might help in weeding out systematic false-positives generated by a certain query and at the same time widening the horizons of newly detected sequences.

5. Notes

1. The e-value is a parameter that describes the number of hits one can “expect” to see by chance when searching a database of a particular size. It decreases exponentially with the score (S) that is assigned to a match between two sequences. Essentially,

the *e*-value describes the random background noise that exists for matches between sequences.

2. The user can search for the human PCNA gene entry in Entrez Gene (<http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene>) by using the query PCNA AND human (orgn). Entrez Gene is an NCBI resource that provides detailed information about the genes from a number of organisms and links to appropriate resources within and outside NCBI (14). It provides links to the Reference Sequence (RefSeq) entries, when available. The RefSeq database provides nonredundant and curated genomic, transcript and protein sequences for major research organisms (15).
3. The protein nr database consists of conceptual translations of the coding regions annotated on GenBank/EMBL/DDBJ database and protein sequences from databases such as SwissProt and Protein Data Bank. Information about other possible databases can be obtained from <http://www.ncbi.nlm.nih.gov/blast/producttable.shtml#db>.
4. The sequences listed on the page but with *e*-values worse than the threshold 0.005 can be manually selected, by checking the box, for generating the profile for next iteration. Also, the sequences already included by default for generating a profile can be manually removed by unchecking the box next to it.
5. As mentioned in **Note 4**, the BLAST results are retrieved in another page. When clicked on the “Run nth iteration” button, the PSSM generated from the previous BLAST results is searched against the database, the original BLAST search page is refreshed with the new search, and a new request id is assigned. Thus, during several iterations of PSI-BLAST, there will be only two pages, the BLAST search page and the results page. The newly added sequences that were below the threshold in the previous search are indicated as “new” and the green dots indicate the sequences that were identified in the previous iterations.
6. A stand-alone version of PSI-BLAST (<ftp://ncbi.nlm.nih.gov/blast/executables/>) allows the user to run the program for a chosen number of iterations or until convergence.
7. The results can be formatted to obtain PSSM after any iteration, instead of the default pairwise alignment, using the “Alignment” pull down menu next to the “Format” option.
8. The masking of low-complexity by the SEG filter will introduce X in place of the low complexity region or depict them in lower case depending on the user’s choice.

References

- 1 Altschul, S. F., Madden, T. L., Schäffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 2 Aravind, L. and Koonin, E. V. (1999) Gleaning non-trivial structural, functional and evolutionary information about proteins by iterative database searches. *J. Mol. Biol.* **287**, 1023–1040.

- 3 Wootton, J. C. (1994) Non-globular domains in protein sequences: automated segmentation using complexity measures. *Comput Chem.* **18**, 269–285.
- 4 Kornberg, A. and Baker, T. A. (ed.) (1991) *DNA Replication*. W. H. Freeman, New York, NY.
- 5 Kong, X. P., Onrust, R., O'Donnell, M., and Kuriyan, J. (1992) Three-dimensional structure of the beta subunit of E. coli DNA polymerase III holoenzyme: a sliding DNA clamp. *Cell* **69**, 425–437.
- 6 Gulbis, J. M., Kelman, Z., Hurwitz, J., O'Donnell, M., and Kuriyan, J. (1996) Structure of the C-terminal region of p21(WAF1/CIP1) complexed with human PCNA. *Cell* **87**, 297–306.
- 7 Moarefi, I., Jeruzalmi, D., Turner, J., O'Donnell, M., and Kuriyan, J. (2000) Crystal structure of the DNA polymerase processivity factor of T4 bacteriophage. *J. Mol. Biol.* **296**, 1215–1223.
- 8 Berman, H. M., Westbrook, J., Feng, Z., et al. (2000) The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242.
- 9 Marchler-Bauer, A., Address, K. J., Chappey, C., et al. (1999) MMDB: Entrez's 3D structure database. *Nucleic Acids Res.* **27**, 240–243.
- 10 Neuwald, A. F. and Landsman, D. (1997) GCN5-related histone *N*-acetyltransferases belong to a diverse superfamily that include the yeast SPT10 protein *Trends Biochem. Sci.* **22**, 154–155.
- 11 Wolf, E., Vassilev, A., Makino, Y., Sali, A., Nakatani, Y., and Burley, S. (1998) Crystal structure of a GCN5-related *N*-acetyltransferase: *Serratia marcescens* aminoglycoside 3-*N*-acetyltransferase. *Cell* **94**, 439–449.
- 12 Clements, A., Rojas, J. R., Trievel, R. C., Wang, L., Berger, S. L., and Marmorstein, R. (1999) Crystal structure of the histone acetyltransferase domain of the human PCAF transcriptional regulator bound to coenzyme A. *The EMBO Journal* **18**, 3521–3532.
- 13 Schäffer, A. A., Aravind, L., Madden, T. L., et al. (2001) Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Res.* **29**, 2994–3005.
- 14 Maglott, D., Pruitt, K., and Tatusova, T. (2005) Entrez gene: a directory of genes, in *The NCBI Handbook*, (McEntyre, J. and Ostell, J., eds.), National Library of Medicine (US), NCBI, Bethesda, MD.
- 15 Pruitt, K. D., Tatusova, T., and Ostell, J. M. (2005) The Reference Sequence (RefSeq) Project, in *The NCBI Handbook*, (McEntyre, J. and Ostell, J., eds.), National Library of Medicine (US), NCBI, Bethesda, MD, Chapter 18.

Since this chapter was sent for publication the NCBI BLAST site has undergone a major revamping. In the new system the values entered by the user are retained in “memory”. The user can also access multiple old results and reformat or rerun the same search by changing parameters. It is suggested that the readers familiarize themselves with the new front-end, before attempting further experiments along the lines suggested in this chapter.

Organizing and Updating Whole Genome BLAST Searches with ReHAB

David J. Esteban, Aijazuddin Syed, and Chris Upton

Summary

In the current genomics era, protein and DNA sequence databases are continuously growing at an exponential rate. It has become increasingly important and useful to repeat similarity searches at frequent intervals, which then retrieve larger and larger sets of results. In addition, sequence similarity searches are now often performed with many sequences or even whole genomes. ReHAB (Recent Hits Acquired from Basic Local Alignment Search Tool [BLAST]) is a tool for tracking new protein hits in repeated PSI-BLAST searches. It is designed to simplify the analysis of large numbers of database matches and is therefore especially suited to comparative genomics. Results are presented in a user-friendly graphical interface with simple-to-navigate tables and new hits are indicated by highlighted text. In this paper, we describe the use of this software for organizing results from whole virus genome PSI-BLAST searches using a ReHAB database maintained at the Virus Bioinformatics Resource Centre.

Key Words: ReHAB; BLAST; genomics; virus; bioinformatics; database; PSI-BLAST; similarity searches; VBRC.

1. Introduction

Recent advances in sequencing technologies have fostered the capacity to sequence large and complex genomes, ESTs, and other libraries, leading to an exponential increase in the volume of DNA and protein sequence data available in sequence databases. Predicting the function of an unknown protein by sequence or motif similarity to a previously characterized protein is an extremely valuable process and is the lynchpin in the data mining approach to

comparative genomics and systems biology. To keep up with the rapid growth of the databases, researchers need to repeat the same queries frequently. A further confounding problem is that new and potentially significant results are often buried in a long list of hits that were previously obtained in past searches. As a result, it is necessary to search through massive amounts of irrelevant alignment data to retrieve new and interesting matches. This is especially problematic when multiple query sequences, or all the open reading frames (ORFs) of a genome, are submitted in a Basic Local Alignment Search Tool (BLAST) search. ReHAB (Recent Hits Acquired from BLAST) is a software package that addresses these problems (1).

Using a given set of query proteins, ReHAB performs periodic PSI-BLAST (2) searches of a protein sequence database and records all significant alignments ("hits") obtained. These searches are automatically executed on a regular schedule against an updated protein database. For each protein, ReHAB then compares the results obtained in the latest PSI-BLAST search with the list of previously generated hits to identify new matches to sequences recently deposited in the database. These results are viewed through an easy to use graphical user interface, which sorts the data into tables, highlights the new hits, and performs pairwise or multiple alignments with only a few clicks of the mouse.

Along with the ReHAB program, a current ReHAB database containing query sequences and PSI-BLAST results for all translated ORFs from complete

Table 1
Contents of the ReHAB Database at www.virology.ca as of November 10, 2005

Virus family	Query sequences	Target sequences	Hits
Adenoviridae	2305	157,638	661,390
Arenaviridae	176	3635	46,712
Asfarviridae	160	37,112	42,552
Baculoviridae	3756	156,662	110,472
Bunyaviridae	127	8452	63,182
Coronaviridae	3698	12,551	1,077,289
Filoviridae	95	6022	30,215
Flaviviridae	2304	6682	1,762,501
Herpesviridae	6274	217,484	1,908,786
Paramyxoviridae	989	22,142	570,178
Poxviridae	7040	224,154	3,115,821
Togaviridae	561	8387	365,210

genomes of a large number of virus families is maintained by the Virus Bioinformatics Resource–Canada (3) at www.virology.ca (Table 1). This database, which is regularly updated, is provided as a service to the virology community. Recent improvements to ReHAB have increased the speed of the PSI-BLAST searches, allowing more frequent database updates.

The following describes the steps required to retrieve results from the ReHAB database located at www.virology.ca. A specific gene from vaccinia virus will be used to illustrate the steps. We suggest testing the program by following this example; once the user is familiar with the software, follow the same steps to analyze the genes of interest.

2. Materials

1. Computer with internet access (ReHAB is coded in Java and is therefore platform independent and will operate on PC, Apple, or Linux systems).
2. Java Web Start. This is a free program that is already installed on most machines (see Note 1).

3. Methods

The function of ReHAB is to perform periodic PSI-BLAST searches, highlight hits that are new since the last time a search was performed and present the results in a user-friendly graphical interface allowing the user to quickly identify new and interesting hits. As an example, we describe how to find new hits for the poxvirus gene *A18R* from vaccinia virus, strain Copenhagen.

3.1. Accessing the ReHAB Database and Finding New Hits

1. In an internet browser, open the page www.virology.ca.
2. In the left panel, which lists the available tools under the Workbench heading, select ReHAB. A new page opens which provides a brief description of the program.
3. Click Launch Program. A Java Web Start window will open to show the progress of the download (see Note 2). At this step, the client portion of the program is installed on your computer (see Note 3); this allows access to the database that resides on the www.virology.ca server.
4. A window titled ReHAB Management Console will open (Fig. 1). In the left panel is a list of the available ReHAB databases, organized by virus family (see Note 4). Because this example uses a protein from a poxvirus, click once on poxviridae to select this database. Statistics for this database are shown in the right panel, including the total number of query sequences (poxvirus ORFs), the number of target sequences that were found to match the query sequences, and the number of hits (if several query sequences match the same target sequence, the total number

- of hits will be larger than the total number of target sequences). From the Action menu, select Browse by Organism (*see Note 5*).
5. Another window opens, titled Browse Hits for Poxviridae (**Fig. 1B**). In the left panel, a list of poxvirus genomes is displayed (*see Note 6*). Select Vaccinia virus (Copenhagen) by clicking once (*see Note 7*).
 6. Options for filtering and highlighting are shown in the right panel. The drop-down menu headed “Mark hits as New as of” allows the user to choose the cutoff date for defining a hit as “new.” Choose Sunday, November 20 2005 (*see Note 8*).
 7. In the list of hits that will be generated, highlighting is used to mark a given hit as “new” and to show its relevance. The box called “Highlight new hits with bit score at least:” allows the user to define the minimum bit score required for a new hit to be highlighted in red. New hits with scores below the threshold are highlighted in yellow, whereas hits that were found in previous searches remain unhighlighted. Leave the setting at the default score, 50.
 8. Select the box labeled “Don’t show my own sequences.” Selecting this removes highlighting from target sequences that are also in the query sequence database (*see Note 9*).
 9. In the Sorting panel, three radio buttons allow the user to choose the order of the genes that will be presented in the next window. Leave the setting on the default, Name, which will sort alphabetically by gene name.
 10. Click the View Summary of Hits button. This opens a new window titled “Summary of New Hits with Organism: Vaccinia virus (Copenhagen),” containing a table with four columns (*see Note 10*) (**Fig. 1C**).
 11. The ID column shows the name of the query gene. Because we are interested in the gene *A18R*, scroll down to VACV-Cop-A18R. This row is highlighted in red, indicating that at least one new target sequence was found (with a score above the threshold) since the last update. The second column shows the time when the most recent hit was found for the query sequence. The score of the highest scoring new hit (there may be multiple new hits) is displayed in the column Max New Score. To show how this new score compares to previous hit scores, the column Max Score shows the maximum score of all hits for the query sequence.
 12. Select VACV-Cop-A18R by clicking once on the gene name, and choose Hits Manager from the Action menu (*see Note 11*).
 13. A new window opens, displaying the list of target sequences and the associated bit scores for gene *A18R* (*see Note 12*). This list contains all target sequences hit by the query sequence with the new hits highlighted (**Fig. 1D**).

3.2. Viewing Hits in the Hits Manger Window

The Hits Manger window displays a complete list of PSI-BLAST hits for VACV-Cop-A18R. From this window, more information about a hit of interest can be obtained, including pairwise and multiple alignments.



Fig. 1. Workflow for retrieving hits in ReHAB. **(A)** Select the virus family database in the ReHAB Management Console. **(B)** Choose the specific virus, and adjust the filtering and highlighting options to identify high scoring new hits. **(C)** Hits are displayed in the Summary of New Hits window, with genes having new high scoring hits highlighted in red, or new low scoring hits highlighted in yellow. **(D)** Selecting the gene of interest displays all hits for that gene in the Hits Manager window, where alignments and other information is obtained. **(E)** Hits can also be displayed as an HTML report.

1. In the Hits Manager window (**Fig. 1D**), select any target sequence of interest by clicking on it once.
2. To obtain a pairwise local alignment between the target sequence and *A18R*, click the Local button. The protein sequence alignment and accompanying statistics are then displayed in the bottom panel.
3. Clicking the Global button generates a pairwise global alignment between the query and target sequences, displayed in the bottom panel.
4. To generate a multiple alignment, select more than one target sequence. This can be done in three different ways: (1) by clicking and dragging the mouse while holding the mouse button, (2) by first clicking on one target sequence and then a second while holding the shift key (all sequences in between will be selected), or (3) select noncontiguous sequences by clicking on desired sequences while holding the ctrl key (on a PC) or command key (on an Apple). After selecting the sequences of interest, click the Base-By-Base button (**4**) (*see Note 13*). This collects the protein sequences from the ReHAB database and generates a multiple alignment displayed in the application Base-By-Base (*see Note 14*).
5. The GenBank file for a selected target sequence can be retrieved by clicking the GenBank button, which opens the GenBank file in the default web browser.
6. To obtain the query and target sequences in FASTA format, click the Show button. The sequences can then be copied and pasted into a word processor or text editor as desired.

3.3. Viewing Hits in the HTML Report

Though the hits manager window provides a convenient method for viewing PSI-BLAST search results, some users may prefer the traditional BLAST output, provided in the HTML Report.

1. From the Summary of New Hits window, select the gene of interest (*A18R*) by clicking once.
2. From the Action menu, choose HTML Report. A new window will then open in the default web browser (*see Note 15*), showing the PSI-BLAST results (**Fig. 1E**). Hits are displayed in the traditional output style, with a list of target sequences and scores followed by the individual alignments. Users can go directly to the alignment for a given target sequence by clicking on the link showing the hit score. More information about the target sequence can be obtained by clicking on the Info link, which opens the GenBank file in the web browser.

The ReHAB databases for viral genes at www.virology.ca are maintained and updated by the VBRC, allowing this data to be accessed by the general public. In addition to the use described here, ReHAB can also be set up as a stand-alone application utilizing a user's own database server, and with a user's own query sequences, but it should be noted that the system requirements are

different and installation of the program requires advanced computer skills. The software and installation instructions can be viewed and downloaded from <http://athena.bioc.uvic.ca/techDoc/softwaredevelopment/rehab/>.

4. Notes

1. If ReHAB does not launch following the steps given, Java Web Start may need to be installed on the computer. To do this, follow the Java Web Start Setup link at the bottom of the ReHAB web page and follow the download instructions. When ReHAB is started, Java Web Start automatically downloads any updates to the software.
2. A warning message appears when running the software for the first time since the program accesses the user's computer. There is no cause for concern; we recommend simply clicking the start button. More information on the warning is available by clicking Warnings Dialog on the ReHAB webpage.
3. A second warning will appear to ask if you would like to have ReHAB integrated into your desktop environment. Clicking "yes" will create a ReHAB icon on the desktop. To start the program again in the future, the ReHAB icon can be double clicked.
4. Each database has an arrow icon. Clicking on the arrow reveals two subcategories: Jobs and Query Sequences. Jobs shows the update progress of the database. In most cases, no information is displayed, however, if the server is currently running an update on the virus family, a progress bar will be displayed. Query Sequences displays a list of the names of the query sequences.
5. Double clicking on the database name will perform the same operation.
6. The default setting is to populate the left panel with the organism names. The left panel can also be populated with all genes in the poxviridae database organized by gene family instead of by organism. This is done by choosing family in the Group by Annotation box on the right. Click select, and the left panel will be updated with gene families. Subsequent steps for filtering, highlighting and analysis are the same as when organized by organism. This option is only available for viruses for which orthologous genes have been organized into gene families.
7. If default options are desired for **Steps 8–10**, double clicking on the virus name will open the Summary of New Hits window, skipping to **Step 12**.
8. Usually the user will use the most recent date (the default option) but for the purpose of the example we will use the most recent date at the time of this writing.
9. This is particularly useful when a new genome of that virus family has been deposited in the NCBI database and the ReHAB query database.
10. Columns can be sorted by clicking on column headings. Column width can be adjusted by dragging the divisions between headings.
11. Double clicking on the gene name will also open the Hits Manager window.
12. Columns can be sorted by clicking on column headings. Column width can be adjusted by dragging the divisions between headings.

13. Base-By-Base (BBB) can also be used to generate pairwise alignments. BBB uses Muscle (5) to generate multiple and pairwise alignments.
14. In addition to viewing the alignment, BBB provides additional editing, export and viewing features.
15. This may take several minutes to load, especially if there are many hits.

Acknowledgments

Thanks to Angelika Ehlers for assistance with programming and Cristalle Watson for assistance with editing the manuscript.

References

- 1 Whitney, J., Esteban, D. J., and Upton, C. (2005) Recent Hits Acquired by BLAST (ReHAB): a tool to identify new hits in sequence similarity searches. *BMC Bioinformatics* **6**, 23.
- 2 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 3 Esteban, D. J., Da Silva, M., and Upton, C. (2005). New bioinformatics tools for viral genome analyses at Viral Bioinformatics - Canada. *Pharmacogenomics* **6**, 271–280.
- 4 Brodie, R., Smith, A. J., Roper, R. L., Tcherepanov, V., and Upton, C. (2004) Base-by-base: single nucleotide-level analysis of whole viral genome alignments. *BMC Bioinformatics* **5**, 96.
- 5 Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**, 1792–1797.

Alignment of Genomic Sequences Using DIALIGN

Burkhard Morgenstern

Summary

DIALIGN is a software program for multiple alignment of DNA or protein sequences that combines global and local alignment features. During the last years, the program has been used extensively to compare syntenic regions in genomic sequences. An anchoring option speeds up the alignment procedure and makes it possible to use user-defined constraints to improve the quality of the program output. This chapter explains features of DIALIGN that are useful if genomic sequences are to be aligned. The program is online available through Göttingen Bioinformatics Compute Server at <http://dialign.gobics.de/>.

Key Words: Multiple sequence alignment; anchored alignment; DIALIGN; gene prediction; phylogenetic footprinting.

1. Introduction

With a growing number of partially or completely sequenced genomes, comparative analysis of genomic sequences is becoming a crucial tool for genome analysis and annotation. Practically all methods for comparative sequence analysis rely on pairwise or multiple alignments, so aligning genomic sequences is the first and most important step for comparative genomics. Most computer programs for pairwise or multiple sequence alignment that have been developed during the 1980s or 1990s are either global or local alignment methods. Global programs align the input sequence over their entire length, whereas local programs return only the most highly conserved region in the input sequences and ignore the remainder of the sequences. The former

methods are based on the famous dynamic-programming algorithm proposed by Needleman and Wunsch for global pairwise alignment (1); extensions of this algorithm to multiple alignment include CLUSTAL W (2), T-COFFEE (3), and many other approaches. Established methods for local alignment include the Smith-Waterman algorithm (4) and BLAST (5,6) for pairwise alignment and GIBBS sampling (7) for local multiple alignment.

There are two main reasons why these traditional alignment approaches are of limited use for analyzing long genomic sequences. First, many sequence families are not related over the full sequence length, but nevertheless exhibit much more sequence similarity than just a single conserved motif. Such sequence families cannot be aligned with traditional global or local alignment methods that would either align the sequences over their entire length or align only one conserved motif. This is the case for syntenic genomic sequences where conserved structures such as protein-coding genes are interrupted by intergenic regions with little or no sequence similarity, but it is also the case for many protein families. In addition, most of the alignment methods developed until the mid-1990s were designed to align rather short sequences, e.g., single genes or proteins. Consequently, these methods are far too slow for large-scale alignment of genomic sequences. For these reasons, during the last few years, a new generation of alignment program has been developed that can cope with these challenges (8–11).

The program DIALIGN has been developed for multiple alignment of distantly related protein or nucleic-acid sequence families (12,13). It combines global and local alignment features by composing pairwise and multiple alignments from local pairwise alignments, so-called fragment alignments or fragments. This way, it is able to detect and align conserved regions in the sequences but ignores nonrelated parts of the sequences, but it is not restricted to return the single best local similarity as traditional local alignment programs do. The idea is that the program itself should decide if or to what extent homology is detectable among the input sequences. Thus, for globally related sequence sets, the selected fragments will cover the sequences over their entire length and, as a result, the program would return a full global alignment. If only local similarity is detectable, however, the program would return a local alignment or an alignment composed of several local homologies separated by nonhomologous parts of the sequences.

More precisely, fragment alignments or fragments are defined as local gap-free pairwise alignments among the input sequences. Each possible fragment is given a quality score based on statistical significance, and for a given set of input sequences, the program searches a collection of fragments that can be integrated

into one resulting (pairwise or multiple) output alignment; *see* refs. 14 and 15 for more details. This way, DIALIGN tries to align all regions of the sequences that exhibit some statistically significant degree of similarity—provided they fit into one single output alignment. Because the program is able to integrate an arbitrary number of local alignments into one output alignment, it can deal with many sequence families that cannot be reasonably aligned with traditional global or local methods.

Originally, DIALIGN has been developed as an all-purpose multiple-alignment program, but it soon became clear that it is particularly useful to align genomic sequences. In fact, it was the first program that was able to align genomic DNA sequences where islands of local sequence conservation are located between nonrelated sections of the sequences. DIALIGN has been used in many studies in comparative genomics, e.g., refs. 16–18. In a pioneering series of papers, Götting et al. (19–22) used the program to identify small regulatory elements by comparing upstream regions from various vertebrate genomes. Other applications of DIALIGN to comparative genomics include identification of sequence signatures for pathogen detection (23) (*see* ref. 24 for review) and gene finding in eukaryotes (25–27). A systematic evaluation and comparison of software programs for multiple alignment of genomic sequences has been carried out by Pollard et al. (28).

2. Program Features

2.1. Sequence Similarity at the Nucleotide Level and at the Peptide Level

DIALIGN has a number of program features that are useful for aligning genomic sequences. The original version of the program had two different options to compare DNA sequences, a nucleotide-level option, where (local) sequence similarity is measured by comparing sequences in a base-by-base fashion and a peptide-level option, where (local) segments of DNA sequences are translated into peptide segments according to the genetic code, and the similarity of the DNA segments is then calculated based on the similarity of the implied peptide segments. This second option is useful if distantly related protein-coding sequences are aligned where sequence similarity may have been eroded at the nucleotide level but is still detectable among the corresponding proteins. Note that, in principle, the program compares every possible fragment alignment, i.e., every possible pair of equal-length sequence segments, thereby checking for sequence similarity in all possible reading frames. Because the program tries to identify a set of fragments with maximum total score, it selects those segment pairs that have the strongest degree of similarity. This way, the

program output implicitly contains some information about the likely reading frame in protein-coding sequences.

In the original program version, the user had to decide if nucleic-acid sequences were to be compared at the nucleotide level or at the peptide level. We later implemented a so-called mixed-alignment option, where local sequence similarity is measured automatically at three different levels, namely (1) at the nucleotide level, (2) at the peptide level on the forward strand, and (3) at the peptide level on the reverse complement (29); the score of a fragment is then defined by the level of sequence similarity is (locally) strongest. As a result, some regions in the resulting alignment may exhibit stronger similarity at the peptide level in one of the two possible orientations, whereas other regions are more strongly related at the nucleotide level. The mixed-alignment option, together with the information about the potential reading frame contained in the output alignment, is useful for gene-prediction purposes; we recently used this option to obtain additional extrinsic information for the HMM-based gene finder AUGUSTUS (27).

2.2. Anchored Alignment

Practically all software tools for sequence alignment are fully automated. This means that the user enters the input sequences and can adjust a number of program parameters, but there is no direct way of influencing the resulting alignment. However, there are many reasons why such automated alignment procedures may fail to produce biologically meaningful output alignments. We developed an anchored alignment approach where the user can specify certain sites within the input sequences that are to be aligned in the output alignment. The remainder of the sequences are then aligned according to the constraints defined by these user-defined anchor points (30,31). More specifically, the user can specify pairs of equal-length segments of the input sequences, i.e., gapfree local pairwise alignments that are to be contained in the output alignment. If it is not possible to integrate all these user-defined anchoring segment pairs into one output alignment, a suitable subset is automatically selected by the program. To this end, a score is to be defined by the user for each proposed anchor point, in case of conflicting anchor points, the program prioritizes them according to the user-defined scores.

The anchored-alignment option is useful to enforce meaningful alignments in situation where the program is not able to produce a meaningful alignment automatically, but some expert information about sequence homologies is available. Here, anchor points can help to use such expert information for

improved alignment quality. A second application of our anchoring option is to speed-up the alignment procedure for long genomic sequences. Because the program has originally been designed to align rather small input sequences, its standard version is too slow to do large-scale alignment of genomic sequences. Alignment of long sequences, however, can be accelerated considerably if known or presumed homologies are used as anchor points to reduce the alignment search space. To speed-up alignment of genomic sequences, a fast local alignment search tool such as BLAST (5) or CHAOS (32) can be used to identify anchor points.

3. Program Input and Output

3.1. DIALIGN and CHAOS at GOBICS

At Göttingen Bioinformatics Compute Server (GOBICS), we installed a website interface to use DIALIGN for multiple alignment of genomic sequences (33,34). At our website server, the previously explained anchoring option is used to speed-up the alignment procedure; anchor points are identified using the local alignment tool CHAOS (32). Alignments are visualized using the visualization tool ABC developed by Cooper et al. (35). This tool gives a graphical overview about the produced alignment and allows the user to interactively zoom in to inspect the alignment in detail. In addition, the produced alignment is returned in various formats. An online user guide gives all necessary information about input options and output format.

3.2. Command-Line Version of the Program

The source code of DIALIGN is freely available under the GNU open-source licence agreement. If the program is locally installed, more options are available than on the web server. The following program options are available for the command-line version; the corresponding program parameters are explained in the user guide that comes with the program.

3.2.1. Options for the Program Run

1. For DNA sequences, (local) sequence similarity can be calculated at the nucleotide level, at the peptide level and at both levels with the “mixed-alignment” option as previously explained. If the “peptide-level” or “mixed-alignment” option is used, it is possible to translate sequence segments only at the forward strand or to have the program look at both, the forward strand and the reverse complement.
2. There is a threshold parameter T that can be used to filter out all local sequence similarity with score below T.

3. Various options are available to further speed-up the program, but possibly at the expense of alignment sensitivity. For example, it is possible to reduce the maximum possible length of the fragments that the program uses to compose output alignments.
4. Anchor points can be specified using a command-line parameter and a special file containing coordinates and scores for the proposed anchor points. Details are explained on our webpage or in the user guide that comes with the downloadable program version.
5. As previously explained, the basic building blocks of DIALIGN are local gapfree pairwise alignments, i.e., local sequence similarity is primarily measured between pairs of sequences, and homologies involving more than two sequences are detected only as a result of such pairwise comparisons. For multiple alignment, there is a program option that prefers local similarities occurring in more than two sequences above those similarities that exist only between two input sequences. This so-called overlap-weight option generally improves alignment quality, but it is computationally expensive. By default, this option is therefore used, if not more than 35 sequences are aligned; for larger data sets, this option is switched off. With two special program options, the user can use or switch off overlap weights independent of the input data size.

3.2.2. Options for the Output Format

1. In addition to the human-readable DIALIGN alignment format, the program can return the output alignment in FASTA or MSF format.
2. The program can return a list of all fragments (i.e., local pairwise gap-free alignments) that have been used to construct the output alignment. This list contains useful information about the fragments, e.g., their coordinates and similarity scores.
3. For multiple alignment, it is possible to output a list of all fragments contained in the respective pairwise alignments, no matter if they have been finally selected for the multiple alignment (consistent fragments) or not (nonconsistent fragments). This output file distinguishes between consistent and nonconsistent fragments.
4. It is possible to output all fragments (including coordinates, fragment scores, and so on) that have been considered by the program during the alignment procedure, though this list can be very long.
5. In the standard program output, nucleotides that remain unaligned, i.e., nucleotides not involved in any of the fragments selected by the program, are printed in lower-case letters, whereas aligned residues are printed in upper-case letters. It is possible to mask unaligned residues by using a special character instead of printing them in lower-case letters.
6. Two different output options are available to indicate the degree of (relative) local sequence similarity in the alignment.

References

- 1 Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.
- 2 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 3 Notredame, C., Higgins, D., and Heringa, J. (2000) T-Coffee: a novel algorithm for multiple sequence alignment. *J. Mol. Biol.* **302**, 205–217.
- 4 Smith, T. F. and Waterman, M. S. (1981) Comparison of biosequences. *Advances in Applied Mathematics* **2**, 482–489.
- 5 Altschul, S. F., Gish, W., Miller, W., Myers, E. M., and Lipman, D. J. (1990) Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410.
- 6 Altschul, S. F., Madden, T. L., Schäffer, A. A., et al. (1997) Gapped BLAST and PSIBLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 7 Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993) Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science* **262**, 208–214.
- 8 Brudno, M., Do, C., Cooper, G., et al. (2003) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 9 Höhl, M., Kurtz, S., and Ohlebusch, E. (2002) Efficient multiple genome alignment. *Bioinformatics* **18**, 312S–320S.
- 10 Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. (1999) Alignment of whole genomes. *Nucleic Acids Res.* **27**, 2369–2376.
- 11 Bray, N., and Pachter, L. (2003) MAVID multiple alignment server. *Nucleic Acids Res.* **31**, 3525–3526.
- 12 Morgenstern, B., Dress, A. W. M., and Werner, T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA* **93**, 12,098–12,103.
- 13 Morgenstern, B. (2004) DIALIGN: Multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Res.* **32**, W33–W36.
- 14 Morgenstern, B., Frech, K., Dress, A. W. M., and Werner, T. (1998) DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics* **14**, 290–294.
- 15 Morgenstern, B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics* **15**, 211–218.

- 16 Prohaska, S. J., Fried, C., Flamm, C., Wagner, G., and Stadler, P. F. (2004) Surveying phylogenetic footprints in large gene clusters: applications to Hox cluster duplications. *Mol. Phyl. Evol.* **31**, 581–604.
- 17 Wagner, G. P., Fried, C., Prohaska, S. J., and Stadler, P. F. (2004) Divergence of conserved non-coding sequences: rate estimates and relative rate tests. *Mol. Biol. Evol.* **21**, 2116–2121.
- 18 Blanchette, M. and Tompa, M. (2002) Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res.* **12**, 739–748.
- 19 Göttgens, B., Barton, L. M., Gilbert, J. G. R., et al. (2000) Analysis of vertebrate SCL loci identifies conserved enhancers. *Nat. Biotechnol.* **18**, 181–186.
- 20 Göttgens, B., Gilbert, J. G. R., Barton, L. M., et al. (2001) Long-range comparison of human and mouse SCL loci: localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences. *Genome Res.* **11**, 87–97.
- 21 Göttgens, B., Barton, L., Chapman, M., et al. (2002) Transcriptional regulation of the stem cell leukemia gene (SCL) comparative analysis of five vertebrate SCL loci. *Genome Res.* **12**, 749–759.
- 22 Chapman, M. A., Charchar, F. J., Kinston, S., et al. (2003) Comparative and functional analysis of LYL1 loci establish marsupial sequences as a model for phylogenetic footprinting. *Genomics* **81**, 249–259.
- 23 Fitch, J. P., Gardner, S. N., Kuczmarski, T. A., et al. (2002) Rapid development of nucleic acid diagnostics. *Proc. IEEE* **90**, 1708–1721.
- 24 Chain, P., Kurtz, S., Ohlebusch, E., and Slezak, T. (2003) An applications-focused review of comparative genomics tools: capabilities, limitations, and future challenges. *Brief. Bioinform.* **4**, 105–123.
- 25 Rinner, O. and Morgenstern, B. (2002) AGenDA: gene prediction by comparative sequence analysis. *In Silico Biol.* **2**, 195–205.
- 26 Stanke, M., Schöffmann, O., Morgenstern, B., and Waack, S. (2006) Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics* **7**, 62.
- 27 Stanke, M., Tzvetkova, A., and Morgenstern, B. (2006) AUGUSTUS+ at EGASP: using EST, protein and genomic alignments for improved gene prediction in the human genome. *Genome Biol.* **7**, 1–8.
- 28 Pollard, D. A., Bergman, C. M., Stoye, J., Celniker, S. E., and Eisen, M. B. (2004) Benchmarking tools for the alignment of functional noncoding DNA. *BMC Bioinformatics* **5**, 6.
- 29 Morgenstern, B., Rinner, O., Abdeddaïm, S., Haase, D., Mayer, K., Dress, A., and Mewes, H. -W. (2002) Exon discovery by genomic sequence alignment. *Bioinformatics* **18**, 777–787.
- 30 Morgenstern, B., Werner, N., Prohaska, S. J., et al. (2005) Multiple sequence alignment with user-defined constraints at GOBICS. *Bioinformatics* **21**, 1271–1273.

- 31 Morgenstern, B., Prohaska, S. J., Pöhler, D., and Stadler, P. F. (2006) Multiple sequence alignment with user-defined anchor points. *Algorithms Mol. Biol.* **1**, 6.
- 32 Brudno, M., Chapman, M., Göttgens, B., Batzoglou, S., and Morgenstern, B. (2003) Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics* **4**, 66.
- 33 Brudno, M., Steinkamp, R., and Morgenstern, B. (2004) The CHAOS/DIALIGN WWW server for multiple alignment of genomic sequences. *Nucleic Acids Res.* **32**, W41–W44.
- 34 Pöhler, D., Werner, N., Steinkamp, R., and Morgenstern, B. (2005) Multiple alignment of genomic sequences using CHAOS, DIALIGN and ABC. *Nuc. Acids Res.* **33**, W523–W524.
- 35 Cooper, G. M., Singaravelu, S. A. G., and Sidow, A. (2004) ABC: software for interactive browsing of genomic multiple sequence alignment data. *BMC Bioinformatics* **5**, 192.

An Introduction to the Lagan Alignment Toolkit

Michael Brudno

Summary

The Lagan Toolkit is a software package for comparison of genomic sequences. It includes the CHAOS local alignment program, LAGAN global alignment program for two, or more sequences and Shuffle-LAGAN, a “glocal” alignment method that handles genomic rearrangements in a global alignment framework. The alignment programs included in the Lagan Toolkit have been widely used to compare genomes of many organisms, from bacteria to large mammalian genomes. This chapter provides an overview of the algorithms used by the LAGAN programs to construct genomic alignments, explains how to build alignments using either the standalone program or the web server, and discusses some of the common pitfalls users encounter when using the toolkit.

Key Words: Alignment algorithms; rearrangements; alignment visualization.

1. Introduction

Comparing genomic sequences across related species has become one of the standard methods to locate functional regions in genomes. These regions, for example exons, tend to exhibit significant sequence similarity as a result of purifying selection, whereas regions that are not functional evolve more rapidly and hence are not as conserved. The first step in comparing genomic sequences is to align them—that is, to map the letters of the sequence to each other. There are several categories of alignments programs: local aligners identify local regions of similarity between the sequences, without reference to their order. Global alignments find a mapping between the letters of the sequences that is constrained not to

allow changes of order—in biological terms they do not allow for rearrangements such as inversions or transpositions. It is also common to differentiate between pairwise alignment algorithms, which can only compare two sequences, and multiple ones, that can align a larger number. The main challenge in genomic alignments is to develop algorithms that are fast enough to deal with megabase long sequences and gigabase long genomes, while accurately mapping individual basepairs. The main motivation for the development of several new alignment tools, such as LAGAN, MultiZ, MAUVE, MAVID, and TBA (*1–5*) in 2003–2004 was the dearth of tools capable of multiple alignment of long genomic sequences. Pairwise alignment tools such as GLASS, AVID, and BLASTZ (*6–8*) could not align more than two sequences, whereas multiple alignment tools as CLUSTAL-W and DIALIGN (*9,10*) could not handle more than a few kilobases of sequence. Additionally, many tools such as BLAST (*11*) and CLUSTAL-W were developed when the bulk of available biological sequence data were proteins rather than genomic DNA. Because proteins evolve in a very different manner than genomic segments the tools developed for proteins do not work as well for genomic sequence alignment, as has been noted by Bergman and Kreitman (*12*).

This chapter will provide an overview of the LAGAN Toolkit (*see Note 1*) for sequence alignment (*1,13,14*). The toolkit consists of three main alignment programs—CHAOS, a pairwise local aligner, LAGAN, a multiple global aligner, and Shuffle-LAGAN, a pairwise alignment program for sequences with rearrangements. LAGAN has become one of the commonly used alignment programs with over 150 citations in the 3 years since its publication. LAGAN has also been incorporated into several pipelines for whole-genome alignment (*15–17*). The LAGAN Toolkit source code is publicly available under the GNU Public License and can be downloaded directly from <http://lagan.stanford.edu>. LAGAN and Shuffle-LAGAN can also be used through a web server, to which users can submit their sequences and get both a textual alignment and visualization using the VISTA visualization program (*18*). In the following sections, we will provide an overview of the algorithms used in LAGAN to construct genomic alignments, explain how to use the toolkit to build alignments both with the standalone program and through the web, and, finally, discuss some of the common problems LAGAN users encounter and show how they can be solved.

2. Algorithms

This section presents a brief overview of the algorithms used in CHAOS, LAGAN, and Shuffle-LAGAN alignment programs. Many of the details of the algorithms are beyond the scope of this chapter, and we point the interested reader to the original papers.

2.1. CHAOS Local Alignment Algorithm

The CHAOS local alignment program is the “brains” behind the LAGAN aligner. As will be shown in the next section, LAGAN is an anchored alignment approach, in which the accuracy depends on the local alignments that it uses as anchors—these are provided by CHAOS, which is run several times with different parameters.

The CHAOS algorithm works by chaining together pairs of matching sequences, one from each of the two input DNA sequences; these pairs are usually called seeds. More precisely, a seed is a pair of words of length k with at least n identical basepairs (bp). A seed s_1 can be chained to another seed s_2 whenever (1) the indices of s_1 in both sequences are higher than the indices of s_2 , and (2) s_1 and s_2 are “near” each other, with “near” defined as separated by at most distance basepairs in either sequence and at most gap diagonals apart. The chaining algorithm is summarized in [Fig. 1](#). Once a chain cannot be extended any further (no other seeds can be chained to its end) it is referred to as a maximal chain. The initial score of a maximal chain is the total number of

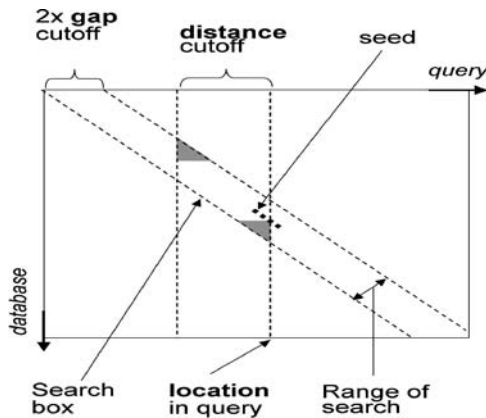


Fig. 1. The CHAOS algorithm. The rectangle represents the Smith-Waterman dynamic programming matrix, with one of the sequences along each axis. The seed shown can be chained to any seed that lies inside the search box. All seeds located less than distance basepair from the current location are stored in a skip list, in which we do a range query for seeds located within a gap cutoff from the diagonal on which the current seed is located. The seeds located in the gray areas are not available for chaining to make the algorithm independent of sequence order. (Figure reprinted with permission from [ref. 14](#).)

matching base pairs in it. The default parameters used by CHAOS are words of length 10, with a degeneracy of one ($n = k - 1$), a distance and gap criteria of 20 and 5 bp, respectively, and a score cutoff of 25. Alternatively, CHAOS can translate the sequences into amino acids in the six possible frames, and do the comparison at the protein level.

After finding all of the maximal chains that pass the score cutoff, CHAOS rescores each chain by performing ungapped extensions in both directions from each seed, and finding the optimal location to insert exactly one gap between them. The matches and mismatches are scored with an arbitrary substitution matrix. CHAOS can be used as a stand-alone program for local sequence alignment or as a preprocessing step to find anchor points for other alignment programs, as it is used in the CHAOS/DIALIGN program (14) and in the LAGAN suite of tools, which are discussed next.

2.2. Pairwise LAGAN

LAGAN is a global alignment algorithm. Global alignments find the correspondence between two strings end-to-end by building a monotonically increasing map between the letters of each sequence. The original global alignment algorithm is Needleman-Wunsch (19), which requires time proportional to the product of the lengths of the aligned sequences. This algorithm was too inefficient for comparing megabase-long genomic sequences, and faster and more accurate methods have been developed recently, such as MUMmer (20–21), GLASS (6), WABA (22), and AVID (7). All these methods rely on an anchoring approach. This can be summarized as follows: (1) generate the fragments (stretches of local homology) between the sequences, (2) resolve the set of fragments into the highest scoring consistent set of anchors using the sparse dynamic programming approach or some alternative (this consistent set is referred to as a rough global map), and (3) run a thorough global alignment algorithm such as Needleman-Wunsch between the anchors.

The LAGAN alignment algorithm, unlike previous methods, which used *k*-mers or other short matching sequences (similar to seeds in the previous section) as anchor points, uses full local alignment (generated by CHAOS) as anchors. These local alignments, although more expensive to compute, often allow for a more accurate global map. However, they also create the additional computational difficulty in that they cannot be used as “fixed anchors” as an optimal alignment which goes through some anchor point is not guaranteed to go through the exact same positions as the local alignment did. Consequently LAGAN adopted a “flexible anchoring” framework, by building “necks” around the anchor points and allowing the global alignment to go anywhere in the

neck. Using these “necks” the program defines the concept of a “limited area” and runs the Needleman-Wunsch global alignment algorithm just in the limited area. The anchoring approach provides for a significant speedup over the regular Needleman-Wunsch algorithm because the limited area is typically much smaller than the whole matrix. **Figure 2** provides a visual explanation of the algorithm.

2.3. LAGAN for Three or More Sequences

For alignment of three or more sequences LAGAN combines the anchoring approach with the progressive alignment technique commonly used in protein alignment programs such as CLUSTAL-W. In the progressive alignment technique the sequences are aligned pairwise up the phylogenetic tree, with each internal node of the tree representing the alignment of its descendants. Because every internal node has exactly two children, each of the alignment steps is pairwise. Furthermore, an alignment can be considered to be a string over a larger alphabet (gaps can be treated as a fifth symbol), and can be aligned using any pairwise alignment algorithm. The two challenges in extending the

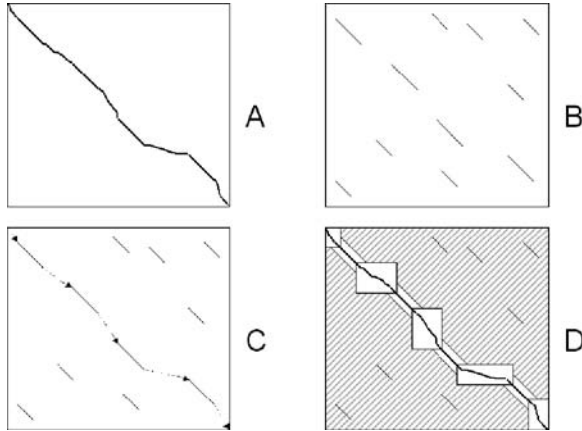


Fig. 2. An overview of the LAGAN algorithm. The algorithm attempts to recreate the optimal map between the sequences, illustrated in **A**. It starts by finding the potential anchors (**B**), finds the highest scoring increasing subset of them (**C**), and finally conducts dynamic programming in the limited area around the anchors (**D**). The shaded areas are ignored, allowing for a large speedup over the classic Needleman-Wunsch algorithm, which evaluates every cell of the rectangular matrix. (Figure reprinted with permission from **ref. 1**).

LAGAN approach to multiple sequence alignments are (1) how to score a multiple alignment and (2) how to create anchors for it.

The most common method used to score alignments is the sum-of-pairs scoring, where the score for a particular column is set defined as the sum of all the pairwise substitution and gap events. Alternatively, it is possible to use a consensus model: for every column one finds the most likely character and penalizes divergence from the character. LAGAN combines these approaches: it uses sum of pair scoring for matches and mismatches and consensus for gaps.

The LAGAN program uses all pairwise local alignments between the sequences in alignments to generate the set of anchor points for progressive alignment. For example, given the sequences X and Y, the alignment between them X/Y and a third sequence Z, the anchors between X/Y and Z are computed as follows: first, all anchors in the rough global maps between X and Z, and between Y and Z, are mapped from their coordinates in the X and Y sequences to their coordinates in the X/Y alignment and become potential anchors between X/Y and Z, with score equal to their original score. Second, for each pair of anchors between X and Z and between Y and Z that overlap, a new potential anchor is created with a new score set to $(s_1 + s_2) * I/U$, where s_1 , s_2 are the scores of the (X,Z) and (Y,Z) anchors, respectively, I is the length of intersection, and U is the length of union of the two original anchors (summed in X/Y and Z). The rough global map between X/Y and Z is the highest scoring consistent chain of all of these anchors (see Fig. 3). This chain is found using the same sparse dynamic programming approach used to find pairwise LAGAN anchor points.

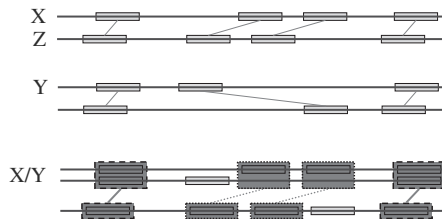


Fig. 3. Generation of anchors during progressive alignment. Multisequence X/Y is aligned to sequence Z. Anchors between X and Z (top) and anchors between Y and Z (middle) are remapped to coordinates in the X/Y multisequence, and given a new score. Then, the Longest Increasing Subsequence algorithm is applied to select a subset of the remapped anchors, as the anchors between X/Y and Z. (Figure reprinted with permission from ref. 1.).

2.4. Shuffle-LAGAN for Rearrangements

The Shuffle-LAGAN (S-LAGAN algorithm) was built on top of the LAGAN global alignment framework to allow for alignment of sequence with rearrangements. The S-LAGAN algorithm consists of three distinct stages. During the first stage the local alignments between the two sequences are found using the CHAOS tool. Second, the subset of the local alignments with the maximum score under certain gap penalties is picked to form a 1-monotonic conservation map. It is the structure of this map, found by a novel chaining technique, that makes S-LAGAN different from standard anchored global aligners. Finally, the local alignments in the conservation map that can be part of a common global alignment are joined into maximal consistent subsegments, which are aligned using the LAGAN global aligner. See [Figure 4](#) for a graphical overview of the algorithm.

2.4.1. Building the 1-Monotonic Conservation Map

Most tools for rapid global alignment start with a set of local alignments, which they resolve into a “rough global map”—the set of anchors described in [Subheading 2.2](#). The rough global map must be nondecreasing in both sequences. To allow S-LAGAN to catch rearrangements, this assumption is relaxed to allow the map to be nondecreasing in only one sequence, called the base, without putting any restrictions on the second sequence. This is called a 1-monotonic conservation map.

To build this map, we first sort all of the local alignments based on their coordinates in the base genome. For every next alignment, we chain it to the previous one that gives the highest overall score subject to the affine chaining

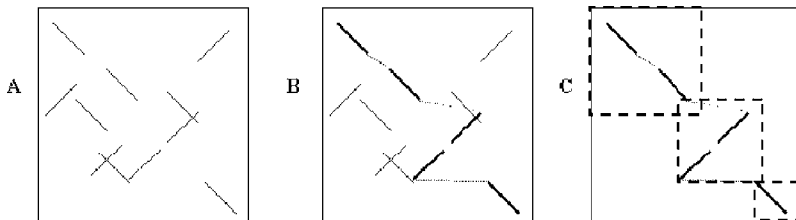


Fig. 4. An overview of the Shuffle-LAGAN algorithm. (A) The local alignment between the two sequences are generated using CHAOS. (B) The highest scoring 1-monotonic map (indicated in bold) is found. (C) The maximal consistent subsegments of the 1-monotonic map (dashed boxes) are aligned using LAGAN. (Figure reprinted with permission from [ref. 2](#)).

penalties. The penalty enforced depends on whether the previous alignment is on the same or different strand than the previous one, and whether it is before or after it in the coordinates of the second sequence. Roughly speaking, the four cases correspond to regular gap (same strand, after), inversion (different strand, after), translocation (same strand, before), and inverted translocation (different strand, before). The resulting highest scoring chain is 1-monotonic (strictly increasing in the base genome, but without any restrictions on the second genome order). The 1-monotonic chain can capture all rearrangement events besides duplications in the second genome.

2.4.2. Aligning Consistent Subsegments

Two local alignments are considered to be consistent if they can both be a part of a global alignment. Once we have a 1-monotonic conservation map it is straight-forward to generate the maximal consistent subsegments of the map by simply sorting all of the local alignments in the 1-monotonic map by their coordinates in the base sequence, taking the first alignment to be the start of a consistent subsegment, and adding additional local alignments while they are all consistent. As soon as an alignment is found to be inconsistent with the current subsegment, we start a new subsegment. Every consistent subsegment is extended to the nearest adjacent local alignment, so as to include areas of homology that did not fall into the local alignment, and are aligned using LAGAN. The overlap between adjacent consistent subsegments is resolved by doing a linear pass through the two alignments. This pass finds the optimal breakpoint that ends the first alignment and starts the second one.

3. Using the LAGAN Toolkit

The previous section provided an overview of the algorithms behind the LAGAN tools. In this section, we will describe the practical usage of the tools both from their stand-alone versions and through the web interface including some of the commonly changed parameters. We will also illustrate two methods of visualizing LAGAN alignments.

3.1. LAGAN Implementation and Availability

The LAGAN toolkit was implemented on a Linux platform in C and Perl, though some parts of Shuffle-LAGAN and some of the utilities are in C++. It is available under the GNU General Public License (Open Source). The source code of the program can be downloaded from the LAGAN website at <http://lagan.stanford.edu>. Although executables are not provided, the LAGAN distribution comes with several README files (in the Readmes directory) that

lay out the steps necessary to compile LAGAN from a command prompt on any UNIX or Linux machine (including Apple Macintosh machines running MacOS X). It is also possible to use LAGAN on a Windows PC by first installing the cygwin package (<http://www.cygwin.com/>) that emulates a UNIX environment under Windows. Finally LAGAN and Shuffle-LAGAN can also be used through the web at the joint LAGAN/VISTA server, also available at the link previously listed.

3.1.1. Tools

The LAGAN toolkit comes with the alignment programs previously described as well as some useful utilities for manipulating and printing sequences and alignments. These are located in the `utils` subdirectory of the distribution and include tools to reverse complement a sequence (`rc`), convert between various alignment formats (BINary, BLast, Multi-Fasta) (`bin2bl`, `bin2mf`, `mf2bin`), pretty-print a multiple alignment in Multi-Fasta format (`mpretty`) and project an alignment into subalignments, including sequences (`mproject`). These utilities are documented in the `README.tools` file.

3.1.2. Visualization

LAGAN is a command-line program and does not have any graphical interface. There are, however, several tools for visualizing sequence alignments that can be used with LAGAN's output. These include the VISTA programs, including Phylo-VISTA (23), Sockeye (24), and any other program that supports Multi-FASTA format input. Figures 5 and 6 show parts of LAGAN multiple alignments in eight and four species displayed with Phylo-VISTA and Sockeye, respectively. The plots allow the biologists to quickly visualize which regions of a genomic sequence are conserved and which ones are not.

For the original VISTA software, the LAGAN utilities include the script `mviz.pl` for automatically generating a VISTA plot from a multiple alignment. The script requires the user to define a `$VISTA_DIR` showing the location of the VISTA executables and the script `RunVista` in that directory for launching VISTA. The script will automatically create the plotfile used by VISTA using user provided parameters.

3.2. LAGAN on the Web

3.2.1. Using the LAGAN/VISTA Server for Sequence Alignments

Lagan is available through the joint LAGAN/VISTA server accessible at <http://lagan.stanford.edu>. The user is asked to input their sequences and give

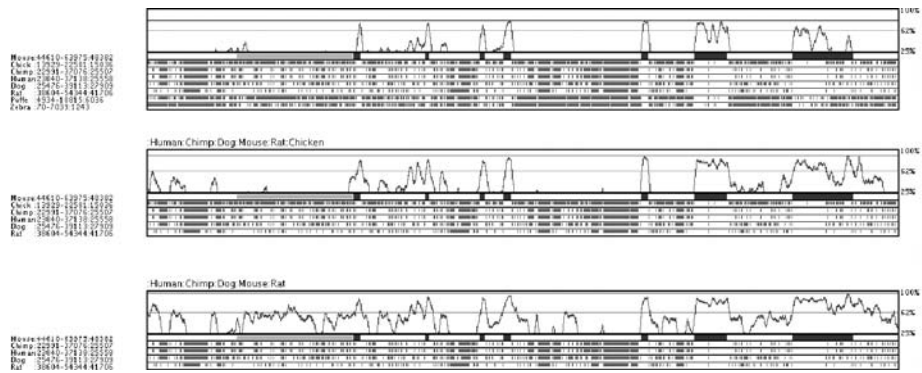


Fig. 5. Phylo-VISTA visualization of the MLAGAN alignment of the stem cell leukemia (SCL) gene in eight vertebrates (human, chimp, mouse, rat, dog, chicken, pufferfish, and zebrafish). The top plot shows the similarity between all the fish sequences and the rest of the organisms, the second one between the chicken and the mammals, and the final one between the rodents and the rest of the mammals. The height of the peak indicates the percent identity in the alignment. As one considers more distant genomes (higher plots) there is less conservation, though the exons are clearly conserved among all vertebrates.

an e-mail address to which a link to the results should be sent. The user is also asked to provide the organisms from which the sequences come to identify the repetitive elements that may confuse the alignment program, and is allowed to choose to use translated anchoring (both repeat masking and translated anchoring are discussed in the next section). The web server allows for alignments with both LAGAN and Shuffle-LAGAN. The results include not only a textual alignment, but also its visualization in the VISTA Browser (15), as well as using the original VISTA standalone software (18). Those users which request a S-LAGAN alignment are also reported a list of rearrangements between the sequences, both as a list of aligned regions and as a dotplot in pdf format.

3.2.2. LAGAN in Whole-Genome Pipelines

LAGAN is a global alignment program, and hence cannot be directly used for whole genome alignment (it does not handle rearrangements). However, it has been widely used as parts of larger pipelines for whole Genome alignment, including the Berkeley Genome Pipeline (15,25) (<http://pipeline.lbl.gov>), Baylor PASH pipeline (16) (<http://brl.bcm.tmc.edu/csa/index.rhtml>), and ENSEMBL (17) (<http://www.ensembl.org/info/data/compara/index.html>). In all

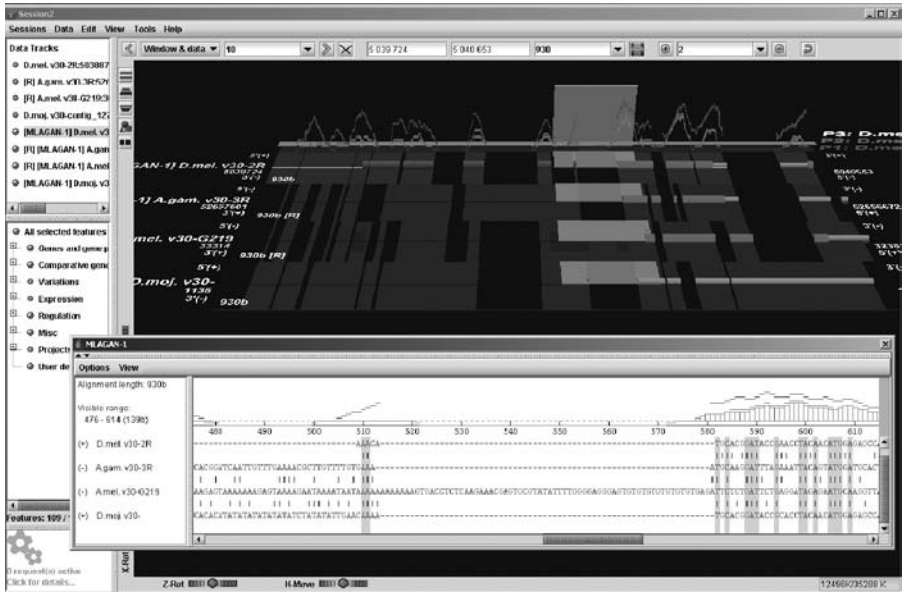


Fig. 6. Sockeye visualization of an MLAGAN alignment of four insect genomes. The sequence on the right represent a conserved block between *Drosophila melanogaster*, *Drosophila mojavensis* (two fruit flies), *Anopheles gambiae* (spider), and *Apis mellifera* (honeybee) around the 5' UTR (red) and the first coding exon (green) of a gene. The plot shows that there is substantial conservation in the coding sequence, right up to the start site, and also some conservation in the 5' UTR and upstream. (Figure courtesy of Erin Pleasance of the British Columbia Genome Sciences Centre.).

of these pipelines a heuristic local alignment program is used to identify likely homologous areas, and LAGAN is then used for alignment between them. The Berkeley Genome Pipeline in particular uses a variation of the S-LAGAN chaining algorithm to find the likely homologous blocks after locating the pairwise similarities using the BLAT (26) program.

3.3. Setting Parameters

As most alignment tools, LAGAN has a very large number of parameters that can be set by the user. The pairwise version of LAGAN (executable `lagan.pl`) allows the user to change not only the classical substitution scores and gap penalties, but also anchoring parameters (`-recurse`: the parameters used in the calls to CHAOS), and a “translated” option (`-translate`), where the anchoring is partially done not on the genomic sequences but on their translations into

amino acids. This option has been shown to improve alignment accuracy of exons, especially for distant species. The earlier multiple sequence versions of LAGAN (executable `mlagan`) required the user to specify the phylogenetic tree (-tree), however starting with the current version (2.0) the tree can be generated automatically. In addition to the LAGAN parameters, Shuffle-LAGAN allows the user to set penalties for the various rearrangement events and the local alignment parameters used in the initial step. Most of these parameters are set on the command line, with the most notable exception being the substitution and gap scores that are set in the `nucmatrix.txt` file. This file must be in the directory pointed to by the environment variable `$LAGAN_DIR`. The LAGAN tools allow for the use of repeat masking information during alignment. When one is aligning the sequence in file `seq1.fa` if the file `seq1.fa.masked` is also present in the same directory, the programs will use the masked file for anchoring the alignments, but will align the unmasked sequences. Hence by providing LAGAN with masked sequences it is possible to allow it to use information about commonly found repetitive elements while aligning. When using LAGAN through the LAGAN/VISTA web server the user is asked for which repeat library to use for alignments, and the sequence provided will be automatically masked using the user-chosen libraries.

Although LAGAN's default parameters are optimal under many typical conditions, the user may want to adjust the substitution matrix, gap penalties, and anchoring parameters depending on the sequences being compared. In particular, the Shuffle-LAGAN parameters are less robust, and need to be adjusted especially when comparing long sequences: for sequences > 1 MB in length it is better to add the argument `-chaosfl "-wl 11 -nd 0 -co 15 -ext -rsc 2500 -b,"` which tells the algorithm to run CHAOS with less time and memory intensive parameters, as otherwise it may never complete. In practice, LAGAN has many more parameters than are laid out in the documentation; these are hard-coded into the text of the program, most often in constants defined at or near the top of each source file, and adventurous users with programming experience are welcome to modify the source code of the program to try to achieve better results.

3.4. Common Problems

Most of the common LAGAN problems can be resolved by referencing the README files included with the programs. Perhaps the most common problem is the user forgetting to set the `$LAGAN_DIR` environment variable. Whether it is set can be checked using the command `"echo $LAGAN_DIR,"` which will print the current setting. In general, users may want to add the setting to

their configuration file, so that this variable is always set. This can be done by adding an appropriate line to the `.bashrc` or `.cshrc` file, depending on the user's shell. The user should contact a system administrator if they are not certain how to do this.

After the package is installed, the most common problem is when users get no “meaningful” alignment from their queries when using LAGAN, but they know that there is a BLAST hit, for example. This is most commonly because of the hits being on the negative strand (reverse-complemented). Two possible solutions are to reverse complement the sequence (for example using the `rc` program included in the tools) and align these, or to try using Shuffle-LAGAN on the sequences, which catches inversions and other rearrangements automatically.

Another common source of bad alignments is not providing the alignment programs with repeat masking information. This can be diagnosed by looking at the output of `lagan`, which prints the names of the files it is using to create the anchor points. If the files specified do not have a `.masked` extension, masking information could not be located and hence was not used.

4. Notes

1. LAGAN is the name of both the whole toolkit, consisting of several programs, and of the global alignment program within the toolkit. We will use “LAGAN Toolkit” or the “toolkit” to refer to the whole suite, and just LAGAN to refer to the program. Additionally, the version of LAGAN that aligns more than two sequences has often been called MLAGAN or Multi-LAGAN. In this chapter, we will use “LAGAN” to refer to both the pairwise and multiple versions of this program.

Acknowledgments

Many people have contributed to the development of the LAGAN Toolkit during its development. Michael F. Kim and Chuong B. Do were actively involved in the original development, including writing many of the utilities. Sanket Malde and Mukund Sundararajan developed the 1-monotonic chaining algorithm, and Serafim Batzoglou supervised the development of the software (and the degrees of the people writing it).

Our early users bore the brunt of the bugs in the package, with Kerrin Small and Gregory M. Cooper (working with Arend Sidow) deserving special recognition for being the first to use the programs on their own. Alexander Poliakov and Inna Dubchak were the first to use LAGAN in a large scale pipeline and hence also helped identify several problems with the software.

This manuscript is partially based on the original Lagan and CHAOS papers as well as on a book chapter appearing in the Handbook of Computational Biology (S. Aluru, ed). The author's research was funded by the NSF Graduate Fellowship Award and the NSERC Discovery Grant during the writing.

Finally, I would like to thank the many users of LAGAN (both standalone and through the website) who have made this into a popular alignment package while also keeping the developers apprised of the problems and being patient as the problems were fixed.

References

- 1 Brudno, M., Do, C. B., Cooper, G. M., et al., and NISC Comparative Sequencing Program. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 2 Schwartz, S., Elnitski, L., Li, M., et al., NISC Comparative Sequencing Program. (2003) MultiPipMaker and supporting tools: alignments and analysis of multiple genomic DNA sequences. *Nucleic Acids Res.* **31**, 3518–3524.
- 3 Darling, A. C., Mau, B., Blattner, F. R., and Perna, N. T. (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res.* **14**, 1394–1403.
- 4 Bray, N. and Pachter, L. (2004) MAVID: constrained ancestral alignment of multiple sequences. *Genome Res.* **14**, 693–699.
- 5 Blanchette M, Kent WJ, Riemer C, et al. (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* **14**, 708–715.
- 6 Batzoglou, S., Pachter, L., Mesirov, J. P., Berger, B., and Lander, E. S. (2000) Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.* **10**, 950–958.
- 7 Bray, N., Dubchak, I., and Pachter, L. (2003) AVID: A global alignment program. *Genome Res.* **13**, 97–102.
- 8 Schwartz, S., Kent, W. J., Smit, A., et al. (2003) Human-mouse alignments with BLASTZ. *Genome Res.* **13**, 103–107.
- 9 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 10 Morgenstern, B., Frech, K., Dress, A., and Werner, T. (1998) DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics* **14**, 290–294.
- 11 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.

- 12 Bergman, C. M. and Kreitman, M. (2001) Analysis of conserved noncoding DNA in *Drosophila* reveals similar constraints in intergenic and intronic sequences. *Genome Res.* **11**, 1335–1345.
- 13 Brudno, M., Malde, S., Poliakov, A., et al. (2003) Glocal alignment: finding rearrangements during alignment. *Bioinformatic* **19**, 54i–62i.
- 14 Brudno, M., Chapman, M., Gottgens, B., Batzoglou, S., and Morgenstern, B. (2003) Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics* **4**, 66.
- 15 Brudno, M., Poliakov, A., Salamov, A., et al. (2004) Automated whole-genome multiple alignment of rat, mouse, and human. *Genome Res.* **14**, 685–692.
- 16 Kalafus, K. J., Jackson, A. R., Milosavljevic, A. (2004) Pash: efficient genome-scale sequence anchoring by positional hashing. *Genome Res.* **14**, 672–678.
- 17 Hubbard T, Andrews D, Caccamo M, et al. (2005) Ensembl 2005. *Nucleic Acids Res.* **33**, D447–D453.
- 18 Mayor, C., Brudno, M., Schwartz, J. R., et al. (2000) VISTA: visualizing global DNA sequence alignments of arbitrary length. *Bioinformatics* **16**, 1046–1047.
- 19 Needleman, S. B. and Wunsch, C. D. (1970) An efficient method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol.* **48**, 444–453.
- 20 Delcher, A. L., Kasif, S., Fleischman, R., Peterson, J., White, O., and Salzberg, S. L. (1999) Alignment of whole genomes. *Nucleic Acids Res.* **27**, 2369–2376.
- 21 Delcher, A. L., Phillippy, A., Carlton, J., and Salzberg, S. L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.* **30**, 2478–2483.
- 22 Kent, W. J. and Zahler, A. M. (2000) Conservation, regulation, synteny, and introns in a large-scale *C. briggsae*-*C. elegans* genomic alignment. *Genome Res.* **10**, 1115–1125.
- 23 Shah, N., Couronne, O., Pennacchio, L. A., et al. (2004) PhyloVISTA: an interactive visualization tool for multiple DNASequence alignments. *Bioinformatics* **20**, 636–643.
- 24 Montgomery, S. B., Astakhova, T., Bilenky, M., et al. (2004) Sockeye: a 3D environment for comparative genomics. *Genome Res.* **14**, 956–962.
- 25 Couronne, O., Poliakov, A., Bray, N., et al. (2002) Strategies and tools for whole genome alignments. *Genome Res.* **13**, 73–80.
- 26 Kent, J. (2002) BLAT: the BLAST-like alignment tool. *Genome Res.* **12**, 656–664.

Aligning Multiple Whole Genomes with Mercator and MAVID

Colin N. Dewey

Summary

The availability of an increasing number of whole genome sequences presents us with the need for tools to quickly put them into a nucleotide-level multiple alignment. Mercator and MAVID are two programs that can be combined to accomplish this task. Given multiple whole genomes as input, Mercator is first used to construct an orthology map, which is then used to guide nucleotide-level multiple alignments produced by MAVID. These programs are both fast and freely available, allowing researchers to perform genome alignments on a single laptop. This tutorial will guide the researcher through the steps required for whole-genome alignment with Mercator and MAVID.

Key Words: Orthology map; whole-genome alignment; multiple alignment.

1. Introduction

This tutorial will guide the user through the process of aligning multiple whole genome sequences with Mercator (1) and MAVID (2). Both programs are freely available and allow researchers to align moderately sized genomes on a single laptop. The combination of Mercator and MAVID is an example of a hierarchical strategy for aligning genomes (3). First, Mercator is used to construct an orthology map between the input genomes, which is a high-level one-to-one mapping between genomic segments. The second step is to run MAVID, a global multiple alignment program, on the sets of orthologous (and colinear) segments specified by the orthology map. The result is a set of

multiple alignments with the property that every nucleotide is part of at most one multiple alignment.

For the tutorial, we will align the genomes of three fruit fly species: *Drosophila melanogaster*, *Drosophila yakuba*, and *Drosophila ananassae*. The genome sequences of the first two species are organized into chromosomes, whereas that of the third is currently comprised of more than 10,000 unmapped scaffolds. The tutorial will begin with the downloading of the raw genome sequences. The author will then describe how to prepare the genome sequences and create an orthology map between them using Mercator. Procedures for comparatively scaffolding genomes and discovering rearrangement breakpoints with Mercator will also be described. The tutorial will conclude with the generation of nucleotide-level alignments using MAVID, and the extraction of a specific interval from the resulting whole-genome alignment.

2. Materials

For the purposes of this tutorial, it is assumed that a UNIX-like computing environment (e.g., Linux, Mac OS X, or Cygwin on Microsoft Windows) is being used. All software distributions listed in **Table 1** should be downloaded and compiled. Compiled binaries should all be made available through the PATH environment variable.

3. Methods

This tutorial will specify every command, in order, for the processing and alignment of three fruit fly genomes. Commands to be run will be specified by lines beginning with \$. The output for some commands will be shown and truncated output will be indicated by ellipses (...). Approximate running times for selected commands will be specified as comments. Running times are for an Apple PowerBook with a 1.25 GHz PowerPC G4 processor and 1 GB of RAM.

Table 1
Websites of Programs for the Alignment of Multiple Whole Genomes

Mercator	http://bio.math.berkeley.edu/mercator/
MAVID	http://bio.math.berkeley.edu/mavid/
RepeatMasker	http://www.repeatmasker.org/
WU-BLAST	http://blast.wustl.edu/
SNAP	http://homepage.mac.com/iankorf/
BLAT	http://www.cse.ucsc.edu/~kent/

We will begin by starting in an empty directory and creating subdirectories for the input and output files of the alignment process.

```
$ mkdir input
$ mkdir output
```

3.1. Obtaining Genome Sequences

Genome sequences can be obtained from many sources on the internet. Most sources are either genome sequencing centers or databases that collect from many primary sources. We will download the *D. melanogaster* release 4 and *D. yakuba* release 2 assemblies from a database site, the University of California Santa Cruz (UCSC) Genome Browser (4) (<http://genome.ucsc.edu>). The *D. ananassae* CAF1 assembly will be downloaded from the AAA *Drosophila* website (<http://rana.lbl.gov/drosophila>) (see **Note 1** for additional information on obtaining sequence from the UCSC Genome Browser).

```
$ cd input
$ # Define a variable for the UCSC URL
$ GOLDENPATH=http://hgdownload.cse.ucsc.edu/goldenPath/
$ # Download the DroMel genome from UCSC (39 MB)
$ wget $GOLDENPATH/dm2/bigZips/chromFa.zip
...
$ mv chromFa.zip DroMel.zip
$ # Download the DroYak genome from UCSC (49 MB)
$ wget $GOLDENPATH/droYak2/bigZips/chromFa.tar.gz
...
$ mv chromFa.tar.gz DroYak.tar.gz
$ # Download the DroAna genome from AAA (317 MB)
$ wget http://rana.lbl.gov/drosophila/cafl/dana\_cafl.tar.gz
...
$ mv dana_cafl.tar.gz DroAna.tar.gz
```

In the case that any of these assemblies are no longer found at the URLs previously cited, the author has placed copies of them at <http://bio.math.berkeley.edu/mercator/tutorial/>.

To check that the downloaded assemblies are valid and to get some basic statistics about them, we will use the `faCount`, `faLen`, and `stats` utilities (Mercator distribution). The `faCount` utility calculates nucleotide frequencies within each input FASTA record (chromosomes or contigs in our case) and

the `faLen` utility simply outputs the length of each sequence. Combining `faLen` with `stats`, which calculates some basic descriptive statistics of a set of numbers, allows us to calculate useful statistics for the draft assembly of *D. ananassae*.

```
$ unzip -p DroMel.zip | faCount
#seq len      A      C      G      T      N      cpG
chr4 1281640 415025 225495 224520 416500 100    40533
chrM 19517   8152   2003   1479   7883   0     132
chrU 8724946 1494654 978040 986285 1522538 3743429 186802
...
$ tar zxOf DroAna.tar.gz dana/scaffolds.bases | faLen | stats

      N = 13749
      SUM = 230993012
      MIN = 55
1ST-QUARTILE = 1191
      MEDIAN = 1517
3RD-QUARTILE = 3575
      MAX = 23697760
      MEAN = 16800.7136519
      N50 = 4599533
```

From the output of the last command, we see that half of the bases in the *D. ananassae* assembly are in scaffolds of length 4,599,533 or greater (this is the N50 statistic for a genome assembly).

3.2. Preparing the Genome Sequences

Unfortunately, it is often the case that two whole genome sequences downloaded from the Internet are in different formats, so some work must be done to prepare the sequences for alignment.

3.2.1. Masking Repeats

For the best genome annotations and alignments, the genome sequences must be “masked” for repeats. See **Note 2a** for details on the different ways in which a sequence can be masked. Fortunately for us, the sequences obtained from the UCSC Genome Browser website are already softmasked with RepeatMasker (5) and Tandem Repeats Finder (6). For the *D. ananassae* sequence, we will need to do the masking ourselves. We will use the RepeatMasker program, as well

as the `nmerge` (WU-BLAST distribution, often required by RepeatMasker) and `faSoftMask` (Mercator distribution) utilities.

```
$ # Extract sequence for DroAna (1 min)
$ tar zxOf DroAna.tar.gz dana/scaffolds.bases > DroAna.fa.unmsk
$ # Mask interspersed repeats (19 hours)
$ ln -s DroAna.fa.unmsk DroAna.fa.int
$ RepeatMasker -no_is -nolow -species drosophila DroAna.fa.int
RepeatMasker version open-3.1.5
Search engine: WUBlast
```

```
analyzing file DroAna.fa.int
identifying matches to drosophila genus sequences in batch 1 of
6036
```

```
...
$ # Mask low complexity repeats (13 hours)
$ ln -s DroAna.fa.unmsk DroAna.fa.low
$ RepeatMasker -no_is -noint -species drosophila DroAna.fa.low
RepeatMasker version open-3.1.5
Search engine: WUBlast
```

```
analyzing file DroAna.fa.low
identifying simple repeats in batch 1 of 6036
identifying more simple repeats in batch 1 of 6036
identifying low complexity regions in batch 1 of 6036
```

```
...
$ # Merge masking into one hardmasked file (1 min)
$ nmerge DroAna.fa.int.msk DroAna.fa.low.msk > DroAna.fa.msk
$ # Create softmasked file (2 min)
$ faSoftMask DroAna.fa.unmsk DroAna.fa.msk > DroAna.fa
```

3.2.2. Creating Sequence Database Files

For efficiency purposes, we need to put our FASTA-formatted sequences into another format. The author has developed a file format, the Sequence Database format (SDB), that allows for fast random access to multiple sequences stored in a single file. See **Note 2b** for descriptions of the command-line utilities available (as part of the Mercator distribution) for creating and accessing SDB files. We will use the `fa2sdb` utility to put our softmasked genomes into SDB format.

```
$ unzip -p DroMel.zip | fa2sdb -c DroMel.sdb
$ tar zxOf DroYak.tar.gz | fa2sdb -c DroYak.sdb
$ cat DroAna.fa | fa2sdb -c DroAna.sdb
```

To get a listing of the *D. melanogaster* chromosomes and their lengths, we can use the `sdbList` utility.

```
$ sdbList -l DroMel.sdb
chr2L 22407834
chr2R 20766785
chr2h 1694122
chr3L 23771897
...
```

To get the sequence from a specific genomic interval, we can use the `sdbExport` utility.

```
$ # Get sequence of 2nd coding exon of gene "dachshund"
$ sdbExport -r DroMel.sdb chr2L 16477453 16477480 -
>chr2L:16477453-16477480-
ATGCCTATCGATCAAGCCACCAGAAAG
```

3.3. Obtaining Gene Annotations

The simplest way to use Mercator for orthology map creation is to use coding exons as map anchors. Therefore, we need to obtain gene annotations for each of our genomes. For the *D. melanogaster* and *D. yakuba* genomes, we will simply download annotations. For the *D. ananassae* genome, we will have to produce our own annotations through the use of gene prediction software. See **Note 3** for tips on obtaining annotations and details on the annotation format required by Mercator.

First, download annotations for *D. melanogaster* and *D. yakuba* from the UCSC Genome Browser and convert them to GFF using the utility program `ucsc2gtf` (Mercator distribution).

```
$ # Obtain annotations for DroMel
$ wget $GOLDENPATH/dm2/database/flyBaseGene.txt.gz
...
$ zcat flyBaseGene.txt.gz | ucsc2gtf flybase > DroMel.gff
$ # Obtain annotations for DroYak
$ wget $GOLDENPATH/droYak2/database/genscan.txt.gz
...
$ wget $GOLDENPATH/droYak2/database/xenoRefGene.txt.gz
...
```

```
$ zcat genscan.txt.gz | ucsc2gtf genscan > DroYak.gff
$ zcat xenoRefGene.txt.gz | ucsc2gtf xenoRefSeq >> DroYak.gff
```

Notice that we have combined two independent annotations of *D. yakuba* into one GFF file. Users can use as many annotation sets as they like and, in fact, the more the better (sensitivity is all that matters).

Now we will generate an annotation of the *D. ananassae* genome using the SNAP (7) gene prediction program (wrapped by the runSnap script, Mercator distribution). The program zff2gtf (Mercator distribution) is used to convert from SNAP's ZFF format to GFF.

```
$ # Run SNAP with D. melanogaster parameters (2 hours)
$ runSnap /usr/local/snap/HMM/fly \
< DroAna.fa.int.msk > DroAna.zff
$ cat DroAna.zff | zff2gtf --source=SNAP > DroAna.gff
```

3.4. Generating Input for Mercator

With SDB and GFF files for each genome in hand, we are now ready to generate the input files for Mercator. The easiest way to do this is with the makeMercatorInput script (Mercator distribution). We simply supply the names of the assemblies as arguments to this script. The makeMercatorInput script will look in the current directory for each genome's SDB and GFF file. See **Note 4** for information regarding custom jobs with or without makeMercatorInput.

```
$ # Create input files for Mercator (15 min)
$ makeMercatorInput DroMel DroYak DroAna
Making chromosome file for DroMel...done
Making anchors for DroMel...done
Extracting protein sequences for anchors...done
Making chromosome file for DroYak...done
Making anchors for DroYak...done
Extracting protein sequences for anchors...done
Making chromosome file for DroAna...done
Making anchors for DroAna...done
Extracting protein sequences for anchors...done
BLATing anchors pairwise...
DroMel-DroYak
Loaded 10029188 letters in 98948 sequences
Searched 7247210 bases in 53254 sequences
...
```

This script performs the following tasks:

1. Creates a file for each genome specifying the names and lengths of the sequences that make up that genome.
2. Creates a set of nonoverlapping anchor intervals for each genome from the CDS records of the GFF files.
3. Creates a file for each genome of the protein sequences coded for by each of the anchor intervals.
4. Compares the protein sequences of each genome pairwise using the BLAT (8) program to create “hit” files.

Also required by some components of Mercator and by MAVID is a phylogenetic tree relating the input species. The branch lengths of the tree should be the expected number of substitutions per site along each branch. The tree must be in Newick format (<http://evolution.genetics.washington.edu/phylip/newicktree.html>). We will put our tree in the file `treefile`.

```
$ echo "((DroMel:0.1,DroYak:0.1):0.4,DroAna:0.6);" > treefile
```

3.5. Constructing an Orthology Map With Mercator

Running Mercator is simple and fast once all of the input files have been generated. Because the *D. ananassae* assembly is still in scaffolds, we will tell Mercator that it should be treated as a draft genome by using the `-d` flag.

```
$ cd ..
$ mercator -i input -o output DroMel DroYak -d DroAna
...
Loading input files...
Loading chromosome files...
DroMel 13 chromosomes
DroYak 21 chromosomes
DroAna 13749 contigs
Loading anchor files...
DroMel 53254 anchors
DroYak 98948 anchors
DroAna 89541 anchors
Loading hit files...
DroMel-DroYak 75082 hits (2380 filtered)
DroAna-DroMel 75397 hits (4355 filtered)
DroAna-DroYak 110120 hits (3324 filtered)
Sorting edges...
Time spent loading files: 16 seconds
Making map...
```

```

...
Assembling draft genomes...
Number of runs: 1177 (using 46614 cliques)
Checking cliques...
Map-making completed
Number of runs: 1177
Number of cliques: 46614
Mean run length: 39.6041
Median run length: 19
Max run length: 513
Min run length: 1
Coverage of DroMel anchors: 98.4133 % (52409/53254)
Coverage of DroYak anchors: 81.5964 % (80738/98948)
Coverage of DroAna anchors: 81.738 % (73189/89541)
Writing coverage files...
Coverage of DroMel: 82.3921 %
Coverage of DroYak: 69.2232 %
Coverage of DroAna: 58.1449 %
...
Run time: 38 seconds
$ # Mercator has finished, let us look at the output files
$ cd output
$ ls

DroAna.agp      DroMel.coverage  genomes
DroAna.anchors DroMel.mgr       map
DroAna.coverage DroYak.agp       pairwisehits
DroAna.mgr      DroYak.anchors  runs
DroMel.agp      DroYak.coverage  pre.map
DroMel.anchors  DroYak.mgr

```

After running the main Mercator program, we now have an orthology map where the orthologous intervals are defined by the boundaries of the landmarks in the file “pre.map” and a map with the breakpoint regions cut in half in the file “map.” See **Note 5** for more details on Mercator.

3.6. Comparatively Scaffolding Draft Genomes

When a genome is specified as “draft” to Mercator (using the -d option), the program will attempt to comparatively scaffold that genome’s component sequences. That is, it uses information from the other genomes to orient and join the draft genome’s contigs or scaffolds. Mercator specifies the comparative scaffolding of a draft genome in the form of an AGP file

(<http://www.ncbi.nlm.nih.gov/Genbank/WGS.agpformat.html>). Later steps in the alignment process will not be aware of comparative scaffolding, so we must provide updated SDB files for each genome. In our alignment, *D. ananassae* has been comparatively scaffolded by Mercator, so we must “assemble” its component sequences into a new SDB file using the `sdbAssemble` program (Mercator distribution). For the other genomes, we will simply make a link to original SDB files. See **Notes 6** for additional details on the comparative scaffolding aspect of Mercator.

```
$ sdbAssemble ../input/DroAna.sdb DroAna.sdb < DroAna.agp
$ ln -s ../input/DroMel.sdb
$ ln -s ../input/DroYak.sdb
```

3.7. Refining the Map via Breakpoint Finding

Because Mercator has only used exons as landmarks for determining orthologous segments, the exact boundaries of the orthologous segments are not yet determined. If we wish to refine the boundaries of the identified orthologous segments, we can use the breakpoint finding program included in the Mercator distribution. This program attempts to find the best position within each “breakpoint region” (intervals in between segments identified in the “pre.map”) at which to break and add the left and right intervals to the flanking segments. This procedure may be skipped if the exact boundaries of the segments are not required. Locating breakpoints involves a number of steps. Note that SDB files for each genome must be present in the current directory (output), as set up in the last section. See **Notes 7** for additional information on the breakpoint finding process.

```
$ # The breakpoint finding algorithm requires the tree
$ ln -s ../input/treefile
$ # Convert the orthology map into a more general homology map
$ omap2hmap genomes < pre.map > rough.homology.map
...
$ # Create the graph relating the breakpoint regions
$ makeBreakpointGraph rough.homology.map treefile
$ # Make pairwise alignments for breakpoint regions (2 hours)
$ mkdir bp_alignments
$ makeBreakpointAlignmentInput --out-dir=bp_alignments
$ mavidAlignDirs --init-dir=bp_alignments
$ # Find a good configuration of breakpoints (8 min)
$ findBreakpoints rough.homology.map treefile edges\
bp_alignments > breakpoints
```

```
$ # Refine the map by splitting the breakpoint regions
$ breakMap breakpoints < rough.homology.map > better.homology.map
$ # Convert back to the orthology map format
$ hmap2omap genomes < better.homology.map > better.map
```

3.8. Generating Input for MAVID

Now that we have an orthology map, we are ready to run a global multiple alignment program on each orthologous segment set identified by the map. To help in the alignment process, we will give the alignment program a set of “constraints:” short intervals within the orthologous segments that we know should be aligned. These constraints are derived from the sequence similarities identified between the anchors given to Mercator. To make the constraints file, we run the following command:

```
$ # Convert pairwise hits to alignment constraints (2 min)
$ phits2constraints -i ../input < pairwisehits > constraints
```

The input files for MAVID are then generated by `makeAlignmentInput`.

```
$ # Create directories and files for alignment (3 min)
$ mkdir alignments
$ makeAlignmentInput --map=better.map . alignments
```

See **Notes 8** for information on the input files that are required for MAVID and that are generated by `makeAlignmentInput`.

3.9. Aligning Orthologous Segments With MAVID

With the input for MAVID generated, all that is left is to run MAVID on the sequences for each orthologous segment set. Each segment set is stored in a separate subdirectory. This is a good step at which to parallelize, but if that is not an option, the `mavidAlignDirs` script (Mercator distribution) can be used. See **Note 9** for details on the nucleotide-level alignment step.

```
$ # Align all sequence files in directory structure (13 hours)
$ mavidAlignDirs --init-dir=alignments
```

We now have a multiple whole-genome alignment of *D. melanogaster*, *D. yakuba*, and *D. ananassae*.

3.10. Extracting Subalignments

We may now extract parts of the whole-genome alignment that are of particular interest using the `sliceAlignment` program (Mercator distribution). For example, we may wish to get the alignment of the second coding exon of the gene *dachshund*. The `sliceAlignment` program outputs alignments in multi-FASTA format, so we will use the `fa2clustal` utility (Mercator distribution) to put the exon alignment into a more readable form. See **Note 10** for more details on `sliceAlignment`.

```
$ sliceAlignment alignments \
DroMel chr2L 16477453 16477480 -> exon.mfa
$ fa2clustal < exon.mfa
CLUSTAL
```

```
DroMel   ATGCCTATCGATCAAGCCACCAGAAAG
DroYak   ATGCCTATCGATCAAGCCACCAGAAAG
DroAna   ATGCCTATCGATCAAGCCACCAGAGAG
*****
```

3.11. Concluding Remarks

This tutorial has taken the user through the basic steps of creating a multiple whole-genome alignment using Mercator and MAVID. There are many additional details and options that have been left out of this tutorial at each step. More details are available in the full documentation of each of the programs.

4. Notes

1. **Obtaining genome sequences.** To download genomes from the UCSC Genome Browser, it is easiest to go through the “Downloads” section of the website. For the assembly of interest, click on the “Full data set” link to access complete genome sequences as compressed FASTA files. Other databases from which to obtain genome assemblies are at the NCBI (<http://www.ncbi.nlm.nih.gov>) and Ensembl (<http://www.ensembl.org>).
2. **Preparing the genome sequences.**
 - a. **Masking repeats.** An “unmasked” FASTA formatted file has all characters in uppercase. A masked sequence can either be “hardmasked” or “softmasked.” In hardmasked files, characters that are part of repetitive sequence are changed to Ns, whereas in softmasked files they are changed to lowercase. Unmasked and

softmasked sequence may also have Ns, which are commonly used to indicate assembly gaps. Ideally, we would like our genome sequences to be softmasked, so that we have repeat annotations as well as full sequence information.

Masking repeats is a bit of an art, and the author will not go into all of the details here. Very briefly, one needs to mask both interspersed and simple (or low complexity) repeats. Masking of these two types of repeats should be done separately because gene finding is best done on sequence hardmasked for interspersed repeats (simple repeats can occur within genes).

- b. **Creating sequence database files.** There are four command-line utilities made available in the Mercator distribution for handling SDB files. The Mercator library code may also be used for writing C++ programs that access SDB files directly. The command-line utility `fa2sdb` is used to create or append to a SDB file from sequence records in FASTA format. DNA sequences may be compressed (two nucleotides per byte) inside of a SDB file if the `-c` option is specified. The `sdbExport` utility is used for the extraction of specific genomic intervals from a SDB file. It can extract one or more intervals at a time and outputs sequences in FASTA format. The `sdbList` utility is used to list the names and lengths (with the `-l` option) of the records inside of a SDB file. Lastly, the `sdb2fa` utility is used to convert a SDB file into FASTA format.
3. **Obtaining gene annotations.** Gene annotations for many genomes can be obtained at the same database sites that provide whole genome sequences. For the UCSC Genome Browser site, annotations can be obtained either through the “Table Browser,” or directly from the “Downloads” section. If annotations are not available online, you can produce them using gene prediction software. The easiest prediction programs to use in this case are single-genome *ab initio* gene finders (e.g., `geneid` [9], `GENSCAN` [10], and `SNAP` [7]). Regardless of how the annotations are obtained, they need to be converted to the GFF format (<http://www.sanger.ac.uk/Software/formats/GFF/>). Three scripts (`genscan2gtf`, `ucsc2gtf`, and `zff2gtf`) in the Mercator distribution are available for converting to GFF from some common formats. Mercator requires that GFF annotations have CDS records (lines with “CDS” in the feature field) for the coding intervals of each exon. It is critical that the “frame” field be specified for each CDS record in the GFF files. This field allows Mercator to translate each coding exon correctly.
4. **Generating input for Mercator.** For custom jobs (e.g., to parallelize some tasks), you may wish to generate the input for Mercator without using the `makeMercatorInput` script. In such cases, consult the `README` file in the Mercator distribution for exact specifications of the various input files that are required. Some routines of `makeMercatorInput` are customizable via command-line options. Use the `--help` option to get full usage information.
5. **Constructing an orthology map with Mercator.** Mercator has a number of user-settable parameters that may be specified as command-line options.

The options that affect Mercator's performance are `--min-run-length`, `--prune-pct`, `--join-distance`, `--max-eval`, `--repeat-num`, and `--repeat-pct`. Consult the Mercator README file for descriptions of these options.

6. **Comparatively scaffolding draft genomes.** When Mercator comparatively scaffolds the components of a “draft” genome, it joins components that it believes should be adjacent to each other into new sequences with names beginning with `assembled`. For example, in our fruit fly alignment, the `scaffold_13770`, `scaffold_13165`, and `scaffold_13337` sequences from the *D. ananassae* assembly are joined into a new sequence called `assembled6`, with a string of Ns separating the component sequences within `assembled6`. The number of separating Ns may be specified by Mercator's `--padding` command-line option. These Ns are meant to indicate gaps of unknown length between the component sequences.
7. **Refining the map via breakpoint finding.** The breakpoint finding process can be very computationally intensive, depending on the input genomes. If a cluster is available to the user, it is a good idea to parallelize the `mavidAlignDirs` step. When running the `findBreakpoints` program, accuracy may be traded for speed via the `--resolution` option. Breakpoints will be found more accurately with larger “resolution” values.
8. **Generating input for MAVID.** MAVID requires, at a minimum, three input files. These files are a phylogenetic tree in Newick format, unmasked sequences in a multi-FASTA file, and a hardmasked version of the multi-FASTA file. When Mercator is used, alignment constraints may be given to MAVID via the `-c` command-line option. In this tutorial, the `makeAlignmentInput` and `mavidAlignDirs` take care of generating and passing the correct files to MAVID.
9. **Aligning orthologous segments with MAVID.** Although the focus of this tutorial is on the application of Mercator and MAVID, the hierarchical strategy for whole-genome alignment allows for the components to be substituted with similar programs independently of each other. For example, in cases where the orthologous segments are very small, CLUSTAL W (II) could be used to do the multiple nucleotide alignment instead of MAVID. However, there is a significant advantage to using MAVID as the nucleotide-level aligner with Mercator: alignment constraints. By using the alignment constraints output by Mercator, MAVID can more accurately align coding regions and is able to process longer sequences.
10. **Extracting subalignments.** The `sliceAlignment` program is designed to efficiently extract subalignments from a multiple whole-genome alignment. It extracts alignments based on the coordinates given as input for a specified reference genome. A single interval may be given as command-line arguments or multiple

intervals can be given on the standard input. With multiple intervals as input, the program will be very efficient if the intervals are sorted by their start coordinates.

References

- 1 Dewey, C. N. (2006) Whole-genome alignments and polytopes for comparative genomics. Ph.D. thesis, University of California, Berkely.
- 2 Bray, N. and Pachter, L. (2004) MAVID: constrained ancestral alignment of multiple sequences. *Genome Res.* **14**, 693–699.
- 3 Dewey, C. N. and Pachter, L. (2006) Evolution at the nucleotide level: the problem of multiple whole-genome alignment. *Hum. Mol. Genet.* **15**, R51–R56.
- 4 Karolchik, D., Baertsch, R., Diekhans, M., et al. (2003) The UCSC Genome Browser Database. *Nucleic Acids Res.* **31**, 51–54.
- 5 Smit, A. F., Hubley, R., and Green, P. (1996–2004) RepeatMasker Open-3.0. <http://www.repeatmasker.org>.
- 6 Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.* **27**, 573–580.
- 7 Korf, I. (2004) Gene finding in novel genomes. *BMC Bioinformatics* **5**, 59.
- 8 Kent, W. J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.* **12**, 656–664.
- 9 Guigo, R. (1998) Assembling genes from predicted exons in linear time with dynamic programming. *J. Comput. Biol.* **5**, 681–702.
- 10 Burge, C. and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* **268**, 78–94.
- 11 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.

Mulan

Multiple-Sequence Alignment to Predict Functional Elements in Genomic Sequences

Gabriela G. Loots and Ivan Ovcharenko

Summary

Multiple sequence alignment analysis is a powerful approach for translating the evolutionary selective power into phylogenetic relationships to localize functional coding and noncoding genomic elements. The tool Mulan (<http://mulan.dcode.org/>) has been designed to effectively perform multiple comparisons of genomic sequences necessary to facilitate bioinformatic-driven biological discoveries. The Mulan network server is capable of comparing both closely and distantly related genomes to identify conserved elements over a broad range of evolutionary time. Several novel algorithms are brought together in this tool: the tba multisequence aligner program used to rapidly identify local sequence conservation and the multiTF program to detect evolutionarily conserved transcription factor binding sites in alignments. Mulan is integrated with the ERC Browser, the UCSC Genome Browser for quick uploads of available sequences and supports two-way communication with the GALA database to overlay GALA functional genome annotation with sequence conservation profiles. Local multiple alignments computed by Mulan ensure reliable representation of short- and large-scale genomic rearrangements in distant organisms. Recently, we have also introduced the ability to handle duplications to permit the reliable reconstruction of evolutionary events that underlie the genome sequence data. Here, we describe the main features of the Mulan tool that include the interactive modification of critical conservation parameters, visualization options, and dynamic access to sequence data from visual graphs for flexible and easy-to-perform analysis of differentially evolving genomic regions.

Key Words: Multiple alignment; alignment tool; evolutionary conservation; conserved elements; conserved transcription factor binding sites.

1. Introduction

It has been determined that blocks of evolutionary conservation identified through cross-species comparisons correlate with functional DNA segments such as protein coding genes (1,2) and transcriptional regulatory elements (3,4). Several available web-based tools implement multiple sequence analysis either as a series of pairwise alignments with a selected reference sequence (5–7) or as a full multisequence global or pseudo-global alignment (8–12). Applications of these tools differ by the type of sequences (nucleotide or amino acid) they are capable of processing, as well as by the maximum length and number of allowable input sequences.

The Mulan alignment engine consists of several data analysis and visualization schemes for high-throughput identification of functional sequences conserved across large evolutionary distances. Mulan (1) determines phylogenetic relationships among the input sequences and generates phylogenetic trees, (2) constructs graphical and textual alignments, (3) dynamically detects evolutionary conserved regions (ECR) in alignments, and (4) presents users with dynamic visual display options for flexible generation of conservation profiles. In addition, this tool is capable of implementing the phylogenetic shadowing strategy for identifying slow-mutating elements in comparisons of multiple closely related species (11). Alignments generated by the Mulan tool can be directly processed by the MultiTF program to identify evolutionarily conserved transcription factor binding sites (TFBS) shared by all analyzed species. This feature allows users to derive useful information toward decoding the sequence structure of regulatory elements that are functionally conserved among different species. Mulan is publicly available at <http://mulan.dcode.org>.

2. Methods

2.1. Alignment Strategy

Mulan provides two complementary alignment strategies for performing comparative sequence analysis of multiple sequences that are either (1) “finished” or (2) “draft” quality. The first approach operates with multiple high quality single-contig (finished) sequences, whereas the second method allows the construction of an alignment of multiple draft-quality sequences to a base (or reference) finished-quality sequence by effectively ordering-and-orienting draft sequences based on homology to the base sequence. Genomic sequences submitted to Mulan are aligned by the tba program (13) for “finished” sequences and by the refine program for “draft” sequences. The local alignment approach utilized for both sequence types reassures reliable representation of inversions and genomic reshuffling

events that have occurred in a subset of lineages since the last common ancestor. It is important to mention that colinearity between input sequences (as in the case of a global alignment) is not required. Mulan generates different projections of the “threaded block-set alignment or tba” to different reference sequences that are selected by the user to ensure the detection of evolutionarily conserved elements throughout the alignment in the event orthologous regions have been repositioned or inverted in a subset (*see* **Note 1**).

2.2. Generating Alignments

1. Access Mulan via the internet at <http://mulan.dcode.org/> (**Fig. 1A**). Alternatively access Mulan via the ECR Browser at <http://ecrbrowser.dcode.org>, through the “Synteny/Alignments” link. Click on each box next to the sequence to be aligned and then click on the “Mulan” button provided at the bottom of the page (**Fig. 1B**).
2. At the Mulan homepage, indicate the number of species that will be used in the analysis, and select the desired alignment type: tba-based (left button) or refine-based (right button) (**Fig. 1A**). (It is advised to select the tba-based approach if the user is unsure of which option is best suited, or have sequences in single-contig format. The tba-method includes more options and provides more sensitive alignments than refine.)
3. Sequence input.
 - a. Submit sequences in FASTA format and gene annotation in format described in the Mulan documentation, and select the appropriate option for masking repetitive elements (**Fig. 2A**). Although Mulan is capable of running RepeatMasker locally (<http://www.repeatmasker.org/>) to mask repetitive elements in input sequences, submitting premasked sequences will significantly reduce the total processing time.
 - b. If sequences of interest are available from fully sequenced genomes, Mulan can automatically fetch these sequences from the University of California Santa Cruz (UCSC) Genome Browser (<http://genome.ucsc.edu/>). To do so, the user needs to click “Upload” (**Fig. 2A**), and provide the necessary information for the automated upload feature to fetch the sequences directly from the UCSC Genome Browser (**Fig. 2B**). The required information includes: (1) the organism or genome to be used, (2) the assembly version, (3) the type of annotation, and (4) genome coordinates. Once Mulan downloads the sequence along with its annotation onto the server, the successful upload is acknowledged (**Fig. 2C**), and the alignment engine proceeds to create alignments between the input sequences. Note that the upload feature automatically extracts information on repetitive elements along with the sequence data; it also permits selection of different gene annotation sources. This automated upload can be combined with manual input of sequences missing representation in fully sequenced genomes.

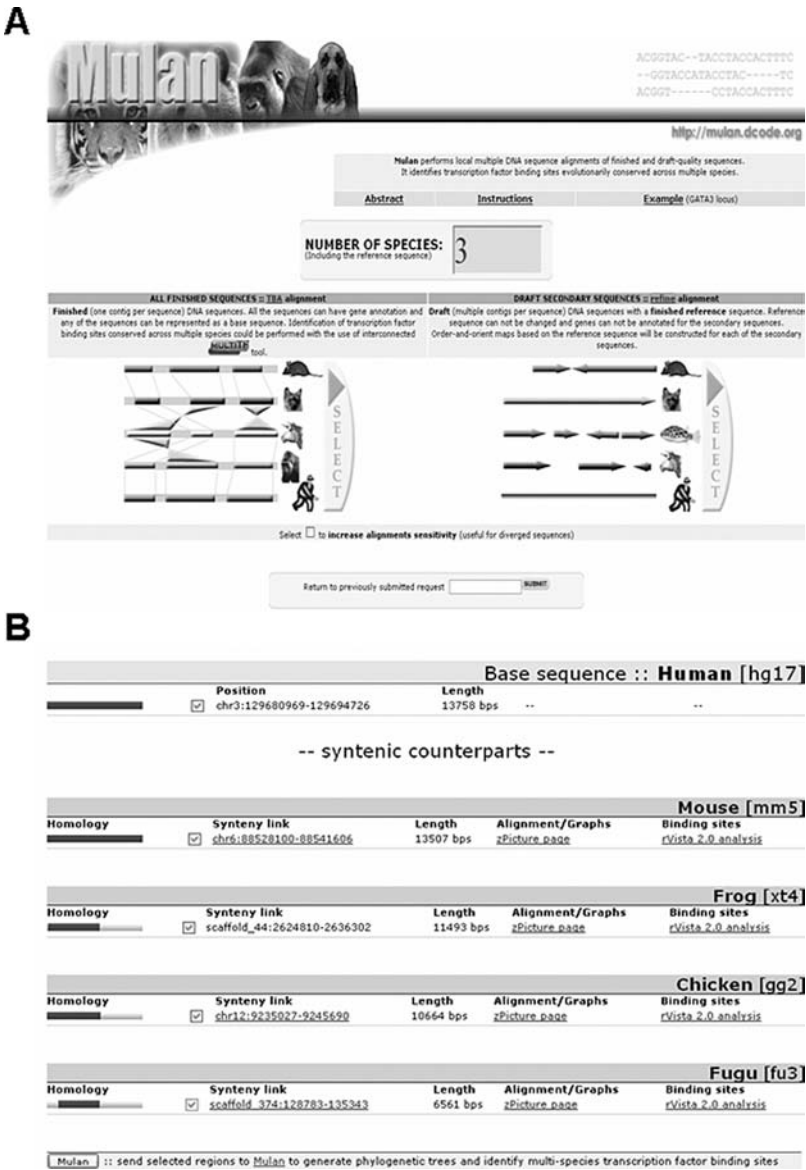


Fig. 1. Accessing the Mulan tool from the homepage at <http://mulan.dcode.org> (A) or from the ECR Browser “Synteny/Alignments” link (B).

- c. Alternatively, if all the sequences of interest are available from fully sequenced genomes, the “Batch Upload System or BUS” integrated into Mulan can be used to simultaneously fetch all the sequences at once. Follow the BUS link on top of the sequence upload page to access this feature (Fig. 2A).

A

☒ Batch Upload System
Concurrently uploads ALL the sequences from the UCSC Genome Browser

SEQUENCE 1
Upload sequence and gene annotation from
UCSC Genome Browser

- or -

☒ Paste sequence
(in FASTA format @)

- or -

☐ FASTA file (.fa)

- or -

☐ NCBI accession #

OPTIONAL :: ANNOTATION 1

Repeats:

☐ Repeats are identified by lower-case letters

☒ Mask repetitive elements | no masking ☒

Gene annotation (if any): @

☒ Paste

☐ File

B

Step 1. Select a specie, genomic interval and genome assembly freeze from the UCSC Genome Browser (link will open in a new window)

Step 2. Describe selected region using the form below

Organism

Assembly

Annotation

Position (Format: chr7:1000-2000)

Step 3. Submit your request for verification

C

SEQUENCE 1
Sequence accepted
seq1.fa 170742 bps

☒

[Delete it](#)

OPTIONAL :: ANNOTATION 1

Repeats:

☒ Repeats are identified by lower-case letters

☐ Mask repetitive elements | human ☒

Gene annotation (if any): @

☒ Paste

☐ File

Fig. 2. Each sequence can be pasted in, in FASTA format, uploaded as a FASTA file, or entered as an accession number along with the available annotation (A). Alternatively, sequences can be fetched from the UCSC Genome Browser individually using the “Upload” function (A), or in groups (Batch Upload System) Browser (B). Once sequences have been uploaded, the program acknowledges the receipt (C).

4. Step-by-step specifics for the tba-based alignment approach. Upon generating a set of preliminary pairwise alignments, a phylogenetic tree is presented to the user, who has the option to accept it by clicking the “Continue” button, or refine it, if it is believed that the tree does not accurately depict the relationship between the input sequences (Fig. 3). This phylogenetic tree will be guiding the construction of the full tba-based multiple-sequence alignment.
5. Once the alignment request is completed, Mulan presents results data analysis on an interactive summary page (Fig. 4). The summary page consists of multiple links to the dynamic conservation profile visualization module, textual multiple sequence alignment (with a dynamically modified base sequence; specific to tba-based alignments), hot-link to multiTF detection of evolutionary conserved TFBS (specific to tba-based alignments), dot-plots describing sequence rearrangements, interactive selection of ECRs, etc. One also has the option to adjust annotation files and sequence titles from this page.
6. The processing information is stored on our servers for a limited amount of time (usually up to 3 months) and the data can be reaccessed anytime from the homepage (Fig. 1A) by providing the job identification number (request ID) listed on the top left corner of the summary page.

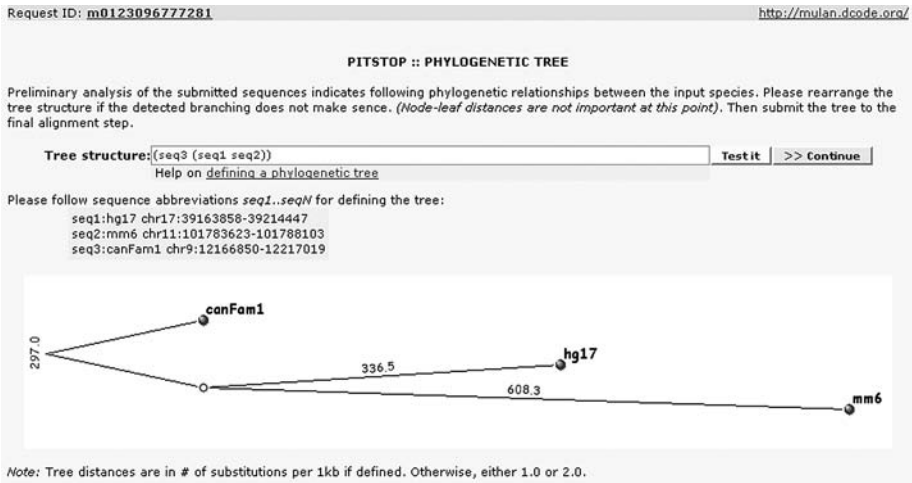


Fig. 3. Mulan defines a guiding phylogenetic tree before proceeding with the detailed sequence alignment. The user has the option to submit modifications to this tree.

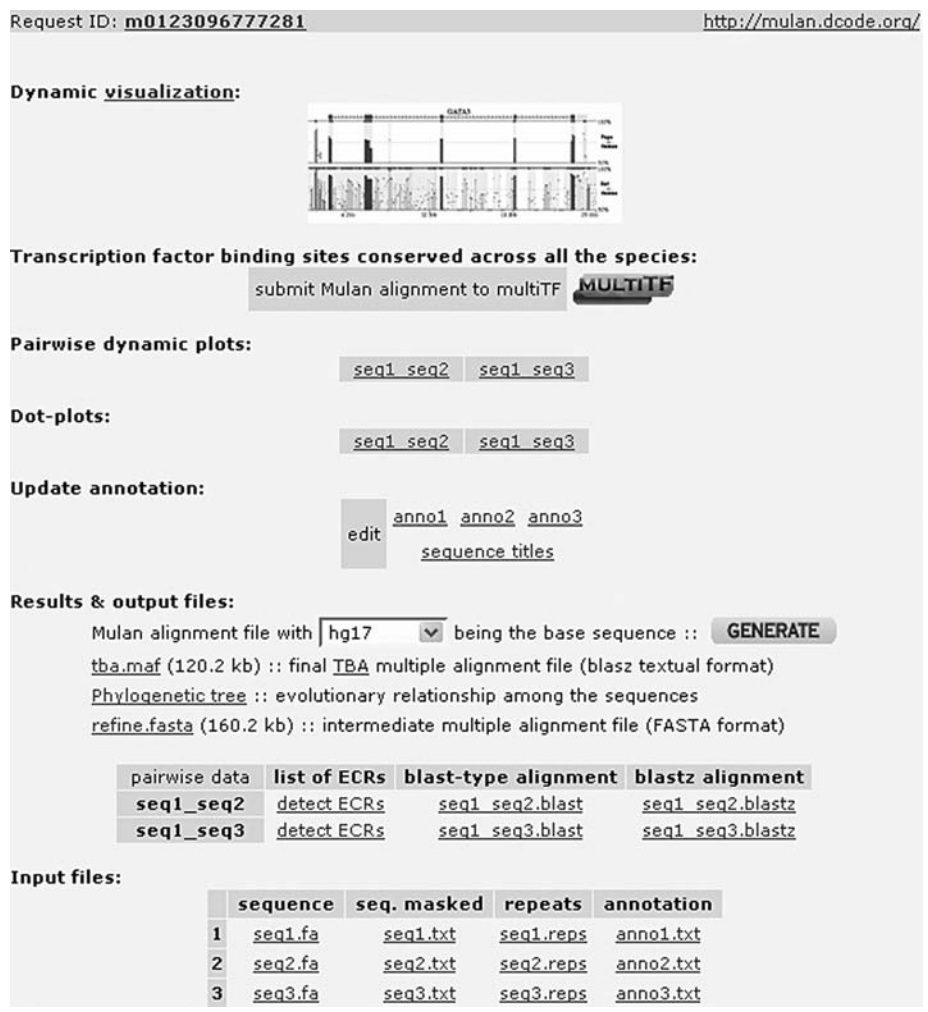


Fig. 4. A completed alignment request results in a “summary page” that provides links to the interactive visualization platform, pairwise dynamic plots, dot plots, annotation files, sequence files, and a portal to the transcription factor binding site analysis software, MultiTF.

2.3. Visualization and Data Analysis Strategies for Multisequence Local Alignments

Multiple-sequence comparative analysis is a challenging task in terms of generating highly reliable alignments and graphically displaying the alignment

results. To address the complexity stemming from user input sequence files that potentially consist of a large number of sequences of varying lengths and different phylogenetic relationships, we provide a set of different visualization options in *Mulan*. In general, *Mulan* alignment visualization is based on the *zPicture* display design (6), where the reference sequence is linear along the horizontal axis and the percent identity is plotted along the vertical axis. All the dynamic visualization options can be accessed through the summary page (Fig. 4). The “Dynamic Visualization” link directs the user to the interactive alignment display (Fig. 5). At this page the top bar (Fig. 5A) allows the user to customize the visual display by selecting the desired:

- 1. The Graphical type of evolutionary conservation profile (smooth or percent identity plot).
- 2. The length of the sequence to be displayed per each line.
- 3. The size and percent identity of the ECR to be highlighted in the graphical alignment display.
- 4. The percent identity for the bottom cut-off.
- 5. The subregion to be indicated as “from” – “to” coordinates.

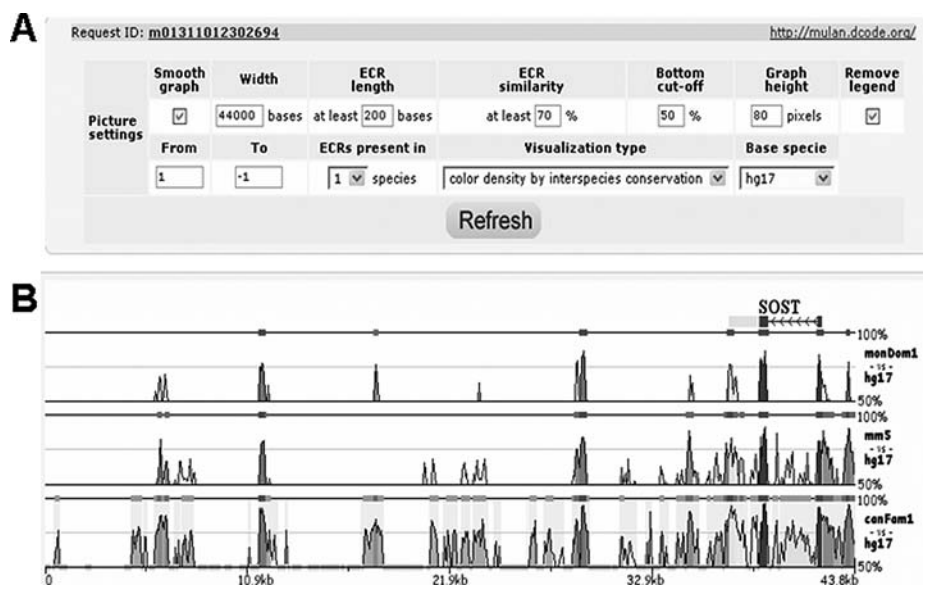


Fig. 5. *Mulan* interactive alignment customization options (A) and graphical display of alignments (B).

To assist in the visual analysis of conservation, Mulan has several additional options available.

1. The user can choose to color code ECRs that are present in a particular number of species (**Fig. 5**). This option will dynamically prioritize regions with variable degree of phylogenetic occurrence (*see Note 2*).
2. The user has the option to change the base genome in the visualization of multi-species sequence evolution. This provides the option to study conservation of regions specific to different lineages and closely related groups of species. By changing the base species, the new stacking order of conservation profiles with the rest of the species will be automatically determined using the evolutionary relationship of each sequence to the reference sequence, where more closely related species are at the bottom. (Option specific to the tba-based alignment.)
3. Visualization scheme provides the means to include or remove the legend in the display as well as to adjust the graph height.
4. Contig names and alignment blocks can be visualized as tracks on top of the conservation profile (**Fig. 6**). In this situation, syntenic blocks are color-coded based on their orientation in respect to the base sequence thus allowing for easy ordering-and-orienting of draft sequences by using the base sequence as the architectural guide. This feature can be used as a preassembly tool when multiple overlapping contigs are available from a homologous interval in a new species with detectable sequence similarity to the base sequence. (Option specific to the refine-based alignment.)
5. “Color density by interspecies conservation” illustrates the relationship between a conserved element and the number of species that share a particular region (**Fig. 7A**) such that, the more species share a sequence, the darker the conservation profile will be displayed. (This analysis is performed for every pixel-wide region of the conservation plot. The number of ECRs from different species that overlap with a particular pixel count toward the number of species sharing this region.)
6. Similar to Picture, Mulan allows interactive and customized ECR analysis. Users can select the evolutionary criteria (length and percent identity) for graphical identification of ECRs from the conservation plot. We have previously shown that longer

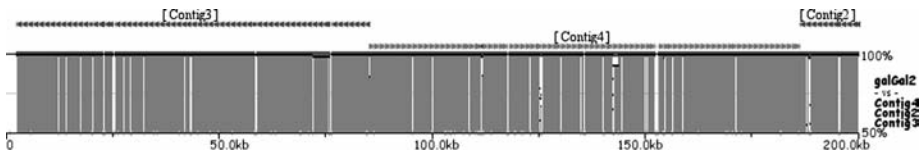


Fig. 6. Contig ordering based on homology to the reference sequence. The top layer of shaded lines indicates the location of contigs from a second sequence aligned to the base sequence where right-turned triangles specify forward strand alignments, and left-turned triangles correspond to reverse strand alignments.

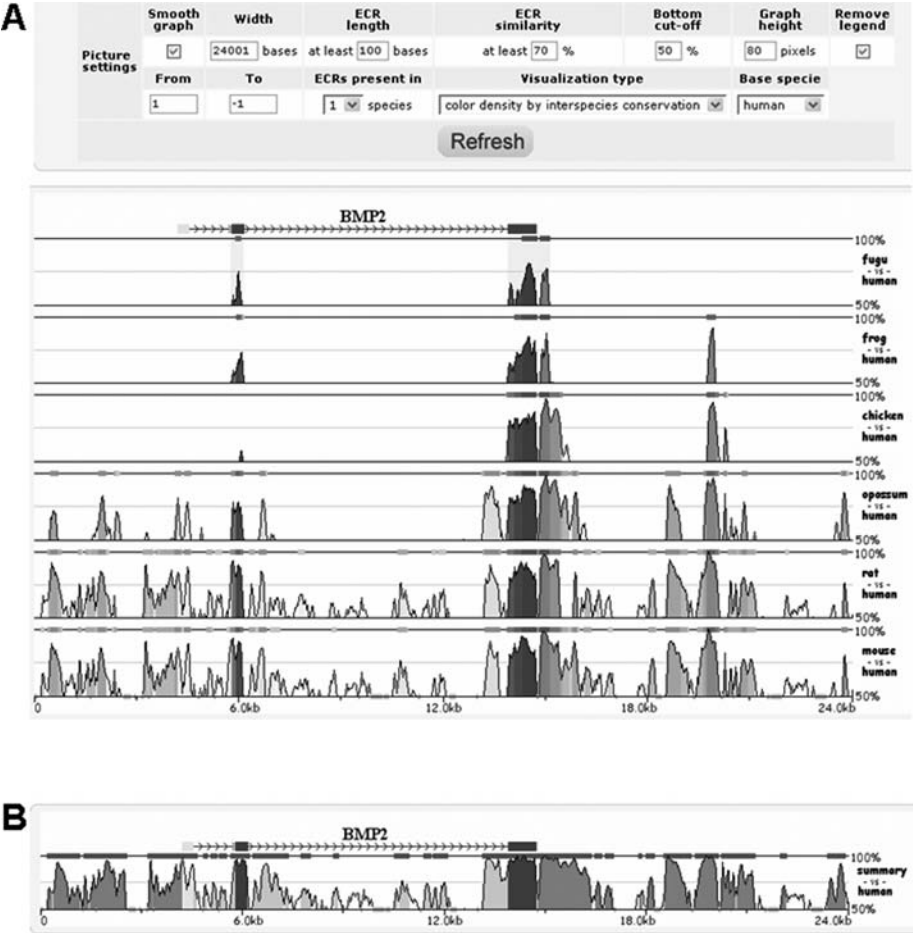


Fig. 7. Mulan alignment analysis options: color density by interspecies conservation (A) and summary of conservation display (B).

and well conserved ECRs can be indicators of functional elements in genomic alignments (14) and this option permits the user to prioritize and define the optimal amount of ECRs in the studied locus—to adjust for highly conserved vs poorly conserved loci.

- Two additional data representation modules are implemented in the Mulan tool: phylogenetic shadowing and summary of conservation. Summary of conservation collects shared similarities from all the pairwise comparisons into a single conservation profile (Fig. 7B), the phylogenetic shadowing option effectively collects

pairwise mismatches (11). Thus, the summary of conservation option will aid in reconstructing conservation profiles in cases of highly diverged sequences, whereas the phylogenetic shadowing option will facilitate the identification of the most conserved elements in alignments of closely related species with a limited number of mismatches (see Note 3).

2.4. Multisequence Conservation of TFBS

The ability to accurately predict active TFBS is a powerful approach for sequence-based discovery of gene regulatory sequences and for elucidating gene regulatory mechanisms (see Note 4). To combat the overabundance of false-positive computational predictions stemming predominantly from the small size of TFBS footprints and from poorly defined position weight matrices (PWM), evolutionary sequence analysis has been proposed as a robust strategy for filtering out false-positive sites (15–18). Methodologically, multiTF is similar to the rVista 2.0 tool (16,17), but implements a different strategy of detecting TFBS present in a multiple alignment. rVista 2.0 works only with pairwise sequence alignments, and requires each site to be present in a short island of high sequence conservation. In contrast, multiTF does not rely on preferential local conservation of functional binding sites vs neutrally evolving background as rVista does, instead it requires a binding site to be present in all the species at the same position as dictated by the alignment. Putative TFBS are identified in all the input sequences by using TRANSFAC PWM matrices to define consensus sequences and the tfSearch utility is used to map these consensus sequences to the genomic sequence of each input species (17,19). MultiTF excludes all TFBS predictions that overlap with exons. Gene annotation for only one of the sequences (the reference sequence) is sufficient to carry out this step. In the final step, multiTF detects TFBS predictions that are shared by all the species and are located at the same position as defined by the alignment. This is achieved by scanning through all the “anchors” or fully conserved nucleotide blocks (nucleotides that are identical in all species in the multiple-sequence alignment; Fig. 9B). If a TFBS from the reference sequence is found to overlap with an “anchor” nucleotide, we project this TFBS position to all the other species by using the alignment and excluding gaps (Fig. 9B). Starting and ending positions of the footprint of the reference sequence TFBS are compared to the starting and ending position for the same TFBS on the same strand as detected by the initial TFBS annotation. If corresponding TFBS can be identified in all the species in the alignment, this is reported by the multiTF.

MULTIF

TRANSSCRIPTION FACTORS

Total number of transcription factor families: 436

SELECT TRANSCRIPTION FACTORS

SELECT SEPARATE TRANSCRIPTION FACTORS

A

☐ TRANSEAC professional library
 Biological species
☒ vertebrates ☐ insects
☐ plants ☐ fungi
☐ nematodes ☐ bacteria
 Matrix similarity
☒ Optimized for function
☐ Predefined as 0.85
 Matrix selection
☐ use only high-specificity matrices
☐ User-defined consensus sequences

SUBMIT

B

Request ID: mlr01302006102923345

Summary:
 1 multi-conserved transcription factor binding sites (TFBS) were identified

Dynamic visualization:
 Dynamically overlay TFBS prediction with the conservation profile and perform clustering

Alignment:
 Highlight TFBS positions in the alignment

40 50

TATAAGAGATAAATCTATT

:: | | | | | | | | | | | | | |

XT--GAGATAAATCTAAAG

} 60

Additional data:

Index	TF binding sites	sequence	gene annotation	sequence name
sequence #1	218 TFBS	seq1.fa	anno1.txt	hg17 chr17:39163858-39214447
sequence #2	40 TFBS	seq2.fa	anno2.txt	mm6 chr11:101783623-101788103
sequence #3	279 TFBS	seq3.fa	anno3.txt	canFam1 chr9:12166850-12217019

Run multif using different parameters

Questions or comments?
 Email Ivan Ovcharenko at ovcharenko@linl.gov

MULTIF

Fig. 8. MultiTF portal available from Mulan “summary page.” First menu allows users to define the types of transcription factor binding site matrices to be used in the

To analyze Mulan alignments for the presence/absence of conserved TFBS shared among all provided, sequences the user needs to follow these steps:

1. Click on the multiTF button on the summary page (**Fig. 4**) to forward the alignments to the multiTF program (**Fig. 8A**).
2. Upon forwarding to the multiTF analysis initiation page, the user selects from methods and parameters to identify TFBS in individual sequences. First, the user has to choose between the use of the TRANSFAC database of TFBS (<http://www.biobase.de/>) or user-defined consensus sequences (**Fig. 8A**).
3. Assuming the most common use of TRANSFAC PWM matrices in description of TFBS binding specificities to scan for binding sites, the user selects the appropriate library of phylogenies (including vertebrate, plant, fungi, nematodes, insects, and bacteria).
4. Two different options are available for detecting TFBS through the use of TRANSFAC libraries. The default option is to use the “optimized for function” search option, which weights individual PWM matrices differently by minimizing and balancing out the abundance of false-negative hits from different matrices. The alternative option is to manually specify matrix similarity cut-off for the annotation of candidate TFBS. The “optimized for function” option utilizes different cut-off parameters for different TFBS, such that no more than three TFBS per 10 kb are predicted in a random sequence (20). Manually selected cut-offs measure sequence similarity to TRANSFAC PWM; the higher the cut-offs are, the fewer sites are predicted.
5. The final option permits the selection of only “high-specificity” matrices in the TFBS annotation. This option subselects a list of TFBS matrices that have ≤ 0.85 cut-off similarity to the TRANSFAC PWM. These are the matrices with the most reliable definitions in the TRANSFAC database.
6. Upon submitting a request, the user is directed to a page that lists all the available transcription factor families alphabetically, where one has to choose the matrices to be used for the analysis by clicking on the provided boxes (**Fig. 8B**). Alternatively, the user can “select all” to obtain a full repertoire of conserved TFBS.
7. A summary page will comprehensively display the results of the TFBS analysis (**Fig. 8C**). Here, users can access position and matrix information provided for each sequence independently, as well as the sites can be visualized “on top of” the alignment and used in subsequent clustering analysis (**Figs. 8C and 9A**). The clustering options are similar to the ones available for the rVISTA 2.0 tool, TFBS can be clustered “individually” or “combinatorially” and the sites can be visualized



Fig. 8. analysis, along with similarity thresholds (A). Next, the user selects either all or a subset of available transcription factors from the TRANSFAC library (B). Results are provided on an interactive summary page.

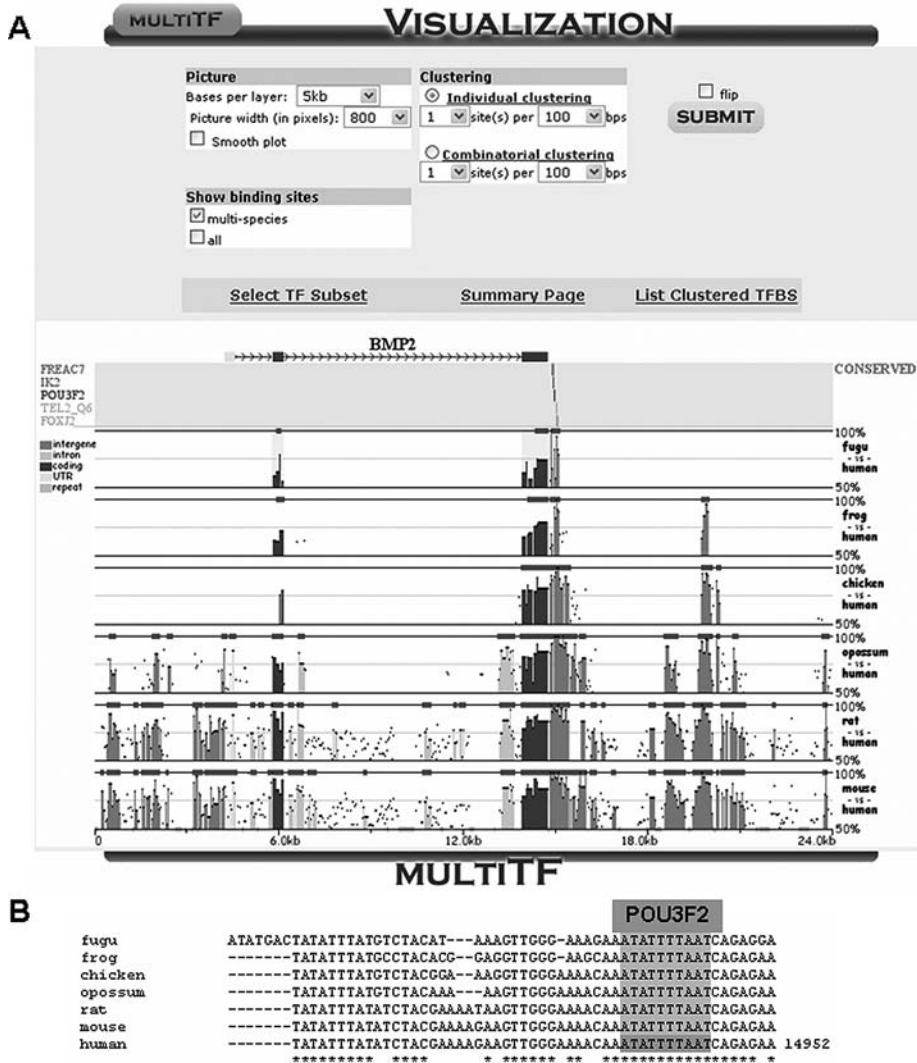


Fig. 9. Transcription factor binding sites can be juxtaposed on the Mulan multiple alignment graphics and several clustering and visualization options are provided for customized analysis (**A**). Similarly, the binding sites can be visualized within the textual alignment (**B**).

as a summary of conservation (show binding sites by multispecies) or in each sequence individually (show all) ([Fig. 9A](#)).

2.5. *Mulan-GALA Interconnection and Finding Orthologous Regions*

The database of genomic DNA sequence alignments and annotations (GALA) allows users to find genomic intervals that meet defined conservation thresholds, alignment-based scores, and gene annotation criteria, TFBS patterns, expression profiles, and other features ([21](#)). Once a region of interest has been found, a user may wish to examine it using the Mulan tool. Likewise, once an ECR element has been identified by using Mulan, users have the option to utilize GALA to find additional information about the region containing it. Thus, two-way data flow has been established between the GALA database and the Mulan server. The interconnection link of GALA to Mulan is established through forwarding a list of homologous regions in different species from GALA to Mulan. Once a DNA interval is specified in GALA, the user can easily access a page to find estimated orthologous positions in other species.

3. Notes

1. The Mulan tool is capable of producing fast and accurate multiple alignments for both distantly and closely related organisms, properly taking into account the complexity of evolutionary sequence rearrangements such as inversions, transpositions, and reshuffling.
2. Mulan provides users with a versatile visualization platform that allows interactive manipulation of both textual alignments and graphical conservation displays to differently address the conservation structure of either closely or distantly related species. In particular, the option to color conserved regions using a gradient based on the depth of conservation, coupled with a module that filters out ECRs that are shared by a requested number of species, permits the user to control the type of analysis performed to identify elements shared by a subset of input sequences.
3. Mulan is capable of handling large genomic sequences within minutes of processing time (up to megabases in length).
4. The dynamic interconnection of Mulan with multiTF presents an effective way to identify TFBS shared by multiple species. In combination, these tools can be used to predict and prioritize functional elements in otherwise anonymous sequences, a method that has been shown to be highly effective in identifying novel genes and regulatory sequences.

References

- 1 Pennacchio, L. A., Olivier, M., Hubacek, J. A., et al. (2001) An apolipoprotein influencing triglycerides in humans and mice revealed by comparative sequencing. *Science* **294**, 169–173.
- 2 Gilligan, P., Brenner, S., and Venkatesh, B. (2002) Fugu and human sequence comparison identifies novel human genes and conserved non-coding sequences. *Gene* **294**, 35–44.
- 3 Elnitski, L., Li, J., Noguchi, C. T., Miller, W., and Hardison, R. (2001) A negative cis-element regulates the level of enhancement by hypersensitive site 2 of the beta-globin locus control region. *J. Biol. Chem.* **276**, 6289–6298.
- 4 Loots, G. G., Locksley, R. M., Blankespoor, C. M., et al. (2000) Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science* **288**, 136–140.
- 5 Mayor, C., Brudno, M., Schwartz, J. R., et al. (2000) VISTA: visualizing global DNA sequence alignments of arbitrary length. *Bioinformatics* **16**, 1046–1047.
- 6 Ovcharenko, I., Loots, G. G., Hardison, R. C., Miller, W., and Stubbs, L. (2004) zPicture: dynamic alignment and visualization tool for analyzing conservation profiles. *Genome Res.* **14**, 472–477.
- 7 Schwartz, S., Zhang, Z., Frazer, K. A., et al. (2000) PipMaker: a web server for aligning two genomic DNA sequences. *Genome Res.* **10**, 577–586.
- 8 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 9 Bray, N., Dubchak, I., and Pachter, L. (2003) AVID: A global alignment program. *Genome Res.* **13**, 97–102.
- 10 Brudno, M., Do, C. B., Cooper, G. M., et al. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 11 Ovcharenko, I., Boffelli, D., and Loots, G. G. (2004) eShadow: a tool for comparing closely related sequences. *Genome Res.* **14**, 1191–1198.
- 12 Schwartz, S., Elnitski, L., Li, M., et al., and NISC Comparative Sequencing Program. (2003) MultiPipMaker and supporting tools: Alignments and analysis of multiple genomic DNA sequences. *Nucleic Acids Res.* **31**, 3518–3524.
- 13 Blanchette, M., Kent, W. J., Riemer, C., et al. (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* **14**, 708–715.
- 14 Ovcharenko, I., Stubbs, L., and Loots, G. G. (2004) Interpreting mammalian evolution using Fugu genome comparisons. *Genomics* **84**, 890–895.
- 15 Aerts, S., Thijs, G., Coessens, B., Staes, M., Moreau, Y., and De Moor, B. (2003) Toucan: deciphering the cis-regulatory logic of coregulated genes. *Nucleic Acids Res.* **31**, 1753–1764.

- 16 Loots, G. G., Ovcharenko, I., Pachter, L., Dubchak, I., and Rubin, E. M. (2002) rVista for comparative sequence-based discovery of functional transcription factor binding sites. *Genome Res.* **12**, 832–839.
- 17 Loots, G. G. and Ovcharenko, I. (2004) rVISTA 2.0: evolutionary analysis of transcription factor binding sites. *Nucleic Acids Res.* **32**, W217–W221.
- 18 Lenhard, B., Sandelin, A., Mendoza, L., Engstrom, P., Jareborg, N., and Wasserman, W. W. (2003) Identification of conserved regulatory elements by comparative genome analysis. *J. Biol.* **2**, 13.
- 19 Wingender, E., Dietze, P., Karas, H., and Knuppel, R. (1996) TRANSFAC: a database on transcription factors and their DNA binding sites. *Nucleic Acids Res.* **24**, 238–241.
- 20 Ovcharenko, I., Loots, G. G., Giardine, B. M., et al. (2005) Mulan: multiple-sequence local alignment and visualization for studying function and evolution. *Genome Res.* **15**, 184–194.
- 21 Giardine, B., Elnitski, L., Riemer, C., et al. (2003) GALA, a database for genomic sequence alignments and annotations. *Genome Res.* **13**, 732–741.

Improving Pairwise Sequence Alignment between Distantly Related Proteins

Jin-an Feng

Summary

Sequence alignment between remotely related proteins has been one of the more difficult problems in structural biology. Improvements have been achieved by incorporating information that enhances the diversity of the substitution matrices. NdPASA is a web-based server that optimizes sequence alignments between proteins sharing low percentages of sequence identity. The program integrates structure information of the template sequence into a global alignment algorithm by employing amino acids' neighbor-dependent propensities for secondary structure as unique parameters for alignment. NdPASA optimizes alignment by evaluating the likelihood of a residue pair in the query sequence matching against a corresponding residue pair adopting a particular secondary structure in the template sequence. The server is designed to aid homologous protein structure modeling. It is most effective when the structure of the template sequence is known. NdPASA can be accessed online at www.fenglab.org/bioserver.html.

Key Words: Sequence alignment; propensity; protein structures; sequence pattern; secondary structure.

1. Introduction

Protein sequence alignment has become an essential part of biomedical research. It is one of the standard approaches to explore potential functional activity of a newly discovered protein by identifying sequence homologues that may be evolutionarily related (1–3). One can often infer the structural and functional information of a new protein from the knowledge of well-characterized homologous proteins (4–7). In general, closely related protein sequences are

relatively easy to align using the existing sequence-based methods (8). However, the success rate of these methods in finding correct alignment is significantly reduced when the sequence identity between two aligned sequences is lower than 30 %, a threshold often referred to as the twilight zone (9).

The performance of a sequence alignment algorithm largely depends on its employed substitution matrix. PAM and BLOSUM are some of the most commonly used substitution matrices in sequence alignment algorithms. They are mainly derived from the frequencies of amino acid substitutions in a series of compiled families of protein sequences (10,11). Algorithms employing BLOSUM62 and PAM250 are most effective in identifying and aligning homologous proteins. Efforts to develop improved sequence alignment of remote homologues have focused on incorporating additional information that improves the diversity of the substitution matrices. These methods include the use of position-specific substitution profiles derived from multiple sequence alignment of protein families (3,12,13). Structure-based substitution matrices have also been developed. Such matrices are derived from the frequency of amino acids occupying similar positions in a series of structurally aligned proteins. Algorithms employing structure-based substitution matrices appear to have improved success in detecting and aligning remotely related protein sequences (14–20). Another approach that has achieved promising improvements in pairwise sequence alignment, particularly for sequences with low homology, is the sequence-template alignment method (17,21). This algorithm incorporates the structural knowledge of the template, as well as the amino acid propensities for secondary structures, into a substitution matrix for sequence alignment.

Although methods relying on structure-based sequence profiles are effective in identifying specific functional motifs in proteins, the overall improvement of these methods over the purely sequence-based methods in sequence alignments is limited. It appears that the inherent limitation of these methods is their dependence on the structure and the sequence diversity of the derived profile. If the profile construction emphasizes too much on sequence diversity, it would result in profiles with limited signature value. On the other hand, profiles derived from conserved proteins are not sensitive in detecting distantly related homologs. It has always been a challenge to reach an optimal balance in selecting protein sequences for constructing profiles. Some studies have suggested that the most effective profiles could be derived from sequences sharing 30–50 % homologue (22). Another key weakness of the structure-based sequence profiles is the lack of information in the loop regions, which are often ignored in structural alignments. For sequences with long and functional loop regions, the structure-based profile alignment methods could be ineffective. A

method incorporating both sequence- and structure-based profiles (the hybrid sequence profiles) has shown an improved performance in aligning proteins with distant homologues (23).

NdPASA is a new pairwise protein sequence alignment algorithm that incorporates the structure information of the aligning sequences. By incorporating the neighbor-dependent propensities (NDPs), the method has shown significant improvements over conventional methods in aligning proteins with remote sequence homologies (24). The NDP reflects the likelihood of an amino acid pair adopting a particular secondary structure conformation (α -helices, β -strands, and loops) in proteins (25,26). The rationale for applying the NDPs in sequence alignment is easily recognized. Methods employing sequence-based substitution matrix often have limited success in aligning sequences sharing low percentage of sequence identity. The incorporation of NDPs has allowed us to estimate the probability of an amino acid pair to be aligned with a corresponding amino acid pair adopting a specific secondary structure in the template sequence. For example, an amino acid pair in the query having a low neighbor-dependent propensity for α -helical conformation would be less likely aligned with an amino acid pair in an α -helix of the template. A structure-dependent gap opening and extension penalty scheme is also implemented. A higher gap penalty is applied for gaps within the regular secondary structures than for gaps in the loops. The performance of the NdPASA is compared with the PSI-Blast, the standard global alignment (GA) using BLOSUM62, and the individual propensity assisted sequence alignment (IPASA), a global sequence alignment algorithm that incorporates individual amino acid propensity for protein secondary structures. NdPASA performs most effectively when the structural information of the template sequence is available. It shows significant improvements over conventional methods when aligning sequence pairs sharing less than 20 % sequence identity (24).

2. Materials

NdPASA is written in JAVA with CGI interface. It is compiled on a LINUX-based workstation. The web server can be freely accessed online at www.fenglab.org/bioserver.html.

3. Methods

3.1. Neighbor-Dependent Sequence Analysis

3.1.1. Data Extraction From the PDB for Sequence Analysis

All analyses are performed using a rational database derived from a nonredundant set of PDB (PISCES) (27,28). The limit of redundancy chosen in this

study is at 25 % sequence identity. Only high-resolution structures are selected with a resolution cut-off at 2.5 Å. NMR structures are not included in this study. A total of 1430 proteins are selected from the PDB databank. The database containing these PDB entries (the sPDB) is used in the subsequent parsing protocols.

Two independent secondary structure assignment criteria are used to extract the sequences and the secondary structure information from the sPDB entries: the assignments given by the experimentalists and the assignments generated by the DSSP algorithm, which assigns secondary structure based on the analysis of backbone dihedral angles and hydrogen bonds (29). For the sake of comparing the two assignment methods, the more sophisticated DSSP structure assignments are converted as follows: helices, 3_{10} helices and the π helices are all considered as helices; the single-stranded β -strand and the multiple-stranded β -sheets are simply considered as strands; and a loop is defined as a region of a protein that is assigned as neither an α -helix nor a β -strand by either methods. The database system used in this study is the PostgreSQL packaged in the Redhat Linux 7.2. The sequence and secondary structure information of the sPDB entries are parsed into rational tables. Two sets of tables are compared. One derived from parsing the author assigned secondary structure information and the other is parsed according to the DSSP calculations. Considering that the authors' assignments are experimentally observed, we choose to use them as the standard for defining secondary structures in the protein structures. A double-check mechanism is installed to avoid erred assignments: the author assigned structural elements are compared with the DSSP assignment. The selected secondary structural elements are consistent with both assignment methods. The parsing exercise of the sPDB database results in libraries containing specific secondary structural elements (SSLs).

3.1.2. Neighbor-Dependent Amino Acid Propensities for Secondary Structures

The NDP reflects the neighboring probability of a pair of amino acids, in any combination, in the three classes of protein secondary structures S_j ($j = \alpha$ -helix, β -strand, and loop) (25,26). Neighbors can be defined as the first neighbor, where the amino acids in the pair are immediately next to each other in sequence; the second neighbor, where the pair of amino acid is separated by one amino acid residue in sequence; the third neighbor, where the pair of amino acid is separated by two amino acids; or the fourth neighbor, where the pair of amino acids is separated by three amino acids. First introduced by Chou and Fasman, the amino acid propensity (ϵ_a) is defined as the relative distribution of

amino acids in between the SSLs and the sPDB (30). It reflects the likelihood of amino acids adopting secondary structures. For neighbor-dependent propensity analysis, the frequency of occurrence of the residue type x at neighboring positions of the residue a in a specific secondary structure S_j is calculated as follows:

$$\varepsilon_{x(a \pm i), j} = \frac{[\sum x(a \pm i)_{S_j}] / n_{(pair)S_j}}{[\sum x(a \pm i)_P] / n_{(pair)P}} \quad (1)$$

where $\sum x(a \pm i)_{S_j}$ and $\sum x(a \pm i)_P$ are the occurrences of the residue type x at the $\pm i$ th positions of the residue a in the SSLs (S_j) and in the sPDB (P), respectively; $n_{(pair)S_j}$ and $n_{(pair)P}$ are the total number of residue pairs in S_j and P , respectively. The numerator in Eq. 1 calculates the frequency of occurrence of residue x neighbors the residue type a in the secondary structure S_j , whereas the denominator in the equation calculates the frequency of occurrence of the residue x neighboring the residue type a in the sPDB (P). The ratio of these values would be the propensity of residue x in S_j when it is neighbored with residue type a .

We applied the neighbor-dependent sequence analysis on the residues of immediate neighbors in secondary structures of proteins (25,26). The NDPs are represented as $\varepsilon_{x(a \pm 1), j}$. An $\varepsilon_{x(a \pm 1), j}$ value of 1.0 means that the occurrence of the residue pair, ax (or xa), in the secondary structure j is the same as its frequency of occurrence in proteins. A value greater than 1.0 means the pair has occurrence in the secondary structure j higher than that in proteins, i.e., the pair has preference for adopting the j secondary structure conformation. A $\varepsilon_{x(a \pm 1), j}$ value lower than unity would suggest that the pair is less likely to adopt the j secondary structure than random distributions. For example, a $\varepsilon_{p(A-1), \alpha} = 1.52$ in short helices means that Pro has 62 % more chance to be found in short helices than it in the proteins when it precede Ala, i.e., Pro at -1 position of Ala. On the other hand, a $\varepsilon_{p(A+1), \alpha} = 0.47$ suggests that Pro is less likely to be found in short helix when it follows Ala, i.e., Pro at $+1$ position of Ala.

The neighbor-dependent sequence analysis of proteins revealed that the amino acid pair preference for secondary structures has its unique pattern and that such pattern are not always predictable by assuming proportional contributions from the propensity values of the individual amino acids (25,26). Some of those sequence patterns are most pronounced in some subgroups of secondary structures, such as the short loop and helix subgroups. Our analysis also yielded a series of amino acid dyads that showed preference for a particular secondary

structure. The dyads are defined as the amino acid pairs “*ax*” having strong preferences for the particular secondary structure, whereas the amino acid pairs of “*xa*” having weak preferences for the secondary structure. It is evident that the neighbor-dependent protein sequence analysis method revealed some “hidden” sequence codes in proteins (25,26).

3.2. Sequence Alignment Incorporating Structure Information

3.2.1. Data Set of Sequence Pairs

The first step to developing a pairwise sequence alignment algorithm is to setup a data set of sequence pairs for training and testing (Fig. 1). The dataset of sequence pairs is constructed as follows. Protein sequences sharing less than 90 % sequence identity are first extracted from the nonredundant PISCES database (28). The resolution cut-off is at 2.5 Å. The homologous sequence pairs selected for this study satisfies two criteria: (1) the pair shares certain level of sequence homology and (2) the protein pair adopts same structural fold. Proteins homologous to the selected sequences in the PISCES database are identified by PSI-BLAST searches running against a nonredundant nucleotide database, the NCBI-nr. Sequences returned with *e*-values in the range of 10^{-6} to 1000 are retained. Sequence pairs with multiple high segment pairs are removed. The sequence pairs are then checked against the SCOP classification database to ensure that they belong to the same structural fold (31). Sequence pairs of the same protein family, protein super family, and fold in the SCOP database are selected. All selected sequence pairs also have comparable sizes where the shorter sequence of the pair is at least 80 % of the length of the longer sequence. The data set of sequence pairs contains a total of 2521 sequence pairs that shares sequence identities ranging between 13 and 25 %. Five hundred of these protein sequences pairs are selected as a training set, and 2021 pairs are selected as a testing set (Fig. 1). The training set is used to optimize parameters in the alignment algorithm.

3.2.2. The NdPASA Algorithm

NdPASA incorporates the information of secondary structure propensity into the Needleman-Wunsch global alignment algorithm with affined gap penalties (32). The Needleman-Wunsch dynamic programming algorithm is effective in finding the optimal scoring alignment or a set of alignments. To incorporate NDPs, we formulate $M(i, j)$ as follows:

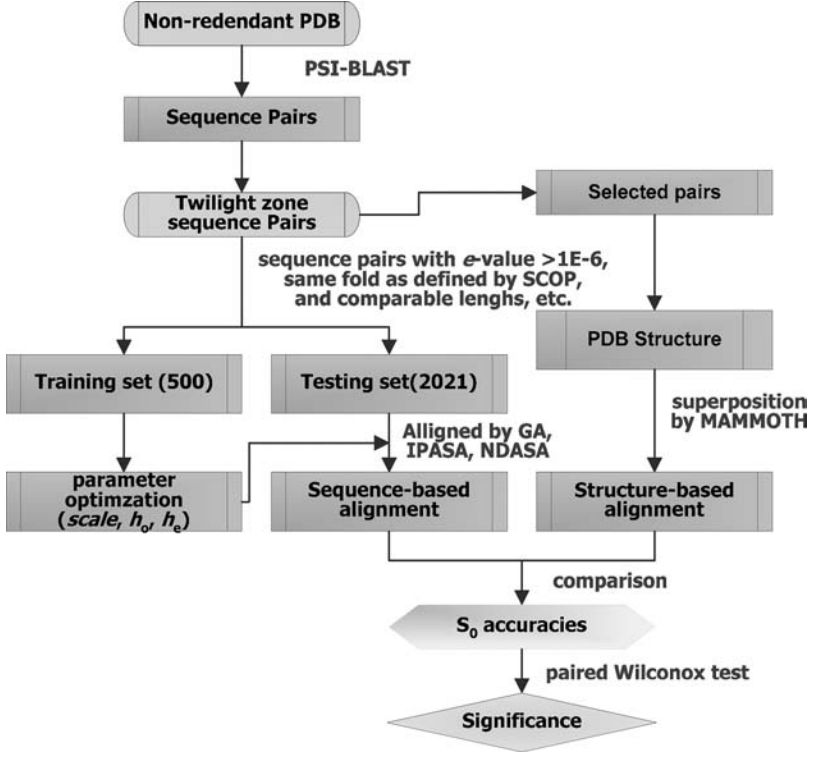


Fig. 1. A schematic workflow of the NdPASA algorithm.

$$M(i, j) = \max \left\{ \begin{array}{l} M(i-1, j-1) + s(x_i, y_j) + scale * p(x_{i-1}, x_i, x_{i+1} | ss_j) \\ I_x(i-1, j-1) + s(x_i, y_j) + scale * p(x_{i-1}, x_i, x_{i+1} | ss_j) \\ I_y(i-1, j-1) + s(x_i, y_j) + scale * p(x_{i-1}, x_i, x_{i+1} | ss_j) \end{array} \right\} \quad (2)$$

$$I_x(i, j) = \max \left\{ \begin{array}{l} M(i-1, j) - d(ss_j) \\ I_x(i-1, j) - e(ss_j) \end{array} \right\} \quad (3)$$

$$I_y(i, j) = \max \left\{ \begin{array}{l} M(i-1, j) - d(ss_j) \\ I_y(i-1, j) - e(ss_j) \end{array} \right\} \quad (4)$$

where $M(i, j)$ is the best score up to (i, j) given that x_i is aligned to y_j ; and $I_x(i, j)$ is the best score given that x_i is aligned to a gap; $I_y(i, j)$ is the best score given that y_j is in a insertion with respect to x ; and $s(x_i, y_j)$ is the amino acid substitution score of x_i and y_j given a substitution matrix (BLOSUM62 for this study) (33). Scale is the scaling factor, denoting a relative weight between

propensity score and substitution score. $p(x_{i-1}, x_i, x_{i+1}|ss_j)$ is the neighbor-dependent secondary structure propensity score for adopting the secondary structure element of the template sequence at position j . It depends on the amino acid at i th position and the amino acid types of its neighboring positions both proceeding $(i - 1)$ and following $(i + 1)$ the position i .

$$p(x_{i-1}, x_i, x_{i+1}|SS_j) = \frac{p(x_{i-1}, x_i|SS_j) + p(x_i, x_{i+1}|SS_j)}{2} \quad (5)$$

where $p(x_{i-1}, x_i|ss_j)$ and $p(x_i, x_{i+1}|ss_j)$ are the log values of the NDPs calculated using **Eq. 1** for residues pairs (x_{i-1}, x_i) and (x_i, x_{i+1}) to adopt the secondary structure element in the template at position j (ss_j), respectively. If a gap is applied at positions either proceeding or following the residue x_i , we take overall propensity as its available propensity.

The gap opening and extension penalties are also depended on the template's secondary structure element:

$$d(ss_j) = \begin{pmatrix} h_o & ss_j = 'H'/'S' \\ l_o & ss_j = 'L' \end{pmatrix} \quad (6)$$

$$e(ss_j) = \begin{pmatrix} h_e & ss_j = 'H'/'S' \\ l_e & ss_j = 'L' \end{pmatrix} \quad (7)$$

The parameters of the NdPASA that included *scale*, h_o , h_e , l_o , and l_e are optimized using a bootstrap approach. The 500 sequence pairs in the training set are used for the optimization procedure (**Fig. 1**). The performance of NdPASA was evaluated by comparing alignment results with the structure alignments of sequence pairs in the training set using MAMMOTH (**35**). The best performance is obtained by setting the parameters: *scale* = 2.2; h_o = 19.0; h_e = 2.0; l_o = 11.0; l_e = 1.0 (**24**).

3.2.3. NdPASA Performance Evaluation

The performance of NdPASA is compared with three different algorithms, including PSI-Blast, pairwise GA, and the IPASA (**24**). The alignment accuracies (S_0) of four algorithms are measured against the MAMMOTH structural alignments. The S_0 is defined as the ratio between the correctly aligned residue pairs and the total number of structurally aligned residue pairs (**24**). The sequence pairs selected for the evaluation share 13–25 % sequence identity. NdPASA demonstrated an average of 5–32 % improvement over other algorithms. Its improvement over the GA algorithm was 5–23 %, 2–10 % over the IPASA algorithm, and 17–48 % over the PSI-BLAST. The significantly larger improvement over PSI-BLAST could be

attributed in part to the fact that PSI-BLAST aligns sequence pairs locally (34). NdPASA is most effective in aligning sequence pairs with low sequence homology. For sequence pairs with approx 23–25 % identity, NdPASA has an average accuracy of 3–5 % better than that of the ISAPA and GA algorithms, respectively. For sequence pairs sharing approx 15–17 % identity, NdPASA performed by an average of 6–14 % better than that of the IPASA and GA algorithms, respectively (24). These results suggest that the sequence patterns derived from the neighbor-dependent sequence analysis of protein structures have more significant contribution to sequence alignments for sequence pairs that are remotely related.

3.3. The NdPASA Server

The NdPASA server is designed mainly to aid homologous protein structure modeling of query sequences (see notes). Figure 2 shows a schematic diagram of the algorithm implemented in the server. The first step to homologous modeling is to identify a template that shares the same fold as that of the query sequence. An input page is designed with three options. Option 1: user-identified template with an input of either a sequence or a PDB_ID. With an input of PDB_ID, NdPASA will automatically retrieve the corresponding sequence from the PDB and assign secondary structure elements using DSSP before performing sequence alignment with the query (29). When a template sequence is entered, NdPASA server will perform a PSI-BLAST search against

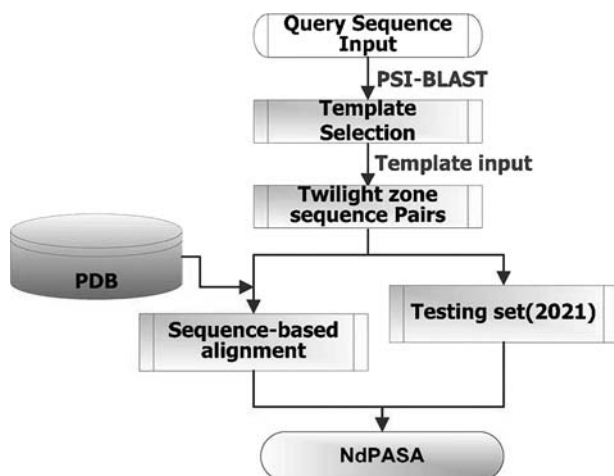


Fig. 2. A schematic diagram of the NdPASA server.

NdPAASA

Home
Query
Example Page
User Guide

Query Sequence:

1	DTVEYLKEAEAYFESLGGSSMKANTGVAKAFDAEAKVEAKLEKTLATVAPLFFAGTSS	6
61	TDLPQGTAAKAAIYANNDDVGKAGKARHEAGGVIAAANAGDGAAPGAALOKLGGTCKAC	12
121	HYDREED	127

Search results from PSI-BLAST (PDB database):

(1) 1CPR-
 gi|19425262|pdb|1CPR| St. Louis Cytochrome C; From The Purple Phototropic Bacterium, Rhodobacter Capsulatus
 Score=412 Expect=1.10432E-41 Identities=128/128(99%)

2	DTVEYLKEAEAYFESLGGSSMKANTGVAKAFDAEAKVEAKLEKTLATVAPLFFAGTSS	6
61	TDLPQGTAAKAAIYANNDDVGKAGKARHEAGGVIAAANAGDGAAPGAALOKLGGTCKAC	12
122	HYDREED	127

(2) 1EKY-A
 gi|7546519|pdb|1EKY| Chain A, Model Structure From Non-Noe Based Nmr Structure Calculation
 Score=412 Expect=1.10432E-41 Identities=128/128(100%)

2	DTVEYLKEAEAYFESLGGSSMKANTGVAKAFDAEAKVEAKLEKTLATVAPLFFAGTSS	6
62	TDLPQGTAAKAAIYANNDDVGKAGKARHEAGGVIAAANAGDGAAPGAALOKLGGTCKAC	12
122	HYDREED	127

(3) 1RCP-B
 gi|1633319|pdb|1RCP| Chain B, Cytochrome C;
 Score=412 Expect=1.10432E-41 Identities=128/128(100%)

2	DTVEYLKEAEAYFESLGGSSMKANTGVAKAFDAEAKVEAKLEKTLATVAPLFFAGTSS	6
62	TDLPQGTAAKAAIYANNDDVGKAGKARHEAGGVIAAANAGDGAAPGAALOKLGGTCKAC	12
122	HYDREED	127

(4) 1GQA-D
 gi|27573566|pdb|1GQA| Chain D, Cytochrome C; From Rhodobacter Sphaeroides
 Score=412 Expect=2.65296E-11 Identities=56/128(44%)

2	DAEYVFEARKGYPSLVALEFGLFALAKGKEMPTDAAALAKNSGLTITKTYPSDLTYPG	6
62	TSDDVEVG- TAAKAAIYQDAGQAKNGAFFEVAALRPAAGAGOKELAAA-VGKVGVD	12
120	KSHDD	127

(5) 1BBH-B
 gi|493868|pdb|1BBH| Chain B, Cytochrome C;
 Score=83 Expect=0.00155892 Identities=31/124(25%)

6	DTLITRPGAGVYPSWHNNGKIAKLEGGYNAQPEAAATIAAIAHNSGNGALVPGDTCKH	12
62	VDGTYTKYKFFQEMEDVGKIAEFYGAANTIAEVAATGEAAVATFGDVGAAKSCN	127
126	EXYR	127

Please select the hit sequence to align

- ☐ 1CPR- St. Louis Cytochrome C; From The Purple Phototropic Bacterium, Rhodobacter...
- ☐ 1EKY-A Chain A, Model Structure From Non-Noe Based Nmr Structure Calculation
- ☐ 1RCP-B Chain B, Cytochrome C;
- ☐ 1GQA-D Chain D, Cytochrome C; From Rhodobacter Sphaeroides
- ☐ 1BBH-B Chain B, Cytochrome C;

[illegible]

Fig. 3. A working example of NdPASA identifying template sequences for the query, cytochrome C' from the purple phototropic bacterium, using PSI-BLAST searches. The selected template is 1GQA-D, cytochrome C' from *Rhodobacter Sphaeroides*, which shares 44 % sequence identity with the query. The output of the alignment includes secondary structure assignments of the template.

the PDB database and return results containing PDB entries that share at least 80 % sequence identity with the template (3). The user is asked to select the desired template from the returned results. When a template is identified, the NdPASA server will perform secondary structure assignments and sequence alignment with the query sequence. Option 2: input template sequence with user-assigned secondary structures. When the structure of the template is not known, user can submit template sequence to a secondary structure prediction server, such as the PSIPRED server (<http://bioinf.cs.ucl.ac.uk/psipred/psiform.html>) (36). NdPASA accepts the returned secondary structure assignments for subsequent alignment. Option 3: template unknown. In this case, NdPASA server performs a PSI-BLAST search against the nonredundant protein structure database (PDB) for sequences homologous to the query using the BLOSUM62 matrix (Fig. 3). The user also has the option to choose different scoring matrices, including PAM250, PAM300, PAM120, BLOSUM35, BLOSUM45, BLOSUM50, BLOSUM60, BLOSUM62, and BLOSUM80. The gap opening and extending penalty parameters can also be changed. The default options of BLOSUM62, the gap opening (−11) and extending (−1) penalties are selected based on experimental test results (24). All returned results from the PSI-BLAST searches are displayed with their sequence names, PDB entry-ID, PSI-BLAST scores and the percentage sequence identities as determined by the PSI-BLAST when compared with the query protein. To limit the scope of template selection, we specified the PSI-BLAST output to contain only the sequences with either the top 5-ranked scores or the top 15-ranked scores for inspection. An optional filter is also implemented where the user may limit the output of the PSI-BLAST search to those sequences that share sequence homology above a defined threshold (Fig. 3). When a template candidate is identified, the user may select the radial button next to the sequence and click “submit” for optimized pairwise sequence alignment by NdPASA (Fig. 3). Alternatively, the user may choose one of the more sophisticated fold recognition servers, such as the GenTHREADER (36), 3D-PSSM (37), or Pcons5 (38), to identify template. Upon receiving command to align the query sequence against one of the templates identified by PSI-BLAST, the program fetches the template sequence from the PDB and assigns secondary structures for the template by using DSSP before performing pairwise sequence alignment. The result of NdPASA alignment is displayed in a pop-up window with the query and the template sequences aligned (Fig. 3). The secondary structure information of the template sequence is also displayed for inspection. NdPASA also produce an alignment in the standardized FASTA format so the results can be easily integrated with other bioinformatics tools.

4. Notes

1. The greatest strength of NdPASA is optimizing sequence alignment between proteins sharing less than 20 % sequence identity. Its improvement over conventional methods in aligning proteins sharing more than 30 % sequence identity is however not significant (24). It is highly dependent on the knowledge of the template secondary structure assignments. It performs most effectively when the structure of the template is available. Adopting a global sequence alignment algorithm, NdPASA is not suitable for aligning sequence pairs with large differences in size. In such cases, the user is advised to perform local sequence alignment using PSI-BLAST. When the homologous regions in both sequences are identified, NdPASA can be applied to optimize the alignment.

Acknowledgment

The author would like to thank Wei Li, Junwen Wang for their contributions in developing NdPASA. The author also thanks for the financial support from the National Institutes of Health (GM54630), the American Cancer Society (PRG9926301GMC), and an appropriation from the commonwealth of Pennsylvania.

Reference

- 1 Pearson, W. R. and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* **85**, 2444–2448.
- 2 Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) *J. Mol. Biol.* **215**, 403–410.
- 3 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25**, 3389–3402.
- 4 Chothia, C. and Lesk, A. M. (1986) The relation between the divergence of sequence and structure in proteins. *EMBO J.* **5**, 823–826.
- 5 Scharf, M., Schneider, R., Casari, G., et al. (1994) GeneQuiz: a workbench for sequence analysis. *ISMB* **2**, 348–353.
- 6 Abagyan, R. A. and Batalov, S. (1997) Do aligned sequences share the same fold? *J. Mol. Biol.* **273**, 355–368.
- 7 Teichmann, S. A., Chothia, C., and Gerstein, M. (1999) Advances in structural genomics. *Curr. Opin. Struct. Biol.* **9**, 390–399.
- 8 Feng, D. F., Johnson, M. S., and Doolittle, R. F. (1985) Aligning amino acid sequences: comparison of commonly used methods. *J. Mol. Evol.* **212**, 112–125.
- 9 Rost, B. (1999) Twilight zone of protein sequence alignments. *Protein Eng.* **12**, 85–94.

- 10 Dayhoff, M., Schwartz, R. M., and Orcutt, B. C. (1978) A model of evolutionary change in proteins, in *Atlas of Protein Sequence and Structure*, (Dayhoff, M. ed.), National Biomedical Research Foundation, Silver Springs, MD, pp. 345–352.
- 11 Henikoff, S. and Henikoff, J. G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* **89**, 10,915–10,919.
- 12 Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA* **84**, 4355–4358.
- 13 Marti-Renom, M. A., Madhusudhan, M. S., and Sali, A. (2004) Alignment of protein sequences by their profiles. *Protein Sci.* **13**, 1071–1087.
- 14 Shi, J., Blundell, T. L., and Mizuguchi, K. (2001) FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.* **310**, 243–257.
- 15 Ogata, K., Ohya, M., and Umeyama, H. (1998) Amino acid similarity matrix for homology modeling derived from structural alignment and optimized by the Monte Carlo method. *J. Mol. Graph. Model.* **16**, 178–189.
- 16 Johnson, M. S. and Overington, J. P. (1993) A structural basis for sequence comparisons An evaluation of scoring methodologies. *J. Mol. Biol.* **233**, 716–738.
- 17 Russell, R. B., Saqi, M. A., Sayle, R. A., Bates, P. A., and Sternberg, M. J. (1997) Recognition of analogous and homologous protein folds: analysis of sequence and structure conservation. *J. Mol. Biol.* **269**, 423–439.
- 18 May, A. C. and Johnson, M. S. (1995) Improved genetic algorithm-based protein structure comparisons: pairwise and multiple superpositions. *Protein Eng.* **8**, 873–882.
- 19 Prlic, A., Domingues, F. S., and Sippl, M. J. (2000) Structure-derived substitution matrices for alignment of distantly related sequences. *Protein Eng.* **13**, 545–550.
- 20 Blake, J. D. and Cohen, F. E. (2001) Pairwise sequence alignment below the twilight zone. *J. Mol. Biol.* **307**, 721–735.
- 21 Yang, A. S. (2002) Structure-dependent sequence alignment for remotely related proteins *Bioinformatics* **18**, 1658–1665.
- 22 Panchenko, A. R. and Bryant, S. H. (2002) A comparison of position-specific score matrices based on sequence and structure alignments. *Protein Sci.* **11**, 361–370.
- 23 Tang, C. L., Xie, L., Koh, I. Y. Y., Posy, S., Alexov, E., and Honig, B. (2003) On the role of structural information in remote homology detection and sequence alignment: New methods using hybrid sequence profiles. *J. Mol. Biol.* **334**, 1043–1062.
- 24 Wang, J. and Feng, J. A. (2005) NdPASA: a novel pair-wise protein sequence alignment that incorporates neighbor-dependent amino acid propensities. *Proteins* **58**, 628–637.
- 25 Crasto, C. J. and Feng, J. A. (2001) Sequence codes for extended conformation: a neighbor-dependent sequence analysis of loops in proteins. *Proteins* **42**, 399–413.
- 26 Wang, J. and Feng, J. A. (2003) Exploring the sequence patterns in the alpha-helices of proteins. *Protein Eng.* **16**, 799–807.

- 27 Bernstein, F. C., Koetle, T. F., Williams, G. J. B., et al. (1977) The protein data bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* **112**, 535–542.
- 28 Wang, G. and Dunbrack, R. L. (2003) PISCES: a protein sequence culling server *Bioinformatics* **19**, 1589–1591.
- 29 Kabsch, W. and Sander, C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **22**, 2577–2637.
- 30 Chou, P. Y. and Fasman, G. D. (1974) Conformational parameters for amino acids in helical, β -sheet, and random coil regions calculated from proteins. *Biochemistry* **15**, 211–221.
- 31 Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* **247**, 536–540.
- 32 Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.
- 33 Ginalski, K., Pas, J., Wyrwicz, L. S., von Grotthuss, M., Bujnicki, J. M., and Rychlewski, L. (2003) ORFeus: detection of distant homology using sequence profiles and predicted secondary structure. *Nucl. Acids Res.* **31**, 3804–3807.
- 34 Smith, T. F. and Waterman, M. S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197.
- 35 Ortiz, A. R., Strauss, C. E., and Olmea, O. (2002) MAMMOTH: matching molecular models obtained from theory: an automated method for model comparison. *Protein Sci.* **11**, 2606–2621.
- 36 Bryson, K., McGuffin, L. J., Marsden, R. L., Ward, J. J., Sodhi, J. S., and Jones, D. T. (2005) Protein structure prediction servers at University College London. *Nucl. Acids Res.* **33**, W36–W38.
- 37 Jones, D. T. (1999) GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.* **287**, 797–815.
- 38 Kelley, L. A., MacCallum, R. M., and Sternberg, M. J. (2000) Enhanced genome annotation using structural profiles in the program 3D-PSSM. *J. Mol. Biol.* **299**, 523–544.
- 39 Wallner, B. and Elofsson, A. (2005) Pcons5: combining consensus, structural evaluation and fold recognition scores. *Bioinformatics* **21**, 4248–4254.

Discovering Sequence Motifs

Timothy L. Bailey

Summary

Sequence motif discovery algorithms are an important part of the computational biologist's toolkit. The purpose of motif discovery is to discover patterns in biopolymer (nucleotide or protein) sequences to better understand the structure and function of the molecules the sequences represent. This chapter provides an overview of the use of sequence motif discovery in biology and a general guide to the use of motif discovery algorithms. This chapter examines the types of biological features that DNA and protein motifs can represent and their usefulness. This chapter also defines what sequence motifs are, how they are represented, and general techniques for discovering them. The primary focus of the chapter is on one aspect of motif discovery: discovering motifs in a set of unaligned DNA or protein sequences. This chapter also provides the steps useful for checking the biological validity and investigating the function of sequence motifs using methods such as motif scanning—searching for matches to motifs in a given sequence or a database of sequences. A discussion of some limitations of motif discovery concludes the chapter.

Key Words: Motif discovery; sequence motif; sequence pattern; protein domain; multiple alignment; position-specific scoring matrix; PSSM; position-specific weight matrix; PWM; transcription factor-binding site; transcription factor; promoter; protein features.

1. Sequence Motifs and Biological Features

Biological sequence motifs are short, usually fixed-length, sequence patterns. Many features of DNA, RNA, and protein molecules can be well approximated by motifs. For example, sequence motifs can represent transcription factor

binding sites (TFBSs), splice junctions, and binding domains in DNA, RNA, and protein molecules, respectively. Consequently, discovering sequence motifs can lead to a better understanding of transcriptional regulation, mRNA splicing, and the formation of protein complexes.

Regulatory elements in DNA are among the most important biological features that are represented by sequence motifs. The DNA footprint of the binding sites for a transcription factor (TF) is often well described by a sequence motif. These TFBS motifs specify the order and nucleotide preference at each position in the binding sites for a particular TF. Discovering TFBS motifs and relating them to the TFs that bind to them is a key challenge in constructing a model of the regulatory network of the cell (1,2). Motif discovery algorithms have been used to identify many candidate TFBS motifs that were later validated by experimental methods.

Protein motifs can represent, among other things, the active sites of enzymes. They can also identify protein regions involved in determining protein structure and stability. The PROSITE, BLOCKS, and PRINTS databases (3–5) contain hundreds of protein motifs corresponding to enzyme active sites, binding sites, and protein family signatures. Motifs can also be used to identify features that confer particular chemical characteristics (such as thermal stability) on proteins (6). Protein sequence motifs can also be used to classify proteins into families (5).

The importance of motif discovery is born out by the growth in motif databases such as TRANSFAC, JASPAR, SCPD, DBTBS, RegulonDB (7–10) for DNA motifs and PROSITE, BLOCKS, and PRINTS (3–5) for protein motifs. However, far more motifs remain to be discovered. For example, TFBS motifs are known for only about 500 vertebrate TFs, but it is estimated that there are about 2000 TFs in mammalian genomes alone (7,11).

Fixed-length motifs cannot represent all interesting patterns in biopolymer sequences. For instance, they are obviously not ideal for representing variable-length protein domains. For representing long, variable-length patterns, profiles (12), or hidden Markov models (HMMs) (13,14) are more appropriate. However, the dividing line between motifs and other sequence patterns (such as HMMs and profiles) is fuzzy, and is often erased completely in the literature. Some of the motif discovery algorithms discussed in the following sections, for example, do allow a single, variable-length “spacer,” thus violating (slightly) our definition of motifs as being of fixed length. However, this chapter does not consider patterns that allow free insertions and deletions, even though these are sometimes referred to as motifs in the literature.

2. Representing Sequence Motifs

Biological sequence motifs are usually represented either as regular expressions (REs) or position weight matrices (PWMs). These two ways of describing motifs have different strengths and weaknesses when it comes to expressive power, ease of discovery, and usefulness for scanning. Motif discovery algorithms exist that output their results in each of these types of motif representation. Some motif discovery algorithms do not output a description of the motif at all, but, rather, output a list of the “sites” (occurrences) of the motif in the input sequences. As we shall see, any set of sites can easily be converted to a regular expression or to a PWM.

REs are a way to describe a sequence pattern by defining exactly what sequences of letters constitute a match. The simplest regular expression is just a string of letters. For example, “T-A-T-A-A-T” is a DNA regular expression that matches only one sequence: “TATAAT.” (We follow the PROSITE convention of separating the positions in an RE by the dash “-” character, to distinguish them from sequences.) To allow more than one sequence to match an RE, extra letters (ambiguity codes) are added to the four-letter DNA sequence alphabet. For example, the IUPAC (15) code defines “W = A or T,” so the RE “T-A-T-A-W-T” matches both “TATATT” and “TATAAT.” For the 20-letter protein alphabet, ambiguity codes would be unwieldy, so sets of letters (enclosed in square brackets) may be included in an RE. Any of the letters within the square brackets is considered a match. As an added convenience, PROSITE protein motif REs allow a list of letters in curly braces, and any letter except the enclosed letters matches at that position. For example, the PROSITE N-glycosylation site motif is “N- {P} - [ST] - {P}.” This RE matches any sequence starting with “N,” followed by anything but “P,” followed by an “S” or a “T,” ending with anything but “P.” As noted earlier, some motif discovery programs allow for a variable-length spacer separating the two, fixed-length ends of the motif. This is particularly applicable to dyad motifs in DNA (16,17). The RE “T-A-C-N(2,4)-G-T-A” describes such a motif, where “N” is the IUPAC “match anything” ambiguity code. The entry “-N(2,4)-” in the RE matches any DNA sequence of length from two to four, so sequences matching this RE have lengths from 8–10, and begin and end with “TAC” and “GTA,” respectively.

Whereas REs define the set of letters that may match at each position in the motif, PWMs define the probability of each letter in the alphabet occurring at that position. A PWM is an n by w matrix, where n is the number of letters in the sequence alphabet (4 for DNA, 20 for protein), and w is the number of positions in the motif. The entry in row a , column i in the PWM, designated $P_{a,i}$,

is the probability of letter a occurring at position i in the motif. Mathematically, PWMs specify the parameters of a position-specific multinomial sequence model that assumes each position in the motif is statistically independent of the others. A PWM defines a probability for every possible sequence of the correct width (w). The positional independence assumption implies that the probability of a sequence is just the product of the corresponding entries in the PWM. For example, the probability of the sequence “TATAAT” according to a PWM (with six columns) is

$$Pr(\text{“TATAAT”}) = P_{T,1} \cdot P_{A,2} \cdot P_{T,3} \cdot P_{A,4} \cdot P_{A,5} \cdot P_{T,6}.$$

As with REs, it is possible to extend the concept of PWMs to allow for variable-length spacers, but this is not commonly done by existing motif discovery algorithms.

For the purposes of motif scanning, many motif discovery algorithms also output a position-specific scoring matrix (PSSM), which is often confusingly referred to as a PWM. The entries in a PSSM are usually defined as

$$S_{a,j} = \log_2 \frac{P_{a,j}}{f_a}, \quad (1)$$

where f_a is the overall probability of letter a in the sequences to be scanned for occurrences of the motif. The PSSM score for a sequence is given by summing the appropriate entries in the PSSM, so the PSSM score of the sequence “TATAAT” is

$$S(\text{“TATAAT”}) = S_{T,1} + S_{A,2} + S_{T,3} + S_{A,4} + S_{A,5} + S_{T,6}.$$

PSSM scores are more sensitive for scanning than probabilities because they take the “background” probability of different letters into account. This increases the match score for uncommon letters and decreases the score for common letters, thus reducing the rate of false-positives caused by nonuniform distribution of letters in sequences.

Underlying both REs and PWMs are the actual occurrences (sites) of the motif in the input sequences. The relationship among the motif sites, an RE and a PWM is illustrated in [Fig. 1](#) which shows the JASPAR “broad-complex 1” motif. The nine motif sites from which this motif was constructed are shown aligned with each other at the top of [Fig. 1](#). The corresponding RE motif (using the IUPAC DNA ambiguity codes) is shown beneath the alignment. Below that, the counts of each letter in the corresponding alignment columns are shown.

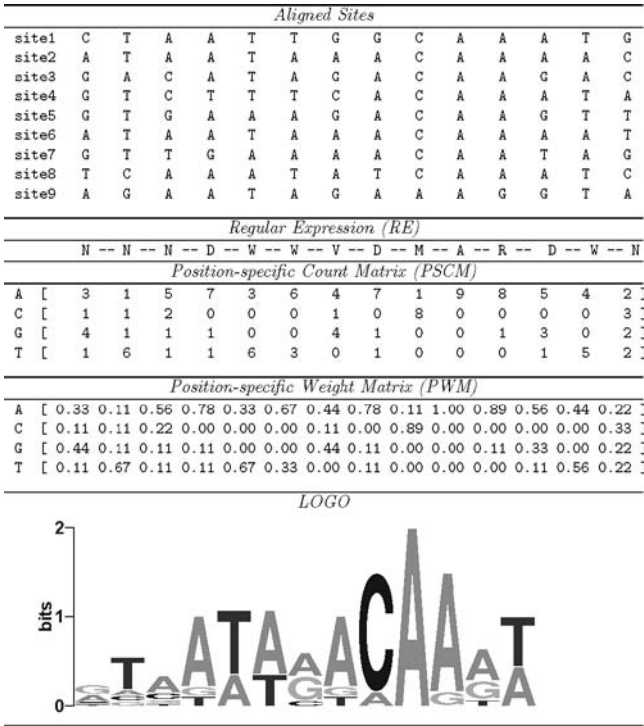


Fig. 1. Converting an alignment of sites into a regular expression (RE) and a position weight matrix (PWM). The alignment of DNA sites is shown at the top. The RE (using the IUPAC ambiguity codes) is shown aligned below the sites. The corresponding counts of each letter in each alignment column—the position-specific count matrix (PSCM)—are shown in the next box. The PWM is shown below that. The last box shows the information content “LOGO” for the motif.

Below those, the corresponding PWM entries are shown. They were computed by normalizing each column in the counts matrix so that it sums to one. Beneath the PWM, the “LOGO” representation (18) for the motif is shown, where the height of each letter corresponds to its contribution to the motif’s information content (see Eq. 2).

Any alignment of motif sites can be converted into either an RE or PWM motif in the manner illustrated in Fig. 1. Usually a small amount (called a “pseudocount”) is added to the counts in the position-specific count matrix before the PWM is created by normalization to avoid probabilities of zero being assigned to letters that were not observed. This is sensible because, based on

only a fraction of the actual sites, we cannot be certain that a particular letter never occurs in a real site.

Both PWMs and REs are used by motif discovery algorithms because each has advantages. The main advantages of REs are that they are easy for humans to visualize and for computers to locate. It is also easier to compute the statistical significance of a motif defined as a regular expression (16,19). On the other hand, PWMs allow for a more nuanced description of motifs than regular expressions, because each letter can “match” a particular motif position to varying degrees, rather than simply matching or not matching. This makes PWM motifs (converted to PSSMs using **Eq. 1**) more suitable for motif scanning than REs in most applications. When used to model binding sites in nucleotide molecules, there is evidence that PWMs capture some of the statistical mechanics of protein-nucleotide binding (20–22). An extension of PWMs, called HMMs, has also been shown to be an invaluable way to represent protein domains (for example, the PFAM database of protein domains [23]). The main disadvantage of PWMs for motif discovery is that they are far more difficult for computer algorithms to search for. This is true precisely because PWMs are so much more expressive than REs.

3. General Techniques for Motif Discovery

Many approaches have been tried for *de novo* motif discovery. In general, they fall into four broad classes. The predominant approach can be called the “focused” approach: assemble a small set of sequences and search for over-represented patterns in the sequences relative to a background model. Numerous examples of available algorithms that use this approach are given in **Table 3**. A related approach can be called the “focused discriminative” approach: assemble two sets of sequences and look for patterns relatively over-represented in one of the input sets (24,25). The “phylogenetic” approach uses sequence conservation information about the sequences in a single input set (26–29). The “whole-genome” approach looks for over-represented, conserved patterns in multiple alignments of the genomes of two or more species (30,31). This chapter does not describe the “whole-genome” approach in any detail.

A sequence motif describes a pattern that recurs in biopolymer sequences. To be interesting to biologists, the pattern should correspond to some functional or structural feature that the underlying molecules have in common. None of the computational techniques for motif discovery previously listed can guarantee to find only biologically relevant motifs. The most that can generally be said about a computationally discovered motif is that it is statistically significant, given underlying assumptions about the sequences in which it occurs.

The predominant approach to sequence motif discovery is the focused approach, which searches for novel motifs in a set of unaligned DNA or protein sequences suspected to contain a common motif. We discuss how the sequences can be selected in the next section. RE-based motif discovery algorithms for the focused approach search the space of all possible REs either exhaustively or heuristically (incompletely). Their objective is usually to identify the REs whose matches are most over-represented in the input sequences (relative to a background sequence model, randomly generated background sequences, or a set of negative control sequences). PWM-based motif discovery algorithms search the space of PWMs for motifs that maximize an objective function that is usually equal to (or related to) “log likelihood ratio” of the PWM:

$$LLR(PWM) = \sum_{j=1}^w \sum_{a \in A} P_{a,j} \log_2 \frac{P_{a,j}}{f_a}, \quad (2)$$

where the $P_{a,j}$ are estimated from the predicted motif sites as illustrated in [Fig. 1](#). The appropriateness of this objective function is justified by both Bayesian decision theory ([32](#)), and, in the case of TFBSs, by binding energy considerations ([20,22](#)). When the background frequency model is uniform, LLR is equivalent to “information content”.

4. Discovering Motifs in Sets of Unaligned Sequences

This section describes the steps necessary for successfully discovering motifs using the “focused” approach. Each motif discovery application is different, but most have the following steps in common:

1. Assemble: select the target sequences.
2. Clean: mask or remove “noise.”
3. Discover: run a motif discovery algorithm.
4. Evaluate: investigate the validity and function of the motifs.

In the first step, assemble a “dataset” of DNA or protein sequences that may contain an unknown motif encoding functional, structural, or evolutionary information. Next, if appropriate, mask or remove confounding sequence regions such as low-complexity regions and known repeat elements. Then, run a motif discovery algorithm using the set of sequences and with parameter settings appropriate to the application. The next step is intended to weed out motifs that are likely to be chance artefacts rather than motifs corresponding to functional or structural features, and to try to glean more information about them. This step can involve determining if a discovered motif is similar to a known motif, or if its occurrences are conserved in orthologous genes. Each of these steps will be described in more detail in the following sections.

4.1. Assemble: Select the Target Sequences

The most important step in motif discovery is to assemble a set of sequences that is likely to contain multiple occurrences of one or more motifs (*see Note 2*). For motif discovery algorithms to successfully discover motifs, it is important that the sequence set be as “enriched” as possible in the motifs. Obviously, if the sequences consist entirely of motif occurrences for a single motif, the problem of motif discovery is trivial (*see Fig. 1*). In practice, the guiding idea behind assembling a sequence set is to come as close as possible to such a set. To achieve this, all available background knowledge should be applied to achieve the following goals:

1. Include as many sequences as possible that contain the motifs.
2. Keep the sequences as short as possible.
3. Remove sequences that are unlikely to contain any motifs.

How to assemble an input sequence set depends, of course, on what type of motifs are being looked for and where they are expected to occur. In most applications, there are two basic steps:

1. Clustering.
2. Extraction.

First, cluster genes (or other types of sequences) based on information about coexpression, cobinding, function, environment, or orthology to select those likely to have a common motif. Second, extract the relevant (portions of) sequences from an appropriate sequence database.

As an example, to discover regulatory elements in DNA, you might select upstream regions of genes that show coexpression in a microarray experiment (*33*). Coexpression can be determined by clustering of expression profiles. Alternatively, you could use the sequences that bound to a TF in a ChIP-chip experiment (*1,34*). A third possibility is to use information on coexpressed promoters from CAGE tag experiments (*35,36*). To these sequence sets, you might also add orthologous sequences from related organisms under the assumption that the regulatory elements have been conserved in them.

To discover protein functional or structural sequence motifs, you could select proteins belonging to a given protein family based on sequence similarity, structure, annotation, or other means (*23,37,38*). You might further refine the selection to only include proteins from organisms with a particular feature, such as the ability to live in extreme environments (*39*). Another protein motif discovery application uses information from protein–protein interaction experiments. You can assemble a set of proteins that bind to a common host protein, to discover sequence motifs for the interacting domains.

Most algorithms require sets of sequences in FASTA format. Proteins are usually easily extracted directly from the available sequence databases. Genomic DNA is more problematic, because annotation of genes, promoters, transcriptional start sites, introns, exons, and other important features is not always reliable. Several web servers available to aid in extracting the relevant sequences for discovering regulatory elements in genomic DNA are shown in [Table 1](#).

4.2. Clean: Mask or Remove “Noise”

Many genomic “phenomena” can masquerade as motifs and fool motif discovery algorithms (*see* **Note 3**). Things such as low-complexity DNA, low-complexity protein regions, tandem repeats, SINES, and ALUs all contain repetitive patterns that are problematic for existing motif-finding algorithms. It is therefore advisable to filter out these features from the sequences in the input set. This is done by running one or more of the programs described in [Table 2](#) on the set of sequences. Typically, the programs replace regions containing genomic “noise” with the ambiguity code for “match anything” in the appropriate sequence alphabet. This usually means “N” for DNA sequences and “X” for protein. Most motif discovery algorithms will not find motifs containing large numbers of these ambiguity codes, so they are effectively made invisible by this replacement process.

Table 1
Web Servers for Extracting Upstream Regions and Other Types of Genomic Sequence

Web server name	Function
RSA Tools	Retrieve upstream regions for a large number of organisms http://rsat.ulb.ac.be/rsat/
PromoSer	Retrieve human, rat, and mouse upstream regions including alternative promoters http://biowulf.bu.edu/zlab/PromoSer
Genlynx	Locate genes in the genomic sequence of human, mouse, and rat http://www.genelinx.org
UCSC genome browser (73)	View and extract genomic sequences and alignments of multiple genomes http://genome.ucsc.edu

Table 2
Programs for Filtering “Noise” in DNA and Protein Sequences

Program name	Function
DUST	Filter low-complexity DNA http://blast.wustl.edu/pub/dust
XNU	Filter low-complexity protein http://blast.wustl.edu/pub/xnu
SEG	Filter low-complexity protein http://blast.wustl.edu/pub/seg
RepeatMasker	Filter interspersed DNA repeats and low-complexity sequence http://www.repeatmasker.org/cgi-bin/WEBRepeatMasker
Tandem Repeats Finder	Identifies the positions of DNA tandem repeats http://tandem.bu.edu/trf

Table 2 lists some of the programs available to help to mask or remove confounding regions from the input sequence set. The DUST program (40) can be used to filter out low-complexity DNA. The XNU program (41) will filter low-complexity (short period repeat) amino acid sequences. An alternative program for filtering out low-complexity protein sequences is the SEG program (42). Interspersed DNA repeats and low-complexity DNA sequence can both be filtered using the RepeatMasker program (43). A web server is available for RepeatMasker, whereas at the time of this writing it was necessary to download, compile, and install the DUST, XNU, and SEG programs. Tandem repeats can be identified in DNA using the “Tandem Repeats Finder” program. It has a web server that allows the user to upload a sequence set (in FASTA format) for analysis. Of course, you should be aware that functional motifs can sometimes occur in the types of regions filtered by these programs, so caution is advised. It is important to study the documentation available with the programs to be sure to know what types of sequence they mask or identify. If you suspect that they may be masking regions containing the motifs of interest, the user can always try running motif discovery algorithms on both the original and cleaned sets of sequences, and compare the results.

4.3. Discover: Run a Motif Discovery Algorithm

Many motif discovery algorithms are currently available. Most require installation of software on the user’s computer. **Table 3** lists a variety of algorithms

Table 3
Some Motif Discovery Algorithms With Web Servers

PWM-based algorithms	
MEME	DNA or Protein motifs using EM http://meme.nbcr.net
Gibbs	DNA or protein motifs using Gibbs sampling http://bayesweb.wadsworth.org/gibbs/gibbs.html
AlignACE	DNA motifs using Gibbs sampling http://atlas.med.harvard.edu
CompareProspector	DNA motifs in eukaryotes using “biased” Gibbs sampling; requires multiple alignment http://seqmotifs.stanford.edu
BioProspector	DNA motifs in prokaryotes and lower eukaryotes using Gibbs sampling http://seqmotifs.stanford.edu
Mdscan	DNA motifs; specialized for ChIP-chip probes http://seqmotifs.stanford.edu
RE-based algorithms	
BlockMaker	Protein motifs http://blocks.fhcrc.org/blocks/make_blocks.html
RSA Tools	DNA motifs using RE-based or Gibbs sampler-based algorithms http://rsat.ulb.ac.be/rsat
Weeder	DNA motifs using RE-based algorithm http://www.pesolelab.it
YMF	DNA motifs using RE-based algorithm http://wingless.cs.washington.edu/YMF
Combination algorithms	
TAMO	Yeast, mouse, human; input as gene names or probe names, fetches upstream regions http://fraenkel.mit.edu/webtamo

that have web servers where sequences can be uploaded directly, thus avoiding the need to install any new software. The table groups the algorithms according to whether they search for motifs expressed as REs or PWMs. Some of the algorithms are general purpose and can discover motifs in either DNA or protein sequences (MEME [44], Gibbs [45]). Some algorithms are specialized only for DNA (AlignACE [46], BioProspector [29], MDscan [47], RSA Tools [16,48],

Weeder [49], and YMF [50]). CompareProspector (51) is specialized for DNA sequences and requires that the user to input the sequence set and conservation levels for each sequence position derived from a multiple alignment. Block-Maker (52) finds motifs only in protein sequences. The TAMO algorithm (53) runs multiple motif discovery algorithms (MEME, AlignACE, MDscan) and combines the results.

Many excellent algorithms are not included in Table 3 because they did not appear to have a (working) web server at the time of this writing. Motif discovery algorithms require a great deal of computational power, so most authors have elected to distribute their algorithms rather than provide a web server. Other motif discovery algorithms include ANNSpec (25), Consensus (54), GLAM (55), Improbizer (56), MITRA (57), Motif-Sampler (58), Phyme (26), QuickScore (59), and SeSiMCMC (60).

Different classes of algorithms (RE- and PWM-based) have different strengths and weaknesses, so it is often helpful to run one or more motif discovery algorithms of each type on the sequence set. Doing this can increase the chances of finding subtle motifs. Also, the confidence in a given motif is increased when it is found by multiple algorithms, especially if the algorithms belong to different classes (*see Note 4*).

Some motif discovery algorithms (for example, CompareProspector) can take direct advantage of conservation information in multiple alignments of orthologous sequence regions. This has been shown to improve the detection of TFBSs because they tend to be over-represented in sequence regions of high conservation (61,62). To find subtle motifs, it can also be useful to run each motif discovery algorithm with various settings of the relevant parameters. What the relevant parameters are depends on the particular problem at hand and the motif discovery algorithm being used. The user should read the documentation for the algorithm being used for hints about what nondefault parameter settings may be appropriate for different applications. In general, important parameters to vary include the limits on the width of the motif, the model used to model background (or “negative” sequences), the number of sites expected (or required) in each sequence, and the number of motifs to be reported (if the algorithm can detect multiple motifs).

4.4. Evaluate: Investigate the Validity and Function of the Motifs

One of the most difficult tasks in motif discovery is deciding which, if any, of the discovered motifs is “real.” Three complementary approaches can aid in this. First, you can attempt to determine whether a given motif is statistically significant. Second, you can investigate whether the function of the motif is already known or

can be inferred. Third, you can look for corroborating evidence for the motif. We discuss each of these approaches in what follows.

Most motif discovery algorithms report motifs regardless of whether they are likely to be statistical artefacts. In other words, they “discover” motifs even in randomly generated (or randomly selected) sequences. This is sometimes referred to as the “GIGO” rule: garbage-in, garbage-out. This, however, is not necessarily a bad thing; many truly functional DNA motifs are not statistically significant in the context of the kinds of sequence sets that can be assembled using clustered data from coexpression, ChIP-chip, CAGE, or other current technologies. So, it is important that motif discovery algorithms be able to detect these types of motifs even if they lie beneath the level of statistical significance that we might like. Measures of the statistical significance of a motif above the 0.05 significance level are still useful because they can be used to prioritize motifs for further validation.

Some motif discovery algorithms report an estimate of the statistical significance of the motifs they report. For example, MEME (44), Consensus (54), and GLAM (55) report the *E*-value of the motif: the probability of a motif of equal or greater information content occurring in a sequence set consisting of shuffled versions of each sequence. Motifs with very small (less than 0.05) *E*-values are statistically significant according to the given definition of random (shuffled sequences). The reported *E*-values are known to be conservative (too large), so motifs with *E*-values greater than 0.05 may still be significant. Gibbs (45) uses a different statistical test (Wilcoxon signed-rank test) to determine motif significance. The relative merits of these two methods of assessing motif significance has not been studied.

Sometimes it is advisable to estimate motif significance empirically (63). Many motif discovery algorithms do not make any attempt to report the statistical significance of the motifs they discover relative to the number of possible motifs that might have appeared in a randomly selected or generated sequence-set, so empirical estimation is the only available approach. Another reason to evaluate the significance of motifs empirically is that the motif significance estimates given by algorithms such as those named in the previous paragraph tend to be conservative, causing some biologically significant motifs appear to be artefacts (see **Note 5**).

Empirical significance testing is very computationally expensive and, therefore, should generally be done using motif discovery algorithms installed on the local computer. Empirical significance testing is done by running the motif discovery algorithm hundreds of times on random sets of sequences of the same type and length, and with the same input parameters to the program,

as were used in finding the motifs the user is interested in evaluating. The motif scores for all the motifs found in the random runs are plotted as a histogram—the empirical score distribution. The significance of the real motifs’ scores can be estimated by seeing where they lie on the histogram. The motif score can be either the information content score, or the objective function score of the particular motif discovery method—usually some measure of over-representation. How to select (or generate) the random sequence sets depends on the application. For example, if the real sequences are selected upstream regions of genes from a single organism, a reasonable random model would be to use randomly chosen upstream regions from the same organism.

Whether or not you choose to determine their statistical significance, you probably want to determine as much as possible about the function of the motifs (*see* **Note 6**). To do this, the motifs can be used to search databases of motifs and motif families, and motifs can be used individually and in groups to search databases of sequences for single matches and local clusters of matches. DNA motifs can be searched against known vertebrate TF motifs in JASPAR. The JASPAR database also contains motifs that represent the binding affinities of whole families of TFs. If the motif matches one of these family motifs, it may be the TFBS motif of a TF in that structural family. You can search a protein motif against the BLOCKS or PRINTS (**5**) database using the LAMA program (**64**) to identify if it corresponds to a known functional domain. These databases are summarized in **Table 4**.

You will also want to see if your motif occurs in sequences other than those in the sequence set in which it was discovered. This is done by scanning a database of sequences using your motif (or motifs) as the query. This can help validate the motif(s) and shed light on its (their) function. If the novel occurrences have a positional bias relative to some sequence landmark (for example, the transcriptional start site), this can be corroborating evidence that the motif may be functional (**46**). In bacteria, real TFBSs are more likely to

Table 4
Some Searchable Motif Databases With Web Servers

Database	Description
JASPAR	Searchable database of vertebrate TF motifs and TF-family motifs http://mordor.cgb.ki.se/cgi-bin/jaspar2005/jaspar_db.pl
BLOCKS	Databases of protein signatures
PRINTS	http://blocks.fhcrc.org/blocks-bin/LAMA_search.sh

occur relatively close to the gene for their TF, so proximity to the TF can increase confidence in TFBSs predicted by motif scanning (2). Similarly, when the occurrences of two or more motifs cluster together in several sequences, it may be evidence that the motifs are functionally related. (Care must be taken that the clustering of cooccurrences is not simply because of sequence homology.) The functions of the sequences where novel motif occurrences are detected can also provide a hint to the motif’s function. Scanning with multiple motifs can shed light on the interaction/co-occurrence of protein domains and on *cis*-regulatory modules in DNA.

Numerous programs are available to assist in determining the location, co-occurrence, and correlation with functional annotation of motifs in other sequences. The MAST program (65) allows the search a selection of sequence databases with one or more unordered protein or DNA motifs. The PATSER program (54) allows the searching of sequences that the user uploads for occurrences of their DNA motif. Several tools are available for searching for *cis*-regulatory modules that include the user’s TFBS motifs. They include MCAST (66), Comet (67), and Cluster-buster (68). To determine if the genomic positions of the matches to the user’s motif or motifs are correlated with functional annotation in the GO (Genome Ontology) database (69), GONOME can be used (70). If the genomic positions are strongly correlated with a particular type of gene, this can shed light on the function of the user’s motif. Some tools for motif scanning that are available for direct use via web servers are listed in Table 5.

Table 5
Some Web Servers for Scanning Sequences for Occurrences of Motifs

Program	Description
MAST	Search one or more motifs against a sequence database; provides a large number of sequence databases or allows a set of sequences to be uploaded http://meme.nbcr.net
PATSER	Search a motif against sequences uploaded http://rsat.ulb.ac.be/rsat/patser_form.cgi
Comet, Clusterbuster	Search for <i>cis</i> -regulatory modules http://zlab.bu.edu/zlab/gene.shtml
GONOME	Find correlations between occurrences of the motif and genome annotation in the GO database http://gonome.imb.uq.edu.au/index.html

An important way to validate DNA motifs is to look at the conservation of the motif occurrences in both the original sequences and in sequences the user scans as described in the previous paragraph. It has been shown that TFBSs exhibit higher conservation than the surrounding sequence in both yeast and mammals (30,31). Motifs whose sites (as determined by the motif discovery algorithm) and occurrences (as determined by scanning) show preferential conservation are less likely to be statistical artefacts. Databases such as the UCSC genome browser (*see* Table 1) can be consulted to determine the conservation of motif sites and occurrences.

5. Limitations of Motif Discovery

Awareness of the limitations of motif discovery can guide the user to more success in the use of the approaches outlined in this chapter. Some limitations have to do with the difficulty of discovering weak motifs in the face of noise. Spurious motifs are another source of difficulty. Another limitation is caused by the difficulty in determining which sequences to include in the input sequence set (*see* Note 1).

You can often think of motif discovery as a “needle-in-a-haystack” problem where the motif is the “needle” and the sequences in which it is embedded is the “haystack.” Because motif discovery algorithms depend on the relative over-representation of a motif in the input set of sequences, a motif is “weak” if it is not significantly over-represented in the input sequences relative to what is expected by chance (or relative to a negative set of sequences) (71).

Over-representation is a function of several factors including:

1. The number of occurrences of the motif in the sequences.
2. How similar all the occurrences are to each other.
3. The length of the input sequences.

The more occurrences of the motif the sequences contain, the easier it will be to discover. So adding sequences to the input set that have a high probability of containing a motif will increase the likelihood of discovering it. Conversely, it can be helpful to reduce the number of sequences by removing ones unlikely to contain motif occurrences. Many DNA motifs (for example, TFBSs) tend to have low-levels of similarity among occurrences, so it is especially important to limit sequence length and the number of “noise” sequences (ones not containing occurrences) in the input sequence set. Over-representation depends inversely on the length of the sequences, so it is always good to limit the length of the input sequences as much as possible. Current motif discovery algorithms perform poorly at discovering TFBS when the sequences are longer than 1000 bp.

Spurious motifs are motifs caused by nonfunctional, repetitive elements such as SINES, ALUs, and by skewed sequence composition in regions such as CpG islands. Such regions will contain patterns that are easily detected by motif discovery algorithms and may obscure real motifs. To help avoid this, prefilter the sequences using the methods described in **Subheading 4.2**. In some cases, prefiltering is not an option because the motifs of interest may lie in the regions that would be removed by filtering. For example, DNA regulatory elements often occur in or near CpG islands. In such cases, manual inspection using the methods of the previous section is necessary to remove spurious motifs. Using an organism-specific (or genomic-region-specific) random model is possible with some motif discovery algorithms, and may help to reduce the number of spurious motifs.

It is also important to be aware of the reliability of the methodologies used in selecting the input sequences for motif discovery. For example, sequences selected based on microarray expression data may miss many TFs because their level of expression is too low for modern methods to detect reliably (2). ChIP-on-chip has become a popular procedure for studying genome-wide protein–DNA interactions and transcriptional regulation, but it can only map the probable protein–DNA interaction loci within 1–2 Kbp resolution. Even if the input sequences all contain a TFBS motif, many TFBS motifs will not be detected in such long sequences using current motif discovery algorithms (72). Another difficulty in discovering regulatory elements in DNA is that they can lie very far from the genes they regulate in eukaryotes, making sequence selection difficult.

6. Notes

1. Be aware of the limitations of the motif discovery algorithms used. For example, do not input an entire genome to most motif discovery algorithms—they are not designed for that and will just waste a lot of computer time without finding anything.
2. Use all available background information to select the sequences in which the user will discover motifs. Include as many sequences as possible that contain the motifs. Keep the sequences as short as possible. Remove sequences that are unlikely to contain any motifs.
3. Prepare the input sequences carefully by masking or removing repetitive features that are not of interest such as ALUs, SINES, and low-complexity regions. Filtering programs such as DUST, XNU, SEG, and RepeatMasker can help to do this.
4. Try more than one motif discovery algorithm on the user's data. They have different strengths and one program will often detect a motif missed by other programs.

5. Evaluate the statistical significance of the user's motifs. Remember that most motif discovery algorithms report motifs in any dataset even though they may not be statistically significant. Even if the algorithm estimates the significance of the motifs it finds, these estimates tend to be very conservative, making it easy to reject biologically important motifs. The motif discovery algorithm should be rerun on many sets of sequences selected to be similar to the your "real" sequences, but that are not expected to be enriched in any particular motif. Compare the scores of the "real" motifs with those of motifs found in the "random" sequences to determine if they are statistically unusual.
6. Compare the motifs discovered, to known motifs contained in appropriate motif databases such as those in [Table 4](#).

References

- 1 Blais, A. and Dynlacht, B. D. (2005) Constructing transcriptional regulatory networks. *Genes Dev.* **19**, 1499–1511.
- 2 Tan, K., McCue, L. A., and Stormo, G. D. (2005) Making connections between novel transcription factors and their DNA motifs. *Genome Res.* **15**, 312–320.
- 3 Hulo, N., Bairoch, A., Bulliard, V., et al. (2006) The PROSITE database. *Nucleic Acids Res.* **34**, D227–D230.
- 4 Henikoff, J. G., Greene, E. A., Pietrokovski, S., and Henikoff, S. (2000) Increased coverage of protein families with the Blocks Database servers. *Nucleic Acids Res.* **28**, 228–230.
- 5 Attwood, T. K., Bradley, P., Flower, D. R., et al. (2003) PRINTS and its automatic supplement, prePRINTS. *Nucleic Acids Res.* **31**, 400–402.
- 6 La, D. and Livesay, D. R. (2005) Predicting functional sites with an automated algorithm suitable for heterogeneous datasets. *BMC Bioinformatics* **6**, 116.
- 7 Matys, V., Kel-Margoulis, O. V., Fricke, E., et al. (2006) TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**, D108–D110.
- 8 Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W. W., and Lenhard, B. (2004) JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.* **32**, D91–D94.
- 9 Zhu, J. and Zhang, M. Q. (1999) SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics* **15**, 607–611.
- 10 Makita, Y., Nakao, M., Ogasawara, N., and Nakai, K. (2004) DBTBS: database of transcriptional regulation in *Bacillus subtilis* and its contribution to comparative genomics. *Nucleic Acids Res.* **32**, D75–D77.
- 11 Waterston, R. H., Lindblad-Toh, K., Birney, E., et al. (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**, 520–562.
- 12 Gribskov, M. and Veretnik, S. (1996) Identification of sequence pattern with profile analysis. *Methods Enzymol.* **266**, 198–212.
- 13 Eddy, S. R. (1998) Profile hidden Markov models. *Bioinformatics* **14**, 755–763.

- 14 Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994) Hidden Markov models in computational biology. Applications to protein modeling. *J. Mol. Biol.* **235**, 1501–1531.
- 15 CBN and U.-I.C.o.B.N. (1970) Abbreviations and symbols for nucleic acids, polynucleotides and their constituents. recommendations 1970. *Eur. J. Biochem.* **15**, 203–208.
- 16 van Helden, J., Andre, B., and Collado-Vides, J. (1998) Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.* **281**, 827–842.
- 17 van Helden, J., Rios, A. F., and Collado-Vides, J. (2000) Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res* **28**, 1808–1818.
- 18 Schneider, T. D. and Stephens, R. M. (1990) Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.* **18**, 6097–6100.
- 19 Reinert, G., Schbath, S., and Waterman, M. S. (2000) Probabilistic and statistical properties of words: an overview. *J. Comput. Biol.* **7**, 1–46.
- 20 Schneider, T. D., Stormo, G. D., Gold, L., and Ehrenfeucht, A. (1986) Information content of binding sites on nucleotide sequences. *J. Mol. Biol.* **188**, 415–431.
- 21 Berg, O. G. and von Hippel, P. H. (1987) Selection of DNA binding sites by regulatory proteins. Statistical-mechanical theory and application to operators and promoters. *J. Mol. Biol.* **193**, 723–750.
- 22 Berg, O. G. and von Hippel, P. H. (1988) Selection of DNA binding sites by regulatory proteins. II. The binding specificity of cyclic AMP receptor protein to recognition sites. *J. Mol. Biol.* **200**, 709–723.
- 23 Finn, R. D., Mistry, J., Schuster-Bockler, B., et al. (2006) Pfam: clans, web tools and services. *Nucleic Acids Res.* **34**, D247–D251.
- 24 Sinha, S. (2003) Discriminative motifs. *J. Comput. Biol.* **10**, 599–615.
- 25 Workman, C. T. and Stormo, G. D. (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac. Symp. Biocomput.* 467–478.
- 26 Sinha, S., Blanchette, M., and Tompa, M. (2004) PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics* **5**, 170.
- 27 Moses, A. M., Chiang, D. Y., and Eisen, M. B. (2004) Phylogenetic motif detection by expectation-maximization on evolutionary mixtures. *Pac. Symp. Biocomput.* 324–335.
- 28 Siddharthan, R., Siggia, E. D., and van Nimwegen, E. (2005) PhyloGibbs: a gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput. Biol.* **1**, e67.
- 29 Liu, X., Brutlag, D. L., and Liu, J. S. (2001) BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. *Pac. Symp. Biocomput.* 127–138.

- 30 Xie, X., Lu, J., Kulbokas, E. J., Golub, T. R., et al. (2005) Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. *Nature* **434**, 338–345.
- 31 Kellis, M., Patterson, N., Birren, B., Berger, B., and Lander, E. S. (2004) Methods in comparative genomics: genome correspondence, gene identification and regulatory motif discovery. *J. Comput. Biol.* **11**, 319–355.
- 32 Duda, R. O. and Hart, P. E. (1973) *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc., New York.
- 33 Seki, M., Narusaka, M., Abe, H., et al. (2001) Monitoring the expression pattern of 1300 Arabidopsis genes under drought and cold stresses by using a full-length cDNA microarray. *Plant Cell*. **13**, 61–72.
- 34 Harbison, C. T., Gordon, D. B., Lee, T. I., et al. (2004) Transcriptional regulatory code of a eukaryotic genome. *Nature* **431**, 99–104.
- 35 Kawaji, H., Kasukawa, T., Fukuda, S., et al. (2006) CAGE Basic/Analysis Databases: the CAGE resource for comprehensive promoter analysis. *Nucleic Acids Res.* **34**, D632–D636.
- 36 Kodzius, R., Matsumura, Y., Kasukawa, T., et al. (2004) Absolute expression values for mouse transcripts: re-annotation of the READ expression database by the use of CAGE and EST sequence tags. *FEBS Lett.* **559**, 22–26.
- 37 Tatusov, R. L., Fedorova, N. D., Jackson, J. D., et al. (2003) The COG database: an updated version includes eukaryotes. *BMC Bioinformatics* **4**, 41.
- 38 Andreeva, A., Howorth, D., Brenner, S. E., Hubbard, T. J. P., Chothia, C., and Murzin, A. G. (2004) SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Res.* **32**, D226–D229.
- 39 La, D., Silver, M., Edgar, R. C., and Livesay, D. R. (2003) Using motif-based methods in multiple genome analyses: a case study comparing orthologous mesophilic and thermophilic proteins. *Biochemistry* **42**, 8988–8998.
- 40 Tatusov, R. L., and Lipman, D. J. Dust, in the NCBI/Toolkit available at <http://blast.wustl.edu/pub/dust/>.
- 41 Claverie, J. -M., and States, D. J. (1993) Information enhancement methods for large scale sequence analysis. *Comput. Chem.* **17**, 191–201.
- 42 Wootton, J. C. and Federhen, S. (1996) Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.* **266**, 554–571.
- 43 Smit, A., Hubley, R., and Green, P. Repeatmasker, available at <http://www.repeatmasker.org>.
- 44 Bailey, T. L. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **2**, 28–36.
- 45 Thompson, W., Rouchka, E. C., and Lawrence, C. E. (2003) Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Res.* **31**, 3580–3585.

- 46 Roth, F. P., Hughes, J. D., Estep, P. W., and Church, G. M. (1998) Finding DNA regulatory motifs within unaligned non-coding sequences clustered by whole-genome mRNA quantitation. *Nat. Biotechnol.* **16**, 939–945.
- 47 Liu, X. S., Brutlag, D. L., and Liu, J. S. (2002) An algorithm for finding protein-DNA binding sites with applications to chromatin immunoprecipitation microarray experiments. *Nat. Biotechnol.* **20**, 835–839.
- 48 van Helden, J., Andre, B., and Collado-Vides, J. (2000) A web site for the computational analysis of yeast regulatory sequences. *Yeast* **16**, 177–187.
- 49 Pavesi, G., Mereghetti, P., Mauri, G., and Pesole, G. (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res.* **32**, W199–W203.
- 50 Sinha, S. and Tompa, M. (2003) YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res.* **31**, 3586–3588.
- 51 Liu, Y., Liu, X. S., Wei, L., Altman, R. B., and Batzoglou, S. (2004) Eukaryotic regulatory element conservation analysis and identification using comparative genomics. *Genome Res.* **14**, 451–458.
- 52 Henikoff, S., Henikoff, J. G., Alford, W. J., and Pietrokovski, S. (1995) Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene* **163**, GC17–GC26.
- 53 Gordon, D. B., Nekludova, L., McCallum, S., and Fraenkel, E. (2005) TAMO: a flexible, object-oriented framework for analyzing transcriptional regulation using DNA-sequence motifs. *Bioinformatics* **21**, 3164–3165.
- 54 Hertz, G. Z. and Stormo, G. D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* **15**, 563–577.
- 55 Frith, M. C., Hansen, U., Spouge, J. L., and Weng, Z. (2004) Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.* **32**, 189–200.
- 56 Ao, W., Gaudet, J., Kent, W. J., Muttumu, S., and Mango, S. E. (2004) Environmentally induced foregut remodeling by PHA4/FoxA and DAF-12/NHR. *Science* **305**, 1742–1746.
- 57 Eskin, E. and Pevzner, P. A. (2002) Finding composite regulatory patterns in DNA sequences. *Bioinformatics* **18**, S354–S363.
- 58 Thijs, G., Marchal, K., Lescot, M., et al. (2002) A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J. Comput. Biol.* **9**, 447–464.
- 59 Regnier, M. and Denise, A. (2004) Rare events and conditional events on random strings. *Discrete Math. Theor. Comput. Sci.* **6**, 191–214.
- 60 Favorov, A. V., Gelfand, M. S., Gerasimova, A. V., Ravcheev, D. A., Mironov, A. A., and Makeev, V. J. (2005) A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the

- signal length. *Bioinformatics* **21**, 2240–2245.
- 61 Tagle, D. A., Koop, B. F., Goodman, M., Slightom, J. L., Hess, D. L., and Jones, R. T. (1988) Embryonic epsilon and gamma globin genes of a prosimian primate (*Galago crassicaudatus*). Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.* **203**, 439–455.
 - 62 Duret, L. and Bucher, P. (1997) Searching for regulatory elements in human noncoding sequences. *Curr. Opin. Struct. Biol.* **7**, 399–406.
 - 63 Macisaac, K. D., Gordon, D. B., Nekludova, L., et al. (2006) A hypothesis-based approach for identifying the binding specificity of regulatory proteins from chromatin immunoprecipitation data. *Bioinformatics* **22**, 423–429.
 - 64 Pietrovski, S. (1996) Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucleic Acids Res.* **24**, 3836–3845.
 - 65 Bailey, T. L. and Gribskov, M. (1998) Combining evidence using p-values: application to sequence homology searches. *Bioinformatics* **14**, 48–54.
 - 66 Bailey, T. L. and Noble, W. S. (2003) Searching for statistically significant regulatory modules. *Bioinformatics* **19**, II16–II25.
 - 67 Frith, M. C., Spouge, J. L., Hansen, U., and Weng, Z. (2002) Statistical significance of clusters of motifs represented by position specific scoring matrices in nucleotide sequences. *Nucleic Acids Res.* **30**, 3214–3224.
 - 68 Frith, M. C., Li, M. C., and Weng, Z. (2003) Cluster-Buster: finding dense clusters of motifs in DNA sequences. *Nucleic Acids Res.* **31**, 3666–3668.
 - 69 Ashburner, M., Ball, C. A., Blake, J. A., et al. (2000) Gene ontology: tool for the unification of biology. *Nat. Genet.* **25**, 25–29.
 - 70 Stanley, S., Bailey, T., and Mattick, J. (2006) GONOME: measuring correlations between gene ontology terms and genomic positions. *BMC Bioinformatics* **7**, 94.
 - 71 Keich, U. and Pevzner, P. A. (2002) Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics* **18**, 1382–1390.
 - 72 Tompa, M., Li, N., Bailey, T. L., et al. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.* **23**, 137–144.
 - 73 Kent, W. J., Sugnet, C. W., Furey, T. S., et al. (2002) The human genome browser at UCSC. *Genome Res.* **12**, 996–1006.

Discovery of Conserved Motifs in Promoters of Orthologous Genes in Prokaryotes

Rekin's Janky and Jacques van Helden

Summary

We present a method to predict *cis*-acting elements for a given gene by detecting over-represented motifs in promoters of a set of orthologous genes in prokaryotes (single-gene, multiple-genomes approach). The method has been used successfully to detect regulatory elements at various taxonomical levels in prokaryotes. A web interface is available at the Regulatory Sequence Analysis Tools site (<http://rsat.scmbb.ulb.ac.be/rsat/>).

Key Words: Transcriptional regulation; pattern discovery; pattern matching; phylogenetic footprinting; prokaryotes; RSAT; get-orthologs; retrieve-seq; dyad-analysis.

1. Introduction

1.1. Context

In Chapter 21, we described a method to predict *cis*-acting elements by discovering over-represented motifs in promoters of several coregulated genes of a given organism (single-genome, multi-genes approach). The same method can be used to predict *cis*-acting elements putatively involved in the regulation of a single gene, by discovering over-represented motifs in promoters of orthologous genes from related genomes (single-gene, multi-genomes approach). The same principle also called phylogenetic footprinting ([1](#)), has been initially proposed as a way to detect conserved elements in the regulatory regions of selected mammalian genes ([1–4](#)). It has also been applied to predict regulation

in completely sequenced bacterial genomes (5–7). The approach relies on the assumption that, because of selective pressure, regulatory elements evolve slower than surrounding non-coding sequences. A condition to obtain good results is to dispose of genomes sufficiently related, so that the regulation of the gene of interest has been conserved, but sufficiently distant to have allowed the surrounding noncoding sequences to diverge.

The originality of our implementation is that we use a pattern discovery method, dyad-analysis, which analyzes the frequencies of dyads (spaced pairs of trinucleotides), and estimates the statistical significance of each motif. This permits to select restrictive thresholds to reduce the risk of false positives. Note that dyad-analysis is used here as example but other pattern discovery tools can be used as well. The website supports several pattern discovery algorithms, including a word-counting method (oligo-analysis [8]), a greedy algorithm (consensus [9,10]), and a Gibbs motif sampler (gibbs [11,12]). With more than 300 prokaryote genomes currently available (January 2006) on the National Center for Biotechnology Information website (<http://www.ncbi.nlm.nih.gov/genomes/lproks.cgi>), we are now in state to apply this single-gene, multigenomes approach to several taxonomical groups. In addition, the power of the methods is expected to increase with time, when more genomes will become available (553 microbial genomes are currently in progress).

1.2. Study Case

As a study case, we will illustrate the method on the gene *tyrP*, which codes for a tyrosine-specific permease. This gene is regulated by the TyrR protein, which ensures its repression in the presence of tyrosine and induction by phenylalanine (13). TyrR binding motif, also called TYR R box, is the reverse palindromic dyad *TGTAAAn₆TTTACA*. Two occurrences of this motif have been identified in the upstream region of *tyrP* from *Escherichia coli* and other close organisms like *Salmonella* (14–16).

2. Materials

1. Genome installation. See Chapter 21 for the description of genome sources and formats.
2. Basic Local Alignment Search Tool (BLAST) similarities. To predict homologous genes, BLAST (17,18) is used to detect similarities between all genes from a query organism (*E. coli* K12) and other prokaryote genomes. For each pair of genomes, the whole set of protein sequences were compared using blastp (protein sequences vs

protein sequences) with an upper threshold of 10^{-5} on the *e*-value. The relation between gene *x* in genome *A* and gene *y* in genome *B* is called bidirectional best hits, when *x* is the best hit for *y* among all genes from genome *A* and vice versa. In comparative genomics, getting the bidirectional best hits is the common computational way to identify putative orthologs (see **Note 1**).

3. Methods

The analysis consists in a succession of steps, realized by interconnected tools performing the following tasks.

1. get-orthologs: getting putative orthologs for a gene of interest (e.g., *tyrP*).
2. retrieve-seq: retrieving upstream sequences for these orthologs.
3. dyad-analysis: discovering statistically over-represented patterns in the set of upstream sequences.
4. dna-pattern: locating instances of the discovered patterns in the upstream sequences.
5. feature-map: drawing a feature-map of the predicted sites.

3.1. Getting Homo (get-orthologs)

1. Connect to the RSAT website at <http://rsat.scmbb.ulb.ac.be/rsat/>.
2. The left frame of the home page presents a menu with the available tools. Click on the tool “Get orthologs.” A form appears, where the user can select the parameters of ortholog selection.
3. In the pop-up menu “Organism,” select *Escherichia coli K12*.
4. In the text area under “Query genes,” type the common name of the gene of interest (e.g., *tyrP*).
5. Select the “taxon of interest” (e.g., Gammaproteobacteria) (see **Note 2**).
6. To extract bidirectional best hits (the putative orthologs), make sure that the upper threshold on “rank” is set to 1 (see **Note 3**).
7. Leave all other options unchanged and click on the button “GO.”
8. After a few seconds, the result is displayed (**Table 1**). The comment lines (those starting with a semicolon) indicate the parameters used for the selection of orthologs. The result table displays one row per BLAST hit. The two first columns indicate the ID of the putative orthologous gene, and the name of the organism in which it has been found. The third column indicates the ID of the query gene (this is useful when several genes are analyzed together). Optional information can be displayed in the following columns. For this example, we only selected the percent of identity and the *e*-value, but additional fields (alignment length, number of mismatches, number of gaps, bit score, rank) can be selected by checking them in the “return” option of the previous form.
9. Click on the button “retrieve sequences” to transfer the list of orthologs to the form of the sequence retrieval program.

Table 1

Output Table of the Program Get-Orthologs With Query Gene *tyrP* of *Escherichia coli* Strain K12 and Gammaproteobacteria as Referenced Taxon^a

```
; get-orthologs -v 1 -i tmp/get-orthologs.2006_01_27.221610 -org
Escherichia_coli_K12 -taxon Gammaproteobacteria -return ident -lth ali_len 50 -return
e_value -uth e_value 1e-05 -uth q_rank 1 -uth s_rank 1
; Input files
;   input tmp/get-orthologs.2006_01_27.221610
; Query organism      Escherichia_coli_K12
; Query genes 1
;   tyrP NP_416420.1
; Query filter        (NP_416420.1)
; Reference taxon      Gammaproteobacteria
; Reference organisms  67
;   Haemophilus_ducreyi_35000HP
;   Xanthomonas_citri
;   [...]
;   Xanthomonas_oryzae_KACC10331
; Threshold values
;   Parameter      Lower      Upper
;   e_value         none      1e-05
;   ali_len         50        none
;   s_rank          none      1
;   q_rank          none      1
```

#ref_gene	ref_org	query_gene	ident	e_value
NP_874217.1	Haemophilus_ducreyi_35000HP	NP_416420.1	45.41	6e-74
NP_310642.1	Escherichia_coli_O157H7	NP_416420.1	99.75	0.0
NP_460893.1	Salmonella_typhimurium_LT2	NP_416420.1	88.83	1e-163
NP_934846.1	Vibrio_vulnificus_YJ016	NP_416420.1	46.65	1e-77
YP_049324.1	Erwinia_carotovora_atroseptica_SCR11043	NP_416420.1	73.13	8e-135
YP_069783.1	Yersinia_pseudotuberculosis_IP32953	NP_416420.1	79.16	2e-144
NP_456501.1	Salmonella_typhi	NP_416420.1	88.83	1e-163
NP_245669.1	Pasteurella_multocida	NP_416420.1	48.64	2e-76
NP_837525.1	Shigella_flexneri_2a_2457T	NP_416420.1	100.00	0.0
NP_416420.1	Escherichia_coli_K12	NP_416420.1	100.00	0.0
NP_761142.1	Vibrio_vulnificus_CMCP6	NP_416420.1	46.65	4e-78
NP_233158.1	Vibrio_cholerae	NP_416420.1	48.51	5e-82
NP_820522.1	Coxiella_burnetii	NP_416420.1	32.74	7e-42
NP_754215.1	Escherichia_coli_CFT073	NP_416420.1	99.75	0.0
NP_670279.1	Yersinia_pestis_KIM	NP_416420.1	79.16	2e-144
YP_407569.1	Shigella_boydii_Sb227	NP_416420.1	99.25	0.0
YP_088169.1	Mannheimia_succiniciproducens_MBEL55E	NP_416420.1	47.64	8e-77
YP_402759.1	Shigella_dysenteriae	NP_416420.1	99.26	0.0
YP_340615.1	Pseudoalteromonas_haloplanktis_TAC125	NP_416420.1	47.17	6e-83
NP_804766.1	Salmonella_typhi_Ty2	NP_416420.1	88.83	1e-163
YP_130649.1	Photobacterium_profundum_SS9	NP_416420.1	49.16	8e-86
NP_797805.1	Vibrio_parahaemolyticus	NP_416420.1	47.15	5e-79

(Continued)

Table 1
(Continued)

#ref_gene	ref_org	query_gene	ident	e_value
NP_707797.1	Shigella_flexneri_2a	NP_416420.1	100.00	0.0
NP_438637.1	Haemophilus_influenzae	NP_416420.1	46.88	3e-77
NP_717668.1	Shewanella_oneidensis	NP_416420.1	48.74	6e-81
NP_288342.1	Escherichia_coli_O157H7_EDL933	NP_416420.1	99.75	0.0
YP_204580.1	Vibrio_fischeri_ES114	NP_416420.1	44.50	7e-69
NP_404813.1	Yersinia_pestis_CO92	NP_416420.1	79.16	2e-144
NP_929586.1	Photothabdus_luminescens	NP_416420.1	69.48	8e-130
NP_992307.1	Yersinia_pestis_biovvar_Mediaevails	NP_416420.1	79.16	2e-144
YP_310163.1	Shigella_sonnei_Ss046	NP_416420.1	99.26	0.0
; Job started 2006_01_27.221610				
; Job done 2006_01_27.221615				

^aAmong the 65 Gammaproteobacteria currently installed on RSAT, the program found a putative ortholog in 31 distinct species (47%).

3.2. Retrieving Upstream Sequences (*retrieve-seq*)

After having identified the putative orthologs of the gene of interest in a given taxon, the program *retrieve-seq* can be used to retrieve the corresponding upstream (non-coding) sequences. The single-genome use of this tool has been described in Chapter 21. We will now use the multi-genome option. In principle, no parameter needs to be changed for this: the selection of optimal parameters has been done during the transfer from *get-orthologs* to *retrieve-seq*. To make sure that everything is fine, check the following options:

1. The option “multiple organism” should be checked. “Feature type” must be “CDS” (*see Note 4*).
2. “Sequence type” should be “upstream.”
3. Leave the sequence limits (From, To) to “default” (*see Note 5*).
4. The option “prevent overlap with upstream ORFs” should be checked (*see Note 6*).
5. The “Sequence label” should be set to “gene + organism + name” (*see Note 7*).
6. Click “GO.”
7. After some time, the result appears, in the form of a link giving access to the retrieved sequences (*see Note 8*).
8. Below the link to the sequence, a list of buttons allows the user to send the retrieved sequences as input to a variety of pattern matching and pattern discovery tools. We will send the results to “dyad-analysis.”

3.3. Pattern Discovery (*dyad-analysis*)

In prokaryotes, most transcription factors belong to the Helix-Turn-Helix family (19). Proteins of this class typically form homodimers, whose tridimensional structure is symmetrical. As a consequence, many prokaryote transcription factors bind to spaced motifs (also called dyads), where each half-motif is bound by one element of the homodimer. The width of the spacing between the two contact points is transcription factor-specific, and can vary from 0 to ~20 nt. Because we are working with bacterial sequences, we will illustrate the pattern discovery step by using a tool dedicated to the detection of spaced motifs: *dyad-analysis* (20) (*see Note 9*).

1. In the previous section, we sent the promoter sequences to the program *dyad-analysis*. The browser should now display the *dyad-analysis* input form.
2. Make sure that the option “purge sequences” is checked (*see Note 10*).
3. Make sure that the background model is set to “monad frequencies from the input sequences” (*see Note 11*).
4. Leave all other parameters unchanged and click “GO.”
5. A typical example of result is shown in Table 2.

Table 2

Significant Dyads Discovered With Dyad-Analysis in Upstream Sequences of 31 *tyrP* Gammaproteobacterial orthologs^a

```

Sequence type          DNA
; Nb of sequences      31
; Sum of sequence lengths  7102
; Dyad parameters
;  dyad type          any dyad
;  minimal spacing      0
;  maximal spacing      20
;  dyad positions       6947
;  valid               4068
;  discarded           2879      (contain other letters than ACGT)
;  distinct dyads      43680
;  dyads tested for significance  30893
; Threshold values
; Parameter    Lower      Upper
; occ          1          none
; occ_sig      0          none
; Estimation of expected dyad frequencies
; Monad calibration in input sequences
; Sequences:
[...list of sequence names skipped ...]
; column headers
;  1    sequence
;  3    expected_freq
;  4    occ          observed occurrences

```

```

; 5    exp_occ      expected occurrences
; 6    occ_P        occurrence probability (binomial)
; 7    occ_E        E-value for occurrences (binomial)
; 8    occ_sig      occurrence significance (binomial)
; 9    ovl_occ      number of overlapping occurrences
; 10   all_occ      number of non-overlapping +
overlapping occurrences
; 11   rank         rank
; 13   remark       remark

```

sequence		expected_freq	occ	exp_occ	occ_P	occ_E	occ_sig	ovl_occ	all_occ	rank	remark
gtan{11} aca	0.0006778599160	26	2.46	4.7e- 17	1.4e- 12	11.84	15	41	1		
tgtn{12} aca	0.0003144021321	19	1.13	2.5e- 16	7.8e- 12	11.11	0	19	2	inv_rep	
gtan{10} tac	0.0003653713023	20	1.34	2.7e- 16	8.2e- 12	11.09	0	20	3	inv_rep	
gtan{12} cac	0.0003846013708	18	1.38	1.1e- 13	3.4e- 09	8.47	2	20	4		
gtgn{13} aca	0.0003567683769	15	1.28	5.1e- 11	1.6e- 06	5.80	2	17	5		
taan{10} aca	0.0012977449708	24	4.75	2.2e- 09	6.8e- 05	4.17	6	30	6		
aaan{9} aca	0.0016901901854	25	6.23	7.4e- 08	2.3e- 03	2.64	1	26	7		

(Continued)

Table 2
(Continued)

sequence	expected_freq	occ	exp_occ	occ_P	occ_E	occ_sig	ovl_occ	all_occ	rank	remark
gtgn{11} tta	0.0007363092033	16	2.67	1.2e- 07	3.7e- 03	2.43	1	17	8	
cgtn{12} aca	0.0003300107486	10	1.19	1.5e- 06	4.8e- 02	1.32	0	10	9	
ctgn{13} aca	0.0004192028428	11	1.51	1.9e- 06	5.7e- 02	1.24	0	11	10	
agtn{14} aca	0.0005083949370	12	1.82	1.9e- 06	5.9e- 02	1.23	0	12	11	
tgtn{0} aaa	0.0016901901854	21	6.55	1.1e- 05	3.4e- 01	0.47	0	21	12	
gtgn{0} taa	0.0007363092033	13	2.85	1.6e- 05	4.9e- 01	0.31	0	13	13	
accn{12} cat	0.0002654571797	8	0.96	1.8e- 05	5.4e- 01	0.27	0	8	14	
tcgn{13} aca	0.0003478491674	9	1.25	1.8e- 05	5.4e- 01	0.26	0	9	15	
gtan{9} tta	0.0013989874863	18	5.16	2.9e- 05	8.9e- 01	0.05	10	28	16	

;Job started 27/01/06 22:28:43 CET

;Job done 27/01/06 22:28:51 CET

"The significance of the top dyad is very high ([11](#), [84](#)). Some of the header rows and two columns were suppressed owing to space limitations.

Table 3**The Most Significant Assembly of Dyads Detected With Dyad-Analysis^a**

```

; pattern-assembly -v 1 -subst 0 -2str -maxfl 1 -subst 0 -i
public_html/tmp/dyad-analysis.2006_01_27.222841.res ; Input
file public_html/tmp/dyad-analysis.2006_01_27.222841.res;
Input score column          8
; Output score column       0
; two strand assembly
; max flanking bases        1
; max substitutions         0
; max assembly size         50
; max number of patterns    100
; number of input patterns   16
;
;
;assembly # 1 seed: gtannnnnnnnnnnaca 20 words length
; align rev_cpl score
agtannnnnnnnnnnnnaca.. .tgtannnnnnnnnnnnnact 1.23
.gtgannnnnnnnnnnnntac... .gtannnnnnnnnnnnncac. 8.47
.gtgannnnnnnnnnnnnaca.. .tgtannnnnnnnnnnnncac. 5.80
.gtgannnnnnnnnnntta.... .taannnnnnnnnnnnncac. 2.43
.gtgtaa..... .ttacac. 0.31
..tgtannnnnnnnnnntac... .gtannnnnnnnnnnnnaca.. 11.84
..tgtannnnnnnnnnnnnaca.. .tgtannnnnnnnnnnnnaca.. 11.11
..tgtannnnnnnnnnnnncac. .gtgannnnnnnnnnnnnaca.. 5.80
..tgtannnnnnnnnnntta.... .taannnnnnnnnnnnnaca.. 4.17
..tgtannnnnnnnnnnttt.... .aaannnnnnnnnnnnnaca.. 2.64
..tgtannnnnnnnnnnnnact agtannnnnnnnnnnnnnaca.. 1.23
..tgtaaa..... .tttaca.. 0.47
...gtannnnnnnnnnnnnaca.. .tgtannnnnnnnnnnnntac... 11.84
...gtannnnnnnnnnntac... .gtannnnnnnnnnnnntac... 11.09
...gtannnnnnnnnnnnncac. .gtgannnnnnnnnnnnntac... 8.47
...gtannnnnnnnnnntta.... .taannnnnnnnnnntac... 0.05
...taannnnnnnnnnnnnaca.. .tgtannnnnnnnnnntta.... 4.17
....taannnnnnnnnnnnncac. .gtgannnnnnnnnnntta.... 2.43
....taannnnnnnnnnntac... .gtannnnnnnnnnntta.... 0.05
....aaannnnnnnnnnnaca.. .tgtannnnnnnnnnnttt.... 2.64
agtgtaaannnnnnntttacact agtgtaaannnnnnntttacact 11.84 best
consensus
;
[... ]
;
; Isolated patterns: 1
; align rev_cpl score
accnnnnnnnnnnnnncat atgannnnnnnnnnnnnggt 0.27 isol
; Job started 27/01/06 22:28:52 CET
; Job done 27/01/06 22:28:54 CET

```

^aThree alternative assemblies were returned, all centred around the highest significant dyad ($GTAN_{11}TAC$).

3.4. Assembling the Discovered Patterns (Pattern-Assembly)

The pattern discovery tool, dyad-analysis, typically returns a list of dyads rather than a single one. Actually, these dyads generally show a strong mutual overlap, and they reveal different fragments of a same motif. The program pattern-assembly can be used to assemble overlapping dyads. This assembly step is automatically done in the web interface of dyad-analysis, and the result of pattern-assembly is displayed after the results of dyad-analysis (see Table 3). The best consensus obtained at the level of Gammaproteobacteria is **AGTGTAAANNNNNNTTTACACT**, which corresponds to the TyrR motif binding-site (matching letters are shown in bold) (13).

3.5. Pattern Matching (dna-pattern)

To map the discovered patterns on the input sequences, use dna-pattern and feature-map as described in Chapter 21.

- 1. Click on button “pattern matching (dna-pattern).”
- 2. Leave all other parameters unchanged and click “GO.”
- 3. Scroll down to the bottom of the page and click on “feature-map.”

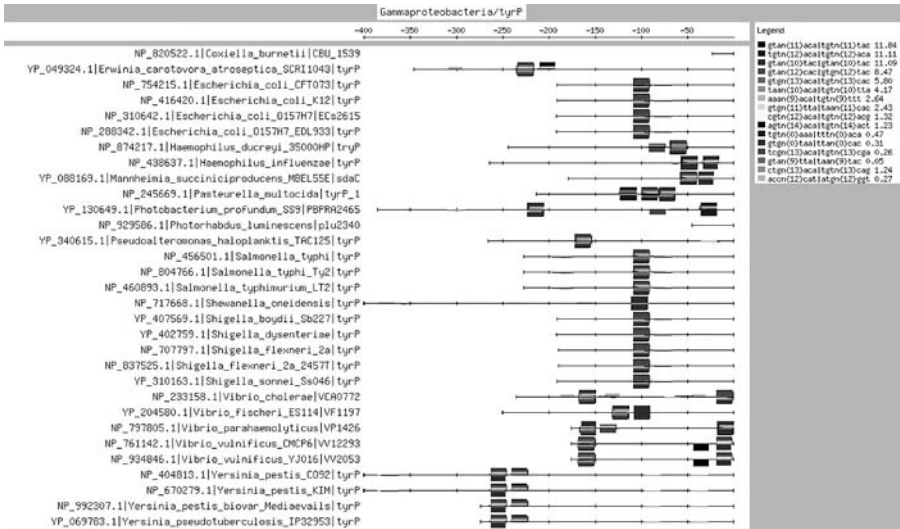


Fig. 1. Feature-map of the pattern-matching (using dna-pattern) of discovered significant patterns on the upstream sequences of all gammaproteobacterial orthologs. Each line represents one upstream sequence and discovered motifs are displayed on the feature-map as colored boxes with a height proportional to the significance score.

4. A form appears with the parameters of the feature-map tool. Leave the default parameters and click “GO.”
5. The feature-map resulting from the analysis of *tyrP* is displayed in [Fig. 1](#). Note that, as a result of publishing constraints, we modified the default parameters to gain in compactness, and to obtain a black and white figure.

4. Notes

1. Ortholog definition. Two genes are considered orthologous if they diverged from a common ancestor by speciation and conserved the original function. Orthology is usually inferred on the basis of sequence similarity. It is clear that such an inference is intrinsically error-prone, because similarity is per se neither a proof of common ancestry (the alternative explanation would be convergent evolution), nor for conserved function (which can only be tested experimentally).
2. Choice of a taxon. The choice of the appropriate taxon is crucial for the discovery of *cis*-acting elements. The taxon should encompass as many organisms as possible (to increase the signal, i.e., the number of regulatory elements), but these organisms should be close enough to have conserved the transcriptional regulation for the gene of interest. There is some robustness however: the pattern discovery method dyad-analysis accepts some rate of sequences without the motif. Thus, motifs can sometimes be detected at a higher taxonomical level, even if they are found only in one subbranch of the taxon. However, the significance of the motif will be reduced (the signal is “diluted”), and some motifs might be missed. We recommend to test several taxonomical levels, and to evaluate the optimal level of significance and coverage (*see* **Note 3**).
3. Getting more sequences. The signal-to-noise ratio of the pattern discovery method crucially depends on the number of available sequences. If the number of orthologs selected for a given gene is too low, several possibilities can be envisaged. (1) One possibility is to wait for a few years, until more genomes will be available for the taxon of interest, but this might pose problems for “hot” research topics. (2) Another possibility is to select a higher taxonomical level, but at some point the regulation will have diverged between the selected groups. (3) Yet another possibility is to be less stringent in the selection of putative orthologs, by allowing the selection of paralogs (homologous genes resulting from a gene duplication). In some cases, a genome might contain multiple copies of a gene, having conserved the function and regulation (e.g., isofunctional enzymes). One should however be very careful with this approach, because all paralogs are not necessarily isofunctional. In addition, in some cases, isofunctional enzymes show differential regulation (e.g., the three aspartate kinases in *E. coli* are regulated by methionine, lysine, and threonine, respectively). Depending on the cases, the inclusion of paralogs will thus increase either the signal, or on the contrary the noise. The program get-orthologs allows to retrieve paralogs, by relaxing the upper threshold

on the hit rank: the default value (rank $\leq$ 1) returns bidirectional best hits. Higher rank values return multiple bidirectional hits (i.e., putative orthologs and putative paralogs).

4. Feature type. When the feature type “CDS” is selected, coordinates are calculated from the start codon, i.e., the first letter of the start codon is the reference point (position 0), and upstream sequences are specified by entering negative coordinates. To detect transcription factor binding sites, it would be preferable to use the transcription start site as origin, but unfortunately, the mRNA are not annotated in prokaryote genomes. Pragmatically, we take thus the option to retrieve sequences upstream from the start codon.
5. Sequence sizes. For prokaryotes, the default upstream limits have been set from -400 to -1 , on the basis of a previous analysis by Julio Collado-Vides on the distribution of *cis*-acting elements in *E. coli* promoters (see **Note 19**).
6. Prevent overlap with upstream ORFs. When this option is checked, intergenic sequences are clipped according to the distance to the closest upstream neighbor gene, to avoid coding sequences. Note that in prokaryotes, many upstream sequences are much smaller than 400 bp. In particular, intergenic sequences located within an operon are generally very short (<50 bp). It is thus essential to activate this option “prevent overlap with upstream ORFs”.
7. Sequence label. Because the gene name is likely to be the same for the different orthologs, avoid the use of label “gene name”, which may cause problems in the feature-map drawing.
8. Link to the sequence file. By default, sequences are not displayed to minimize data transfer through the web browser, but if the connection is sufficiently fast, the link can be clicked to check the sequence. After having checked, come back to the get-orthologs result page.
9. Choice of the pattern discovery algorithm. The program consensus, developed by Jerry Hertz, also gives pretty good results with bacterial promoters. It is, thus, a good idea to test both approaches, and compare the results.
10. Sequence purging. In Chapter 21, we explained the importance of purging to avoid statistical biases in case of repeated sequences. Sequence purging is even more important for promoters of orthologous genes, because the collection of sequenced genomes contains several closely related strains (for example, there are currently five strains of the species *E. coli*). If the selected species are too close from each other, the whole promoter sequence will be conserved. Consequently, all the dyads will be found in multiple copies, but this will reveal redundancy in the sequences rather than over-representation of specific functional elements. Sequence purging is thus highly recommended for the pattern discovery stage (dyad-analysis). By default, the purging masks redundant fragments larger than 40 bp with less than three mismatches. In contrast, pattern matching and feature-map drawing are done with nonpurged sequences, to highlight all the candidate *cis*-acting elements.

11. Background model. We suggest here to use monad frequencies, where the estimation of a dyad probability is the product of frequencies of the two corresponding trinucleotides in the input sequences (e.g., $P[ATGn_3TAC] = F[ATG] * F[TAC]$). We also evaluated an alternative background model (taxon frequencies), where prior probabilities of dyads are estimated by calculating their frequencies in the whole set of upstream sequences of all genes of the considered taxon (Janky and van Helden, in preparation). As a general rule, the monad model is more stringent than the taxon model. The detected motifs are thus more reliable, but the cost is a loss of sensitivity for detecting less conserved motifs.

Acknowledgments

This work was supported by a doctoral grant from Fonds pour la Recherche dans l'Industrie et l'Agriculture (F.R.I.A.) to RJ and by the "BioSapiens Network of Excellence" funded under the sixth Framework programme of the European Communities (LSHG-CT-2003-503265). Genome installation and orthologs identification are done on a 40-node PC cluster contributed by various institutions, including the Belgian Fonds pour la Recherche Fondamentale Collective (F.R.F.C. grant 2005). We are grateful to Raphaël Leplae for his enthusiasm in obtaining, installing, and maintaining this cluster. JvH acknowledges Stéphane Vissers for an inspiring discussion on the importance of good quality protocols for teaching and practicing good science. We are thankful to Stephan Kurtz for making available his very efficient program vmatch, which we use to purge redundant fragments.

References

- 1 Tagle, D. A., Koop, B. F., Goodman, M., Slightom, J. L., Hess, D. L., and Jones, R. T. (1988) Embryonic epsilon and gamma globin genes of a prosimian primate (*Galago crassicaudatus*). Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.* **203**, 439–455.
- 2 Wasserman, W. W. and Fickett, J. W. (1998) Identification of regulatory regions which confer muscle-specific gene expression. *J. Mol. Biol.* **278**, 167–181.
- 3 Fickett, J. W., and Wasserman, W. W. (2000) Discovery and modeling of transcriptional regulatory regions. *Curr. Opin. Biotechnol.* **11**, 19–24.
- 4 Tompa, M. (2001) Identifying functional elements by comparative DNA sequence analysis. *Genome Res.* **11**, 1143–1144.
- 5 McGuire, A. M., Hughes, J. D., and Church, G. M. (2000) Conservation of DNA regulatory motifs and discovery of new motifs in microbial genomes. *Genome Res.* **10**, 744–757.
- 6 McCue, L., Thompson, W., Carmack, C., et al. (2001) Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Res.* **29**, 774–782.

- 7 Alkema, W. B., Lenhard, B., and Wasserman, W. W. (2004) Regulog analysis: detection of conserved regulatory networks across bacteria: application to *Staphylococcus aureus*. *Genome Res.* **14**, 1362–1373.
- 8 van Helden, J., Andre, B., and Collado-Vides, J. (1998) Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.* **281**, 827–842.
- 9 Hertz, G. Z., Hartzell, G. W., III, and Stormo, G. D. (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.* **6**, 81–92.
- 10 Hertz, G. Z. and Stormo, G. D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* **15**, 563–577.
- 11 Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262**, 208–214.
- 12 Neuwald, A. F., Liu, J. S., and Lawrence, C. E. (1995) Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci.* **4**, 1618–1632.
- 13 Andrews, A. E., Dickson, B., Lawley, B., Cobbett, C., and Pittard, A. J. (1991) Importance of the position of TYR R boxes for repression and activation of the *tyrP* and *aroF* genes in *Escherichia coli*. *J. Bacteriol.* **173**, 5079–5085.
- 14 Pittard, A. J. and Davidson, B. E. (1991) TyrR protein of *Escherichia coli* and its role as repressor and activator. *Mol. Microbiol.* **5**, 1585–1592.
- 15 Yang, J., Wang, P., and Pittard, A. J. (1999) Mechanism of repression of the *aroP* P2 promoter by the TyrR protein of *Escherichia coli*. *J. Bacteriol.* **181**, 6411–6418.
- 16 Whipp, M. J. and Pittard, A. J. (1977) Regulation of aromatic amino acid transport systems in *Escherichia coli* K-12. *J. Bacteriol.* **132**, 453–461.
- 17 Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410.
- 18 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 19 Perez-Rueda, E., and Collado-Vides, J. (2000) The repertoire of DNA-binding transcriptional regulators in *Escherichia coli* K-12. *Nucleic Acids Res.* **28**, 1838–1847.
- 20 van Helden, J., del Olmo, M., and Perez-Ortin, J. E. (2000) Statistical analysis of yeast genomic downstream sequences reveals putative polyadenylation signals. *Nucleic Acids Res.* **28**, 1000–1010.

PhyME

A Software tool for Finding Motifs in Sets of Orthologous Sequences

Saurabh Sinha

Summary

Discovery of transcription factor binding sites is a crucial and challenging problem in bioinformatics. Several computational tools have been developed for this problem, popularly known as the motif-finding problem. PhyME is an *ab initio* motif-finding algorithm, which finds overrepresented motifs in input sequences while accounting for their evolutionary conservation in orthologs of those sequences. Here, we describe the usage of this algorithm, publicly available as a Linux-based implementation.

Key Words: Motif finding; multiple species; evolutionary tree; expectation maximization.

1. Introduction

A common approach to understanding gene regulation involves computational discovery of transcription factor binding sites in promoter sequences of genes. In a common application scenario, we are given the promoters of a set of related genes from one species, along with some or all of their orthologous sequences from other species. The goal is to find binding site motifs in such heterogeneous sequence data, by combining two criteria: (1) overrepresentation, which is related to the number of occurrences of the motif in one species, and (2) conservation of each motif occurrence across multiple species. The PhyME software implements an algorithm (**1**) to solve this problem, using the method of Expectation-Maximization, and a probabilistic model of evolution of binding sites.

From: *Methods in Molecular Biology*, vol. 395: *Comparative Genomics, Volume 1*
Edited by: N. H. Bergman © Humana Press Inc., Totowa, NJ

For each gene's promoter, and corresponding orthologous sequences, the program first uses the Lagan alignment tool to identify ungapped blocks of sequence that are highly conserved across species. Such blocks are treated as having common evolutionary origin, and binding sites therein are assumed to be functional in all species. Motif occurrences outside of these blocks of conservation are treated as independent binding sites, and the overall score of a candidate motif includes separate contributions from intra- and interblock sequences. Motifs are modeled as position weight matrices and Expectation Maximization is used to find the highest scoring motif. PhyME is designed to be used in cases where orthologous sequences can be aligned, with the ungapped aligned blocks covering a moderate fraction (about 50 % or more) of the alignment.

2. Materials

1. The PhyME software may be downloaded as a "tar" archive from the website <http://veda.cs.uiuc.edu/cgi-bin/phyme/download.pl>. The following sections describe the use of this software in a Linux environment.

3. Methods

1. Install the PhyME software by following instructions in the distribution's README file, at some location hereafter referred to as "ROOT." (See **Notes 1–3** for troubleshooting and special options.)
2. Begin with the promoter sequences from each species stored in separate Fasta files (see **Note 4**). The name of a promoter is given by its Fasta identifier (the word immediately following the ">" in the header). Orthologous promoters in separate Fasta files must have identical names. One of the Fasta files must correspond to the "reference" species—the species with most complete sequence data. For every promoter corresponding to any of the other species, there must be an orthologous promoter in the reference species' Fasta file. The sample data set included with the distribution has four such Fasta files (cer.fna, kud.fna, bay.fna, mik.fna) in ROOT/data/, containing promoter sequences from four different species.
3. Create an output directory, e.g., ROOT/data/results/, henceforth called "OUTDIR." Run the Lagan alignment program on the Fasta files to align orthologous promoters, and extract ungapped conserved blocks. This is done via the program ROOT/code/helpers/preproc.pl, whose first argument is OUTDIR/, followed by each of the Fasta files, with the "reference" species' Fasta file appearing first (see **Note 5**). This step creates two files for each promoter included in the reference species' Fasta file:
 - a. File <promoterName>.fna is a Fasta file that contains that promoter's orthologs from each species, now renamed as "SPECIES_0," "SPECIES_1," and so on,

with the species number being determined by the order in which that species' Fasta file was provided to the `preproc.pl` program.

- b. File `<promoterName>.blk`, henceforth called a "blocks" file, contains information about conserved blocks between the reference species (`SPECIES_0`) and each of the other species.

The simple format followed by these two types of files allows the user to replace the Lagan based preprocessing step with the use of any other alignment software, as long as these files are in the correct format (*see* **Note 6**).

4. Create a phylogeny file, such as the sample file `ROOT/data/phylogeny_flat.txt`. This file captures evolutionary distances between the reference species (`SPECIES_0`) and each of the other species. The first number must be 0, representing the distance between `SPECIES_0` and itself, and the n^{th} number (separated by space) is the distance between `SPECIES_0` and `SPECIES_(n - 1)`. Each number must lie between 0 and 1, and represents the neutral substitution probability between a pair of species. In reality, these numbers vary from one part of the genome to another; however, PhyME requires the same number to be used for a particular pair of species, regardless of genomic location of the sequence. (*See* **Notes 7 and 8** for more details on the phylogeny file.)
5. Run PhyME from the command-line: `ROOT/code/bin/phyme <arguments>`. The compulsory arguments are the number of species (`-K`), the number of nonorthologous sequences (`-N`), along with the Fasta file and blocks file for each sequence (created in **step 3**), the motif length (`-w`), and the phylogeny file (`-pf`). To get a quick list of options, type `ROOT/code/bin/phyme` at the command-line. For more detailed descriptions of options, *see* **Notes 9–11** and the `ROOT/README` file that comes with the distribution. The output of PhyME is printed to the terminal; hence, use output redirection to a result file. PhyME also prints run-time messages to standard error, which may also be redirected to a file, in a shell-dependent manner. For example, in the bash shell, use `>` to redirect output, and `2>` to redirect run-time messages. Various log files are also created by PhyME, in the current directory—use `-od OUTDIR` to write these to the directory `OUTDIR`. (*See* **Note 12** for troubleshooting tips.)
6. Understanding PhyME's output: the description here can be best understood with reference to excerpts from a sample output given in **Fig. 1**, also available in the file `ROOT/data/sample_output/sites_phyme`. There are as many motifs output as specified by the `-nmotifs` option. For each such motif, PhyME first reports details about the motif itself, followed by a listing of its predicted sites.
 - a. Motif description: this begins with a header line of the form `"#Motif X: Score S1 S2,"` where `"X"` is the motif number. `S1` is the information score of this motif, which is higher for more specific (less fuzzy) motifs, for the same motif length. `S2` is the log likelihood ratio—the motif score that is maximized by PhyME's search algorithm. Following the header line, there is a line of the

```

*****
#Motif 1: Score 8.22 44.07
#>MOTIF_1_10
# 0.0000 0.0000 0.0000 0.9252 0.0742 0.8677 0.0000 0.0000 0.0000 0.0000
# 0.3107 0.1853 0.7904 0.0000 0.9258 0.0000 0.7263 0.0555 0.1343 0.0687
# 0.1065 0.3289 0.0000 0.0748 0.0000 0.1323 0.1174 0.8372 0.0701 0.7230
# 0.5828 0.4858 0.2096 0.0000 0.0000 0.0000 0.1563 0.1073 0.7956 0.2083
#<
#Consensus: TTCACACGTG
#Sites above threshold 0.30
ttcttcTCTACGGCGGgaagct >SPECIES_0_PHO8_data/cer.fna MOTIF_1 34 25 1.00 - Aligned
ttcttcTCCACGGCGGgaagct >SPECIES_1_PHO8_data/mik.fna MOTIF_1 36 27 1.00 - Aligned
ttcttcCCCACGGCGGgaagct >SPECIES_2_PHO8_data/kud.fna MOTIF_1 36 27 1.00 - Aligned
ttcttcTTCACGGCGGgaagct >SPECIES_3_PHO8_data/bay.fna MOTIF_1 48 39 1.00 - Aligned
tgagcCTCACACGTGgttgag >SPECIES_0_PHO5_data/cer.fna MOTIF_1 743 752 0.99 + Not Aligned
cgagcCTCACACGTGgttgag >SPECIES_1_PHO8_data/mik.fna MOTIF_1 100 109 0.99 + Aligned
cgtgcTCCACATTTTggcag >SPECIES_0_PHO8_data/cer.fna MOTIF_1 101 110 0.99 + Aligned
tcagcTGCACACTTTggcag >SPECIES_3_PHO8_data/bay.fna MOTIF_1 103 112 0.99 + Aligned
tgggcCTCACACGTGgggttc >SPECIES_3_PHO5_data/bay.fna MOTIF_1 58 67 0.99 + Not Aligned
tgagcCTTACACGTGgggttc >SPECIES_2_PHO5_data/kud.fna MOTIF_1 26 35 0.99 + Not Aligned
gggatCTCAACGTCGcatct >SPECIES_2_PHO5_data/kud.fna MOTIF_1 422 431 0.99 + Aligned
.
.
.
.
tggtcGTCCACAGGTgtgtgt >SPECIES_3_PHO81_data/bay.fna MOTIF_1 270 261 0.70 - Not Aligned
*****
*****
#Motif 2: Score 8.88 40.50
#>MOTIF_2_10
# 0.9280 0.0000 0.5019 0.0000 0.6513 0.0000 0.6715 1.0000 0.0000 0.0000
# 0.0000 0.0742 0.2502 0.4154 0.0000 0.0000 0.0960 0.0000 0.0000 0.8649
# 0.0720 0.0478 0.0000 0.0000 0.3487 0.0000 0.1620 0.0000 1.0000 0.0587
# 0.0000 0.8780 0.2479 0.5846 0.0000 1.0000 0.0705 0.0000 0.0000 0.0764
#<
#Consensus: ATATATAAGC
#Sites above threshold 0.30
cttcaATTCTGTAAGCcatga >SPECIES_2_PHO81_data/kud.fna MOTIF_2 132 141 1.00 + Aligned
.
.
.
.
aaatgAGCCACTGAGCttgtt >SPECIES_0_PHO81_data/cer.fna MOTIF_2 9 18 0.67 + Not Aligned
*****
*****
#Motif 3: Score 8.00 32.85
#>MOTIF_3_10
# 0.0920 0.0000 0.8030 1.0000 0.0002 0.5497 1.0000 0.4524 0.4014 0.0000
# 0.0000 0.7771 0.1970 0.0000 0.3743 0.4503 0.0000 0.2453 0.0000 0.4691
# 0.9080 0.2229 0.0000 0.0000 0.5299 0.0000 0.0000 0.0000 0.3100 0.0133
# 0.0000 0.0000 0.0000 0.0000 0.0956 0.0000 0.0000 0.3022 0.2886 0.5177
#<
#Consensus: GCAAGAAAAAT
#Sites above threshold 0.30
atagaGCAAGCAAAATtcgag >SPECIES_0_PHO5_data/cer.fna MOTIF_3 978 987 1.00 + Not Aligned
.
.
.
.
gtttaGGAACCACTCaatga >SPECIES_3_PHO81_data/bay.fna MOTIF_3 61 52 0.60 - Not Aligned
ctaacGCAATCAATCaacac >SPECIES_3_PHO81_data/bay.fna MOTIF_3 374 383 0.59 + Not Aligned
*****
*****

```

Fig. 1. Sample output of PhyME.

form “#>MOTIFNAME L,” where MOTIFNAME is the name given by PhyME to this motif, and *L* is its length. Following this are four lines specifying the position weight matrix of the motif. After a beginning character “#,” there is one column for each position of the motif. The rows represent the position specific frequencies of A, C, G, T, respectively. The position weight matrix description ends with a line “#<,” following which a simple consensus of the motif is printed.

- b. Motif sites: all sites of the motif, with posterior probability above a user specified threshold (*see* **Note 13**), are reported in separate lines. Each line has eight tab-separated fields, with the following semantics:

- i. Field 1: sequence of the putative site (in upper case), with flanking bases (in lower case).
- ii. Field 2: name of the input sequence (including species name) in which the site is present. For example ">SPECIES_0_PHO8_data/cer.fna" means that the predicted site is in the sequence named PHO8 of the species SPECIES_0.
- iii. Field 3: name of the motif.
- iv. Fields 4, 5: start and end offsets of the site from the beginning of the sequence. If the site is on the reverse strand (orientation "-" in Field 7), then the end offset is specified first.
- v. Field 6: posterior probability of the site. This may be treated as the degree of confidence in the prediction that this is a site, on a scale of 0 to 1.
- vi. Field 7: orientation ("+" or "-"), i.e., whether the motif matches the sequence on the forward or the reverse strand.
- vii. Field 8: whether this site is inside an aligned block (in which case it must be present with the same posterior probability in all sequences that share the aligned block), or outside.

4. Notes

1. Compilation problems: PhyME makes use of an external software library called "newmat." If this library does not compile, this may be because of incompatibility with newer versions of the gcc compiler. For the latest copy of this library, visit the developer Robert Davies' website (<http://www.robertnz.net/>), download the latest version of newmat, and install it at the location ROOT/code/lib/newmat/. If the compilation problem is with the "mlagan" code (third party software), install the latest version from the website (http://lagan.stanford.edu/lagan_web/index.shtml) in the directory ROOT/code/lib/mlagan/.
2. PhyME models background sequence with a Markov model, which has an order that can be chosen by changing the "-DMARKOV_ORDER=0" flag in the file ROOT/code/Makefile from the default 0th order to any integer. Typically, this should be in the range 0 to 3. A kth order Markov model means that all (k+1)-long substrings in the background sequence are counted for inferring neighboring nucleotide correlations. Using k=0 ignores all correlations, and may discover simple repeats (such as poly-T) as motifs. A very high value of k (e.g., above five) may lead to a model trained on statistically insufficient data.
3. If more than one motif is requested (using the "-nmotifs" command-line option, **Note 10**), PhyME finds these motifs sequentially, and before finding the next motif, it (partially) masks out all occurrences of the current motif that are above some threshold. This threshold can be changed from its default value of 0.5 by changing the "-DMASK_THRESHOLD=0.5" flag in the file ROOT/code/Makefile to use

any number between 0 and 1. A higher threshold means fewer motif occurrences will be masked out before the next round of motif finding. By default, the five most central bases of each qualifying motif occurrence are masked. This may be changed, also at compile time, using the “-D_MASK_THISMANYCENTRALBASES_=5” flag in the file ROOT/code/Makefile.

4. “Fasta” is a popular file format for DNA sequences. A Fasta format file may contain one or more sequences. Each sequence has a header line that begins with the character “>” followed (without spaces) by an identifier (e.g., name) for that sequence, and other descriptive text (if any) following the identifier. Following each header line are one or more lines of DNA sequence. See the file ROOT/data/cer.fna for an example.
5. The ROOT/code/helpers/preproc.pl Perl script uses an external library called “bioperl,” that is not included with the PhyME distribution. This library may be downloaded from <http://bio.perl.org/>, the user should install it at some location, say BIOPERLDIR, and add the line “use lib ‘BIOPERLDIR’;” in preproc.pl. Alternatively, the user may use the program ROOT/code/helpers/preproc_nobioperl.pl that does not use bioperl.
6. The program ROOT/code/helpers/preproc.pl aligns the input sequences using Lagan, and creates special format files to be used as input to PhyME. A different alignment procedure may be used in this preprocessing step, as long as the same set of files, in the same format, are created. The format is explained next:
 - a. <promoterName.fna>: this has all orthologous promoters for gene “promoterName,” in Fasta format. Each sequence has the corresponding species name (“SPECIES_0,” “SPECIES_1,” and so on) as its Fasta identifier.
 - b. <promoterName.blk>: this has information from pair-wise alignment of the promoter from the reference species (SPECIES_0) with that from each of the other species (SPECIES_1, SPECIES_2, and so on). A header line such as “>SPECIES_1” is followed by descriptions of conserved blocks between that species and the reference species. Each block’s description is on a separate line, with the format: (b0 e0)=(b1 e1) <Pct>, where *b0* and *e0* are the beginning and end offset of this block in the sequence from the reference species, *b1* and *e1* are corresponding offsets in the sequence from the other species, and <Pct> is the percentage identity of this block. (This last field is not used by PhyME, and may be any number.) See the file ROOT/data/sample_output/PHO5.blk for an example. Note that the header “>SPECIES_0” is missing, because each header announces a pair-wise alignment of the reference species (SPECIES_0) with one of the other species.
7. It is possible to use a more accurate phylogeny than the “flat” phylogeny described in **Subheading 3., step 4**. If the evolutionary relationship among the species is best captured by a particular binary tree, that tree may be input to PhyME, in a format very similar to the Newick tree format. **Figure 2** shows an example

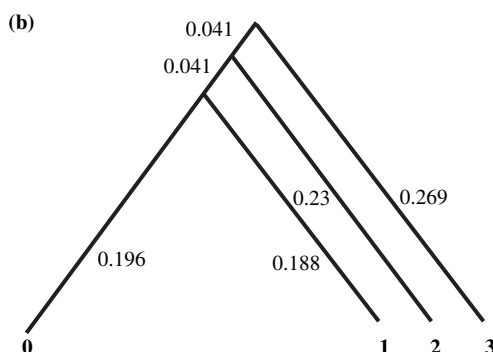


Fig. 2. Sample phylogeny input to PhyME, when using the “-tree” option. (A) Contents of a sample phylogeny file (“-pf”) and (B) phylogenetic tree that is represented by the file. Labels on leaf nodes (0, 1, 2, 3) correspond to the species (SPECIES_0, SPECIES_1, SPECIES_2, SPECIES_3, respectively). Edge labels represent neutral substitution probability on each branch of the tree.

of this format, corresponding to the sample file ROOT/data/phylogeny_tree.txt. A complete description of the Newick format is available at the website <http://evolution.genetics.washington.edu/phylip/newicktree.html>. The branch lengths in the tree input to PhyME are meant to represent neutral substitution probabilities for the respective branches, and hence must be real numbers between 0 and 1. Using the general tree phylogeny is usually much less efficient than the “flat” phylogeny format described in **Subheading 3.**, and should be used only when working with a small data set (≤ 5 species, less than 10,000 bp of sequence, depending on machine specifications). If using the tree phylogeny, do not forget to specify the option “-tree” in addition to the “-pf <phylogeny-file>” option.

8. The choice of numbers for branch lengths in the phylogeny is ad-hoc. One simple strategy would be to take a large number of aligned columns in a pair-wise alignment between the two species that the branch connects, and count what fraction of these is nonidentical. This fraction would be a ballpark figure for the branch length, although trial and error may be required before the user starts getting the optimal results.
9. PhyME scores a candidate motif by modeling the input sequence(s) as being generated by a probabilistic process that plants motif occurrences in a random background. An accurate characterization of typical promoter sequences (background) leads to more accurate probability scores. By default, PhyME uses each input sequence from the reference species to model the background for that sequence. The user may choose to over-ride this default behavior by using the “-b <fastafilename>” command-line option, whereby the entire sequence data in

<fastafilename> will be used to train the background model. This may be particularly useful if a higher order Markov background is being used (*see Note 4*), e.g., third order or higher, and each input sequence is statistically insufficient to train such a model. Roughly speaking, each of the 4^{k+1} possible $(k+1)$ -mers should occur at least a few times in the sequences to train a k^{th} order Markov background.

10. PhyME reports the highest scoring motif that it finds. However, this motif may not always be the true highest scoring motif in the search space, because the algorithm may converge to a “local optimum.”
 - a. One way to address this issue is to ask PhyME to report multiple motifs—use the “-nmotifs” option to specify how many motifs the user wants. PhyME will serially find motifs, and after reporting a motif, it will “mask out” the central base(s) in all occurrences of that motif, so that the same motif is not found again (*see also Note 3*). The running time will multiply by a factor equal to the number of motifs desired.
 - b. Another, quicker strategy to counter the local optimum problem is to use a large number in the “-niter” option. To find each motif, PhyME makes several “random re-starts”—it chooses a seed motif randomly from the input sequences and makes a fixed number of improvements (optionally specified by “-nseediter”) to this seed motif. (Each improvement modifies the motif to give a slightly better score.) After having made a fixed number of such random restarts (which is specified by “-niter”), PhyME picks the seed motif that led to the highest score, executes one final “re-start” from this seed motif, and makes as many improvements as possible, i.e., until the algorithm converges to a local optimum. A higher value of the “-niter” option leads to more seed motifs being tried, hence a greater portion of the search space is explored, and the chances of finding the global optimum are higher.
11. The probabilistic model used by PhyME assigns a fixed probability (say, x) to a motif instance being planted at any location, with a $(1 - x)$ probability of the location being background sequence. By default, PhyME trains this probability from the input data. However, the user may exercise some degree of control on this probability x in two ways:
 - a. The “-nsites N” option may be used to fix this probability to some value that is not changed by the algorithm. The specified integer N is divided by the total length of input promoters (in the reference species) to get the value of x , so that the expected number of sites of the motif is equal to N.
 - b. The “-maxsites N” option may be used to initialize the probability x (exactly as for the “-nsites” option), and the algorithm is then free to change the value of x to improve the score, without going above the initial value. This ensures that the expected number of sites is not more than N. Do not use the “-nsites” option and the “-maxsites” option together.

12. Segmentation fault: if this error happens upon program exit, PhyME crashed.
 - a. Check that all input files are in the correct format.
 - b. Check that the number of Fasta files given is equal to that specified by the “-N” option.
 - c. Check that the number of species (“-K”), number of sequences (“-N”), motif length (“-w”), and the phylogeny file (“-pf”) have been specified.
 - d. Send e-mail to the author’s current e-mail address, specifying the input used.
13. PhyME reports, for each motif, all of its sites with posterior probability above some threshold. This threshold may be specified as a command-line argument using the “-ot” option. Use a number between 0 and 1. The default value is 0.1.

References

- 1 Sinha, S., Blanchette, M., and Tompa, M. (2004) PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics* **5**, 170.

Comparative Genomics-Based Orthologous Promoter Analysis Using the DoOP Database and the DoOPSearch Web Tool

Endre Barta

Summary

Bioinformatic and experimental analyses of promoter regions are available for a long time. Finding of the transcription factor binding sites (TFBSs), however, by either method still faces a number of problems. For example, because of the ambiguity of binding of transcription factors, the number of false-positives and -negatives can be unexpectedly high in these sequence analyses. We can assume that evolutionary conserved motifs or regions in the promoters of the homologous genes function as TFBSs. Thus, a comparative genomic approach can provide a partial resolution for the problem previously outlined.

This chapter describes application of the DoOP database and the DoOPSearch web tools for such a comparative genomic analysis. Orthologous promoter sequences and conserved motifs can be extracted from the DoOP database for further analysis. The web-based tools of the DoOPSearch webpage can be used for searching and comparing conserved motifs. Using these tools, it is possible to compare short sequences with conserved motifs, to map conserved motifs into a longer promoter region, or find sequence patterns in different sets of promoter sequences.

Key Words: Promoter sequences; orthologous promoters; conserved motifs; motif search; transcription regulation.

1. Introduction

The first and most well-known promoter database, the Eukaryotic Promoter Database (EPD) ([1](#)) consists of promoter sequences extracted from the up- and downstream regions of either experimentally or *in silico*-determined transcription

start sites (TSSs). Although EPD describes promoters very precisely, the number of entry records is still very limited. In the genomic era, when full genomes are sequenced rapidly, growing number of sequence data are available for *in silico* analysis. Comparative genomics using bioinformatic methods is a mean to extract and compare promoter regions of homologous genes to see which regions or smaller motifs are evolutionary conserved. To study (determining, comparing, clustering, and so on) these conserved sequences is one of the most challenging task of comparative genomics in these days (2,3).

1.1. Promoter Databases

There are several promoter databases available for searching and retrieving promoter sequences. Link collections in the internet such as the <http://apollo11.isto.unibo.it/Databases.htm>, <http://databases.biomedcentral.com/>, or the NAR database collection, <http://www.oxfordjournals.org/nar/database/c> list those databases. Many of these databases, however, are not designed for comparative genomic analysis or contain only promoter sequences from a limited number of species. In this respect, the most comprehensive orthologous promoter collection is the DoOP database (4).

1.1.1. The DoOP Promoter Database

The two DoOP databases are based on the annotation of two well-known species, *Homo sapiens* and *Arabidopsis thaliana*. To build these databases the annotated first or in some cases the first two exons were used to find the first exons of homologous genes. The 5' upstream regions of the homologous genes are then used as orthologous promoter sequences. In most cases, this method gives reliable results, but it still has its drawbacks:

1. The method is heavily depending on the annotation of the model organism. If the annotation is wrong (i.e., for example there is an additional exon *in vivo* before the first annotated exon), then the extracted promoter sequence might not contain the real promoter. It is very likely though that annotation of the genes in model organisms will be more and more precise.
2. In most cases, the promoter regions used in DoOP database does not mean the 5' upstream region relative to the TSS, but it also contains the 5' untranslated region. It must be mentioned, however, that the positions of known TSSs are annotated if available.
3. The effectiveness of the method is relatively low. Only about 50 % of the human genes gives an orthologous promoter from a nonprimates species. It is possible however to use homologous gene annotations from other methods like the ENSEMBL (5) to determine the position of orthologous promoters and, thus, to increase the number of useable promoter clusters in the database.

1.2. Transcription Factor Binding Sites and Conserved Motifs Collections

There are several transcription factor binding sites database available on the internet for searching like TRANSFAC (6) or for downloading like JASPAR (7). Their data come mostly from manual curation of experimental data. Most recently, the first collections of evolutionary conserved motifs in the promoter regions become available in these databases. Xie et al. (8) used a statistical method to find conserved motifs in the promoter regions of human, dog, mouse, and rat homologous genes. The cisRED (9) and the CORG (10) databases employ the ENSEMBL annotation to find and analyze promoter regions. The consensus motif sequences of DoOP (4) database are generated by extracting the conserved parts of the dialing promoter alignments. There are a number of websites, like the TRED (11), available where uploaded sequences can be searched and analyzed for different motifs. At the time of writing, however, the DoOPSearch is the only website, where it is possible to search a conserved motifs database with a user supplied sequence.

1.2.1. The DoOPSearch Website

The DoOPSearch web tools were designed to find similarities in either short or long sequences to conserved or not conserved short DNA sequences or motifs from the promoter region of genes. The search for similar motifs is performed in two steps. In the first step both the query and all motif consensus sequences are split into overlapping pieces (“wordsize”) of a given length. These segments are then compared one by one with the program MOFEXT (MOtiF sEarch and eXTension) using a scoring matrix. If the calculated score of a pair of segments is above a given limit (cutoff) then the MOFEXT program tries to extend the alignment using the original query and the motif sequence. The result is the best and longest alignment between the user-supplied query sequence and the consensus motif in the database. The DoOPSearch website offers also a simple pattern search method in all promoter sequences using the FUZZNUC program from the EMBOSS (12) package.

1.3. Methods Described in This Chapter (see Note 1)

Using DoOP and DoOPSearch it is possible to:

1. Search and retrieve orthologous promoter sequences from the DoOP database.
2. Find and retrieve conserved motifs from the DoOP database.

3. Search the conserved motifs consensus list of DoOP database for similar motifs.
4. Search the promoter sequences available in the DoOP database for similar patterns.

The retrieved data can be (1) a set of promoter sequences, which can be analyzed further with different bioinformatic tools, (2) a list of genes that contain similar conserved motifs to the query sequence, and (3) a list of genes that contain similar patterns in their promoter region to the query sequence. Besides these data, the websites also provide a starting point for further analysis, because it contains links and cross-references to other databases like ENSEMBL, GOA, or EPD.

2. Materials

1. Hardware: any type of computer with graphical display and internet connection.
2. Software: a web browser with javascript capability.

3. Methods

3.1. Using the DoOP Database

From the DoOP database one can retrieve the promoter region (*see Note 2*) of a given gene or genes and their orthologs. If downloaded, these sequences can be also used in any other type of bioinformatics analysis such as primer design or sequence analysis.

3.1.1. Selecting Promoter Sequences From the Database

1. Open the DoOP homepage (<http://doop.abc.hu>) in the web browser.
2. Select the desired taxonomic category (*see Note 3*) and click on the “use this database” button.
3. In the search page fill out one of the fields to select one or more genes:
 - a. Enter the Cluster ID to the first field (if known from a previous search).
 - b. Type a gene ID in the second field. This is a unique short name of genes. In case of chordates, this is the HGNC name (<http://www.gene.ucl.ac.uk/nomenclature/index.html>).
 - c. Choose from a list of human ENSEMBL (ENSG...) or *Arabidopsis* (At...) IDs.
 - d. Type a keyword into the fourth field to search in the short description of genes.
 - e. Choose a species from a list. This option is useful if to get a promoter sequence from a gene of a rare species, or to see each promoter sequences of a given species from the DoOP database.
 - f. Use a Gene Ontology (GO) term or category to select one or more gene. Here, the user may either type directly an exact GO term or GO ID (GO:...), or after typing a keyword, go to a separate page and choose between the available GO terms that contain that keyword.

- g. Use the final option to find a promoter using sequence similarity (BLAT) search. Enter or upload preferably a human or *Arabidopsis* promoter or cDNA sequence and as all in the above cases push the appropriate search button.
4. Click the Search button beside the chosen field to get the result in the Table View page.

3.1.2. Download Promoter Sequences

After the search for the desired gene(s), in the TableView page;

1. Select one or more or all gene(s).
2. Choose between to download only the promoter sequences of the given model organism (*H. sapiens* for chordates or *A. thaliana* for plants) or to download promoter sequences from all the available species.
3. Choose one or more promoter length (500, 1000, and 3000 bp at the time of writing) to download (see **Note 4**).
4. Click the Download button to get navigated to the download page.
5. Download the files by clicking on them one by one (see **Note 5**).

Or to download only the sequence(es) of one cluster:

1. Click the cluster ID (8...) of the desired gene.
2. In the ClusterView page find the Files box and either click the Sequences link and then copy and paste the sequences, or use the mouse right button to save the target (see **Note 6**).

3.1.3. Getting Conserved Motifs

1. Navigate into the ClusterView page of the desired gene using the method previously outlined.
2. In the bottom of the page there is the graphical representation of the promoter sequences of the cluster. Click the chosen motif box to get into the MotifView page.
3. Here, the user can:
 - a. Copy and paste the motif sequences or the consensus.
 - b. See and then save by copying and pasting the position-specific weight matrix (PSWM) of the motif if available by clicking the PSWM button (see **Note 7**).
 - c. See and then save by copying and pasting the sequence logo of the motif if available (see **Note 7**).

3.2. Searching for Similar Conserved Motifs Using the DoOPSearch Web-Based Tool

One can search the consensus sequences of conserved motifs coming from the DoOP database. Either a shorter (for example, a transcription factor

binding sites) or a longer (for example, an experimentally proven promoter region) sequence, or any consensus sequence that is already in the DoOP database can be used. This is a sequence similarity search where choosing the appropriate parameters is very important (*see Note 8*).

3.2.1. MOFEXT Search With an Annotated Motif From the DoOP Database

1. Select a conserved motif of a given gene from the DoOP database following the previously outlined method starting from the DoOP database homepage (<http://doop.abc.hu>).
2. On the Motifview page either:
 - a. Click the “Run default search with consensus” button to perform an automatic search with default parameters (*see Note 8* and continue with **Subheading 3.2.3.**).
 - b. Click the “Go to search page with this consensus” to paste the consensus sequence of the chosen motif into the appropriate search field of the DoOPSearch website (*see Note 9* and continue with **Subheading 3.2.2.4.**).

3.2.2. MOFEXT Search With a Sequence Pattern

1. Open the DoOPSearch homepage (<http://doops.abc.hu>).
2. Select the desired taxonomic category (*see Note 3*).
3. In the search page type or copy and paste the sequence pattern into the search field (*see Note 9*).
4. Change the parameters to refine the search (*see Note 8*).
5. The user should enter an e-mail address to get the link pointing to the results by e-mail (*see Note 10*).
6. Click on the submit button to send the job. Continue with **Subheading 3.2.3.**

3.2.3. Analyzing the MOFEXT Search Result

After completing the previously described search, the user will be navigated to the TableView page. The resulted hits on this page are sorted by default according to their extended score (i.e., the second score that have been calculated by the MOFEXT program). In this page, it is possible to see the gene clusters (orthologous promoters that belong to one gene) from which the hits (conserved motifs from the motiflists) are originated, the alignments between the query and the hits, or to perform several filtering function to refine the result lists.

1. Click on the Cluster ID (on the first column) to see in the ClusterView page the highlighted motif (the hit) in the picture, and the information from the DoOP database about the given gene and its promoter region.

2. Click on the alignment (the last column) link to see the alignment between the query sequence and the given motif (the hit).
3. To perform a filtering or sorting function on the result (*see Note 11*):
 - a. Select between the available filtering options (like score, extended score, starting position on the query or length of the hit) in the pulldown menu.
 - b. Or type a GO ID or type a GO term keyword, and in the next page click on the appropriate GO category to get back with the previous page with the selected GO ID pasted into the GO ID field.

And click on the submit button.

4. If there is a picture at the top of the page (i.e., the query is longer than 20 bp), click on a position in the graph to list only hits that are presented at that position.

3.3. FUZZNUC Searching of Whole Promoter Sequences With the User's Pattern

1. Open the DoOPSearch homepage (<http://doops.abc.hu>).
2. Select the desired taxonomic category (*see Note 3*).
3. Type the query pattern in the pattern field of the FUZZNUC box (*see Note 12*).
4. Either use the default parameters or change the desired promoter set, number of mismatches, or the searching the complement sequence option.
5. Click the submit button to see the result in the next page (TableView page).
6. Click the Cluster ID (on the first column) to see in the ClusterView page the position of the given hit in the promoter sequences picture, and the information from the DoOP database about the given gene and its promoter region.
7. Click the Seq ID (the second column) to get the fasta format sequence of the given promoter.
8. To perform a filtering or sorting function on the result (*see Note 11*):
 - a. Select between the available filtering options (like Cluster ID, starting position on the hit) in the pulldown menu.
 - b. Or type a GO ID or type a GO term keyword, and in the next page click on the appropriate GO category to get back with the previous page with the selected GO ID pasted into the GO ID field.
 - c. Or type a mismatch value to show only hits with less mismatch.
 - d. Or select the strand to show only hits on that strand.

And click on the submit button.

Notes

1. Both the DoOP database and the DoOPSearch web tool are under constant development. It is possible therefore that the look and the content of one or more webpage will change or new features will be implemented. However, the main

methods that are mentioned in this chapter will be most likely available in the same way as now.

2. The term “promoter region” in this aspect means the upstream genomic sequence relative to the known and annotated translational start point (AUG codon) of a gene, or the beginning of the first totally untranslated exon if exists. In the first case, the whole 5’ untranslated region will be represented in the given promoter sequence, whereas in the second case only the strict promoter region. The reason for this difference is technical (for details *see* **ref. 4**).
3. At the time of writing only two categories available for searching, the chordates (based on the human annotation), and the plants (based on the *A. thaliana* annotation). Other databases, such as the yeast (based on the annotation of *Saccharomyces cerevisiae*) and the insect (based on the annotation of *Drosophila melanogaster*) will be also available in the near future.
4. It is common that the longer promoter clusters contain less orthologs. The reason for this is that if the available promoter sequence of a gene from a given species is shorter then for example 700 bp (most of the single read genomic sequences fall into this category), it will not get into the 1000 bp promoter cluster.
5. The sequences are available for downloading in multiple fasta format. The fasta headers contain the unique accession number of the sequence, the type of the gene (1–4, 5n, and 6n, *see* **ref. 4**), and the name of the species. If there are more then one cluster available to download then the sequences are available in tarred and gzipped formats too.
6. In the “Files” box there are options to see and download the DIALIGN aligned sequences either in multiple fasta or dialign format.
7. PSWM matrices and sequence logos are available from the short high quality motifs.
8. In general, using the default parameters a quick search can be performed, which is sufficient to get an idea about the result to be expected. If this preliminary result is promising, then it is worth to try to obtain data using the other available parameters.

There are our main parameters, which can effect the result significantly:

- a. The word size.
- b. The cutoff score.
- c. The scoring matrix used in the search.
- d. The motif list used as database for the search.

The word size can be as low as 6 bp. In this case a 90 % cutoff value will result only hits with a perfect match, but with a longer word size the same cutoff will allow one or more mismatches or ambiguities.

The cutoff score affecting the number of hits obtained in the first step. Sometimes it is a good strategy, especially in the case of longer query sequences, to use a lower cutoff value, and then filtering the result using the extended score.

There are two matrices available at the time of writing (it is expected that this number will increase). It is important to note that the EDNAFULL matrix coming from the EMBOSS program package does not make difference between the small and capital letters although they have different meaning in the DoOP consensus sequences (for explanation *see* <http://doop.abc.hu/details.html>).

It is worth considering which motiflists will be used in a search. For example, using motifs from low complexity clusters (i.e., where the consensus is coming from close relative species like primates) will result in search of almost the entire promoter region, not only the conserved motifs. It is also meaningless to use motifs of the 1000 or 3000 bp promoter sequences, if the query is a core promoter element.

9. Here, one can enter not only letters for nucleic acid bases (ACGT or acgt) but also capital letters for consensus bases (for example R for purins).
10. Under certain circumstances (such as long query sequence; big difference between the length of the query and the word-size; low cutoff value, larger number of motiflists are in use), the completion time of the search is tend to be long (inextreme cases can be as long as 1 h). In these cases, it is safer and more convenient to get a link pointing to the result instead of keeping the browser window open to wait for the job to finish.
11. It is a good strategy both in the case of MOFEXT and the FUZZNUC search, to try a first run with a rather loose options (i.e., allow lower cutoff or higher mismatch value), and then refine the result using one of the filtering options.
12. It is possible to enter either a consensus sequence like in the case of MOFEXT search (*see* **Note 9**) or the FUZZNUC style pattern. For details see the FUZZNUC documentation (<http://emboss.sourceforge.net/apps/fuzznuc.html>).

Acknowledgments

I am grateful to Dr. Ferenc Marincs for critical reading of the manuscript and for his helpful suggestions.

References

- 1 Schmid, C. D., Perier, R., Praz, V., and Bucher, P. (2006) EPD in its twentieth year: towards complete promoter coverage of selected model organisms. *Nucleic Acids Res.* **34**, D82–D85.
- 2 Papatsenko, D. and Levine, M. (2005) Computational identification of regulatory DNAs underlying animal development. *Nat. Methods* **2**, 529–534.
- 3 Prakash, A. and Tompa, M. (2005) Discovery of regulatory elements in vertebrates through comparative genomics. *Nat. Biotechnol.* **23**, 1249–1256.
- 4 Barta, E., Sebestyón, E., Pálffy, T. B., Téth, G., Ortutay, C. P., and Patthy, L. (2005) DoOP: Databases of Orthologous Promoters, collections of clusters of orthologous upstream sequences from chordates and plants. *Nucleic Acids Res.* **33**, D86–D90.

- 5 Birney, E., Andrews, D., Caccamo, M., et al. (2006) Ensembl 2006. *Nucleic Acids Res.* **34**, D556–D561.
- 6 Matys, V., Kel-Margoulis, O. V., Fricke, E., et al. (2006) TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**, D108–D110.
- 7 Vlieghe, D., Sandelin, A., De Bleser, P. J., et al. (2006) A new generation of JASPAR, the open-access repository for transcription factor binding site profiles. *Nucleic Acids Res.* **34**, D95–D97.
- 8 Xie, X., Lu, J., Kulbokas, E. J., Golub, T. R., et al. (2005) Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. *Nature* **434**, 338–345.
- 9 Robertson, G., Bilenky, M., Lin, K., et al. (2006) cisRED: a database system for genome-scale computational discovery of regulatory elements. *Nucleic Acids Res.* **34**, D68–73.
- 10 Dieterich, C., Grossmann, S., Tanzer, A., et al. (2005) Comparative promoter region analysis powered by CORG. *BMC Genomics* **6**, 24.
- 11 Zhao, F., Xuan, Z., Liu, L., and Zhang, M. Q. (2005) TRED: a Transcriptional Regulatory Element Database and a platform for in silico gene regulation studies. *Nucleic Acids Res.* **33**, D103–D107.
- 12 Rice, P., Longden, I., and Bleasby, A. (2000) EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet.* **16**, 276–277.

Discovery of Motifs in Promoters of Coregulated Genes

Olivier Sand and Jacques van Helden

Summary

We present a method to predict *cis*-acting elements by detecting over-represented motifs in promoters of a set of coregulated genes (single-genome, multigenes approach). The method has been used successfully to detect regulating elements in bacteria and yeast. It can be used with higher organisms as well, but with a loss in reliability of the predictions. A web interface is available at the Regulatory Sequence Analysis Tools site (<http://rsat.scmbb.ulb.ac.be/rsat/>).

Key Words: Transcriptional regulation; pattern discovery; coexpressed genes.

1. Introduction

1.1. Context

In this paper, we describe a method to predict *cis*-acting elements by discovering over-represented motifs in promoters of coregulated genes of a given organism (single-genome, multigenes approach). The method relies on the detection of statistically over-represented motifs. Two types of elements can be detected: oligonucleotides and dyads (a dyad is defined here as a pair of short oligonucleotides separated by a spacing of fixed width but variable content). Distinct tools were designed to detect each type of element: oligo-analysis (1) and dyad-analysis (2), respectively.

In Chapter 18, we describe another type of application, where dyad-analysis is applied to the promoters of a set of orthologs of a single gene, to detect conserved elements putatively involved in its regulation (single-gene, multigenome approach).

1.2. Study Cases

We describe here a method to discover *cis*-regulatory motifs from a set of coregulated genes. A typical example of utilization is the prediction of regulatory signals involved in the regulation of a cluster of coexpressed genes obtained from microarray experiments. As study cases, we selected two clusters of genes showing a significant transcriptional response in two of the microarray experiments published by Gasch et al. (3).

The first set regroups 27 genes upregulated after 30' nitrogen depletion. The description of these genes is given in Table 1. Not surprisingly, this list includes several enzymes and permeases involved in nitrogen and amino acid metabolism.

The second set contains 18 genes significantly downregulated when glucose is provided as carbon source (Table 2).

2. Materials

1. Web server. The Regulatory Sequence Analysis Tools (RSAT) (4) server runs on a single processor PC under Linux operating system. The main server is at <http://rsat.scmbb.ulb.ac.be/rsat/>. Several mirrors are installed in various countries, and can be reached from the home page of the main server.
2. Genome sources. Most genomes were obtained from the National Center for Biotechnology Information (NCBI) genome repository (<ftp://ftp.ncbi.nih.gov/genomes/>). Genomes of higher organisms were obtained from EnSEMBL (<http://www.ensembl.org/index.html>). All genomes are preprocessed and installed as follows.
 - a. The NCBI flat files (format .gbk) are parsed and converted to raw sequence files (one per contig/chromosome) and a table of features indicating the position and description of each gene. For EnSEMBL distribution, the sequences are obtained directly via the EnSEMBL programmatic interface, and stored on the RSAT server.
 - b. A list of gene names (synonyms) is generated from the annotations.
 - c. Oligonucleotide and dyad frequencies are computed on the set of all upstream sequences. These genome-scale frequencies are then used to estimate prior probabilities for oligonucleotides and dyads.
3. Organization of the tools. The tools are organized in a modular way: rather than having a single form for the complete analysis, we found it more convenient to present separate forms for the successive steps of a given analysis. The RSAT home page (<http://rsat.scmbb.ulb.ac.be/rsat/>) displays two frames. The left frame presents a menu of the available tools, the right frame displays the forms and result pages.

Table 1
Study Case 1^a

<i>ADE17</i>	Enzyme of <i>de novo</i> purine biosynthesis containing both 5-aminoimidazole-4-carboxamide ribonucleotide transformylase and inosine monophosphate cyclohydrolase activities, isozyme of Ade16p; ade16 ade17 mutants require adenine and histidine
<i>ARG1</i>	Argininosuccinate synthetase, catalyzes the formation of L-argininosuccinate from citrulline and L-aspartate in the arginine biosynthesis pathway; potential Cdc28p substrate
<i>ARG3</i>	Ornithine carbamoyltransferase (carbamoylphosphate:L-ornithine carbamoyltransferase), catalyzes the sixth step in the biosynthesis of the arginine precursor ornithine
<i>ARG5,6</i>	Protein that is processed in the mitochondrion to yield acetylglutamate kinase and N-acetyl- γ -glutamyl-phosphate reductase, which catalyze the second and third steps in arginine biosynthesis; enzymes form a complex with Arg2p
<i>DAL2</i>	Allantoicase, converts allantoate to urea and ureidoglycolate in the second step of allantoin degradation; expression sensitive to nitrogen catabolite repression and induced by allophanate, an intermediate in allantoin degradation
<i>DAL3</i>	Ureidoglycolate hydrolase, converts ureidoglycolate to glyoxylate and urea in the third step of allantoin degradation; expression sensitive to nitrogen catabolite repression
<i>DAL5</i>	Allantoin permease; ureidosuccinate permease; expression is constitutive but sensitive to nitrogen catabolite repression
<i>DAL7</i>	Malate synthase, role in allantoin degradation unknown; expression sensitive to nitrogen catabolite repression and induced by allophanate, an intermediate in allantoin degradation
<i>DAL80</i>	Negative regulator of genes in multiple nitrogen degradation pathways; expression is regulated by nitrogen levels and by Gln3p; member of the GATA-binding family, forms homodimers and heterodimers with Deh1p
<i>DUR3</i>	Plasma membrane urea transporter, expression is highly sensitive to nitrogen catabolite repression and induced by allophanate, the last intermediate of the allantoin degradative pathway
<i>GAP1</i>	General amino acid permease; localization to the plasma membrane is regulated by nitrogen source
<i>HSP12</i>	Plasma membrane localized protein that protects membranes from desiccation; induced by heat shock, oxidative stress, osmotic stress, stationary phase entry, glucose depletion, oleate and alcohol; regulated by the HOG and Ras-Pka pathways

(Continued)

Table 1
(Continued)

<i>LEU1</i>	Isopropylmalate isomerase, catalyzes the second step in the leucine biosynthesis pathway
<i>LEU2</i>	β -isopropylmalate dehydrogenase, catalyzes the third step in the leucine biosynthesis pathway
<i>MEP2</i>	Ammonium permease involved in regulation of pseudohyphal growth; belongs to a ubiquitous family of cytoplasmic membrane proteins that transport only ammonium (NH_4^+); expression is under the nitrogen catabolite repression regulation
<i>MET10</i>	Subunit α of assimilatory sulfite reductase, which is responsible for the conversion of sulfite into sulfide
<i>MET16</i>	3'-phosphoadenylylsulfate reductase, reduces 3'-phosphoadenylyl sulfate to adenosine-3',5'-bisphosphate and free sulfite using reduced thioredoxin as cosubstrate, involved in sulfate assimilation and methionine metabolism
<i>MET17</i>	O-acetyl homoserine-O-acetyl serine sulphydrylase, required for sulfur amino acid synthesis
<i>MET3</i>	ATP sulfurylase, catalyzes the primary step of intracellular sulfate activation, essential for assimilatory reduction of sulfate to sulfide, involved in methionine metabolism
<i>MHT1</i>	S-methylmethionine-homocysteine methyltransferase, functions along with Sam4p in the conversion of S-adenosylmethionine (AdoMet) to methionine to control the methionine/AdoMet ratio
<i>NCE103</i>	Carbonic anhydrase; poorly transcribed under aerobic conditions and at an undetectable level under anaerobic conditions; involved in non-classical protein export pathway
<i>SER3</i>	3-phosphoglycerate dehydrogenase, catalyzes the first step in serine and glycine biosynthesis; isozyme of Ser33p
<i>SER33</i>	3-phosphoglycerate dehydrogenase, catalyzes the first step in serine and glycine biosynthesis; isozyme of Ser3p
<i>STR3</i>	Cystathionine β -lyase, converts cystathionine into homocysteine
<i>SUL1</i>	High affinity sulfate permease; sulfate uptake is mediated by specific sulfate transporters Sul1p and Sul2p, which control the concentration of endogenous activated sulfate intermediates
<i>SUL2</i>	High affinity sulfate permease; sulfate uptake is mediated by specific sulfate transporters Sul1p and Sul2p, which control the concentration of endogenous activated sulfate intermediates
<i>YBR147W</i>	Putative protein of unknown function; YBR147W is not an essential gene; resistant to fluconazole

^aThe 27 genes upregulated in the dataset from **ref. 3** after 30' of nitrogen depletion.

Table 2
Study Case 2^a

<i>AGX1</i>	Alanine: glyoxylate aminotransferase, catalyzes the synthesis of glycine from glyoxylate, which is one of three pathways for glycine biosynthesis in yeast; has similarity to mammalian and plant alanine: glyoxylate aminotransferases
<i>CYC1</i>	Cytochrome c, isoform 1; electron carrier of the mitochondrial intermembrane space that transfers electrons from ubiquinone-cytochrome c oxidoreductase to cytochrome c oxidase during cellular respiration
<i>ECM13</i>	Nonessential protein of unknown function
<i>FBP1</i>	Fructose-1,6-bisphosphatase, key regulatory enzyme in the gluconeogenesis pathway, required for glucose metabolism
<i>FMP43</i>	The authentic, non-tagged protein was localized to mitochondria
<i>GAL1</i>	Galactokinase, phosphorylates α -D-galactose to α -D-galactose-1-phosphate in the first step of galactose catabolism; expression regulated by Gal4p
<i>GAL10</i>	UDP-glucose-4-epimerase, catalyzes the interconversion of UDP-galactose and UDP-D-glucose in galactose metabolism; also catalyzes the conversion of α -D-glucose or α -D-galactose to their beta-anomers
<i>GAL2</i>	Galactose permease, required for utilization of galactose; also able to transport glucose
<i>GAL7</i>	Galactose-1-phosphate uridyl transferase, synthesizes glucose-1-phosphate and UDP-galactose from UDP-D-glucose and α -D-galactose-1-phosphate in the second step of galactose catabolism
<i>HSP12</i>	Plasma membrane localized protein that protects membranes from desiccation; induced by heat shock, oxidative stress, osmotic stress, stationary phase entry, glucose depletion, oleate, and alcohol; regulated by the HOG and Ras-Pka pathways
<i>HXT6</i>	High-affinity glucose transporter of the major facilitator superfamily, nearly identical to Hxt7p, expressed at high basal levels relative to other HXTs, repression of expression by high glucose requires SNF3
<i>INH1</i>	Protein that inhibits ATP hydrolysis by the F1F0-ATP synthase, inhibitory function is enhanced by stabilizing proteins Stf1p and Stf2p; has similarity to Stf1p and both Inh1p and Stf1p exhibit the potential to form coiled-coil structures

(Continued)

Table 2
(Continued)

<i>JEN1</i>	Lactate transporter, required for uptake of lactate and pyruvate; expression is derepressed by transcriptional activator Cat8p under nonfermentative growth conditions, and repressed in the presence of glucose, fructose, and mannose
<i>MLS1</i>	Malate synthase, enzyme of the glyoxylate cycle, involved in utilization of non-fermentable carbon sources; expression is subject to carbon catabolite repression; localizes in peroxisomes during growth in oleic acid medium
<i>PCK1</i>	Phosphoenolpyruvate carboxykinase, key enzyme in gluconeogenesis, catalyzes early reaction in carbohydrate biosynthesis, glucose represses transcription and accelerates mRNA degradation, regulated by Mcm1p and Cat8p, located in the cytosol
<i>RAM1</i>	Beta subunit of the CAAX farnesyltransferase (FTase) that prenylates the a-factor mating pheromone and Ras proteins; required for the membrane localization of Ras proteins and a-factor; homolog of the mammalian FTase β subunit
<i>RIB2</i>	DRAP deaminase, catalyzes the third step of the riboflavin biosynthesis pathway; cytoplasmic tRNA pseudouridine synthase involved in pseudouridylation of cytoplasmic tRNAs at position 32
<i>YFR038W</i>	hypothetical protein

^aThe 18 genes downregulated in the dataset from **ref. 3** when glucose is provided as carbon source.

3. Methods

The typical analysis, which is described in this protocol, consists in using successively different tools to go from a list of genes to a graphical map showing the instances of the significant motifs (sequence retrieval → pattern discovery → pattern matching → feature-map). For this purpose, the tools are interconnected: the result of one tool can be sent as input for the next tool (piping).

3.1. Retrieve Sequence

This program retrieves upstream sequences for a list of genes.

1. In the left frame, under the title Sequence retrieval, click on the tool “retrieve sequence.” The sequence retrieval form appears on the right frame.
2. On this form, make sure that the option “Single organism” is selected.

3. Choose the organism (e.g., *Saccharomyces cerevisiae*).
4. Besides the title “Genes,” make sure that the option “selection” is checked (*see Note 1*).
5. Paste the list of genes of interest in the text box (*see Notes 2 and 3*).
6. Check that the “Feature type” is set to “CDS” (*see Note 4*).
7. As “Sequence type,” select “upstream” (*see Note 5*).
8. Check that the sequence limits (“From” and “To”) are set to default (*see Notes 6 and 7*).
9. To discard coding sequences located closer than the limit of the sequences being retrieved, select the option “Prevent overlap with previous ORFs” (*see Note 8*).
10. Check that the “Sequence label” is set to “Gene name” (*see Note 9*).
11. Check that the “Output” mode is set to “server” (*see Note 10*).
12. Click on “GO.”
13. After some time, the result is displayed.
14. The actual sequence is not displayed, but instead, a hyperlink gives access to the sequences.

3.2. Oligonucleotide Analysis

This pattern discovery program oligo-analysis ([1](#)) detects significantly over-represented oligonucleotides in a set of input sequences. We will use it to discover putative *cis*-acting signals in the promoters of the 27 genes activated by nitrogen depletion.

1. Below the link to the sequence, the “Next step” section shows a list of buttons allowing the user to send the retrieved sequences as input to the purge sequence utility and a variety of pattern matching and pattern discovery tools.
2. Click “oligo-analysis.” The sequences are passed from the result page of retrieve-seq to the oligo-analysis form (*see Notes 11 and 12*).
3. Make sure the “purge sequences (highly recommended)” option is checked (*see Note 13*).
4. Check that the “Oligonucleotide size” is set to “6” (*see Note 14*).
5. Make sure the “prevent overlapping matches” option is checked (*see Note 15*).
6. Check that the “Count on” mode is set to “Both strands” (*see Note 16*).
7. Check that in “Expected frequency calibration,” “Predefined background frequencies” is selected (*see Notes 17 and 18*), that the “Background model” is set to “upstream-noorf,” and that the selected “Organism” is the one for which the sequences were retrieved (*see Note 19*).
8. Check that the “Lower threshold” is set to “0” for “Significance” (*see Note 20*).
9. Check that the “Output” mode is set to “server” (*see Note 10*).
10. Leave all other options unchanged and click on “GO.”
11. After a few seconds, the result page appears.

3.2.1. Interpretation of the Result of Oligo-Analysis

1. The leading comment lines (starting with “;”) summarize the selected parameters. The result appears in a table where each row corresponds to one significant oligonucleotide, and each column to one statistical criterion (Fig. 1).
2. The most significant hexanucleotide in promoters of the 27 genes activated by nitrogen depletion is GATAAG.
 - a. This word is found in 35 occurrences in the 27 promoters.
 - b. According to the background model, the random expectation would be 8.75.

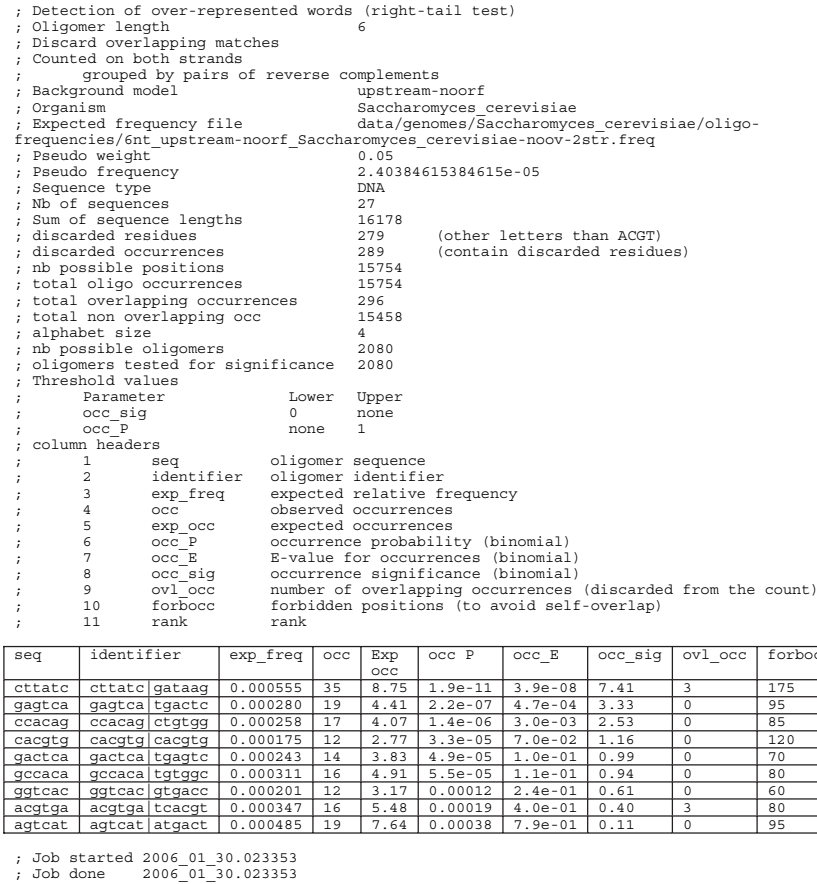


Fig. 1. Result of oligo-analysis on the upstream sequences of the 27 genes responding to nitrogen depletion. The first discovered pattern corresponds to the binding site of the GATA transcription factors and the second one to GCN4.

- c. The p -value ($\text{occ\_P} = 1.9e - 11$) indicates the probability for this word to be a false-positive, i.e., the probability to observe, by chance, 27 occurrences of a word when 8.75 are expected.
 - d. The e -value ($\text{occ\_E} = 3.9e - 8$) is a correction for multitesting, which indicates the number of false-positives that would be expected by chance, if we admit this level of p -value.
 - e. The significance $\text{occ\_sig} = -\log_{10}(e\text{-value})$ is a simple logarithmic conversion of the e -value. See [ref. 1](#) for more details about the probabilistic model.
3. Typically, a promising result consists in a collection of a dozen of words. With our first study case (nitrogen depletion), nine words were selected as significant, among the 4096 possible hexanucleotides. A closer inspection of these significant words in [Fig. 1](#) shows that some of them show strong mutual overlap. Actually, these mutually overlapping words generally represent fragments of a larger motif, or variants of a partly degenerated motif.
4. This interesting property is revealed by assembling (aligning) the significant words, which is automatically done on the website. Below the oligo-analysis table, the result page displays the assembly of the discovered oligonucleotides ([Fig. 2](#)). The program pattern-assembly identified three groups of overlapping words (assemblies), and two isolated words (among which GATAAG, the most significant one).
5. We can now compare the discovered motifs with our prior knowledge about transcriptional regulation in yeast.
 - a. Actually, the most significant motif, GATAAG, is the so-called GATA-box, which is bound by four alternative transcription factors (the GATA factors) mediating nitrogen regulation.
 - b. The second most significant motif, regrouping GAGTCA ($\text{sig} = 3.33$), and four overlapping hexanucleotides, corresponds to the consensus of the transcription factor Gcn4p, involved in the general control of amino acids.
 - c. The third motif (GCCACAG), made of two hexanucleotides, corresponds to the binding site of Met31p and Met32p, two homologous transcription factors regulating methionine biosynthesis.
 - d. The fourth motif, composed of three hexanucleotides assembled in TCACGTGA, is the consensus of the Met4p/Cbf1p/Met28p complex, the main regulator of methionine biosynthesis.
6. In summary, this simple detection of the over-represented words in the promoters of genes responding to nitrogen depletion revealed four motifs, all corresponding to nitrogen metabolism.

3.3. Pattern Matching

We will now use the pattern matching tool *dna-pattern* ([4,5](#)) to locate instances of the hexanucleotides discovered by oligo-analysis in the previous section.

```

; pattern-assembly
; Input score column      8
; Output score column    0
; two strand assembly
; max flanking bases     1
; max substitutions      1
; max cluster size       50
; max number of patterns 100
; number of input patterns 9
;

;assembly # 1 seed: gagtca 5 words length
; align rev_cpl score
tgactc.. ..gagtca 3.33
tgagtc.. ..gactca 0.99
.gagtca. .tgactc. 3.33
.gactca. .tgagtc. 0.99
..agtcac atgact.. 0.11
tgagtcac atgactca 3.33 best consensus

;assembly # 2 seed: ccacag 2 words length 8
;align rev_cpl score
gccaca. .tgtggc 0.94
.ccacag ctgtgg. 2.53
gccacag ctgtggc 2.53 best consensus

;assembly # 3 seed: cacgtg 3 words length 7
; align rev_cpl score
tcacgt.. ..acgtga 0.40
.cacgtg. .cacgtg. 1.16
..acgtga tcacgt.. 0.40
tcacgtga tcacgtga 1.16 best consensus

; Isolated patterns: 2
;align rev_cpl score
cttatc gataag 7.41 isol
ggtcac gtgacc 0.61 isol
;Job started 30/01/06 02:33:54 CET
;Job done 30/01/06 02:33:55 CET

```

Fig. 2. Assembly of the oligonucleotides discovered with oligo-analysis on the 27 genes responding to nitrogen depletion.

1. At the bottom of the oligo-analysis result page, click on the button labeled “pattern matching (dna-pattern).”
2. Leave all parameters unchanged and click “GO.”
3. After a few seconds, the result page appears. The leading comment lines (starting with “;”) summarize the parameters chosen for the pattern search.
4. The result is displayed in a table. Each instance of the significant oligonucleotides is indicated.
 - a. The position is calculated from the end of the sequence, and indicated in negative coordinates.
 - b. The matched instances are returned in uppercases, together with the neighboring residues (lowercases).
 - c. The significance calculated by oligo-analysis is reported in the score column of dna-pattern.

3.4. Feature Map

This program generates a physical map of genetic features for one or several sequences.

1. At the bottom of the dna-pattern result, click on the button “feature-map.” This will transfer this result (matched positions) to the input form of the program feature-map.
2. Leave all parameters unchanged and click “GO.”
3. After a few seconds, a map is displayed (*see Note 21*), showing the instances of the discovered patterns (represented as colored boxes) in the input sequences (**Fig. 3**).
4. The height of the boxes is proportional to the significance of the patterns, as estimated by oligo-analysis.

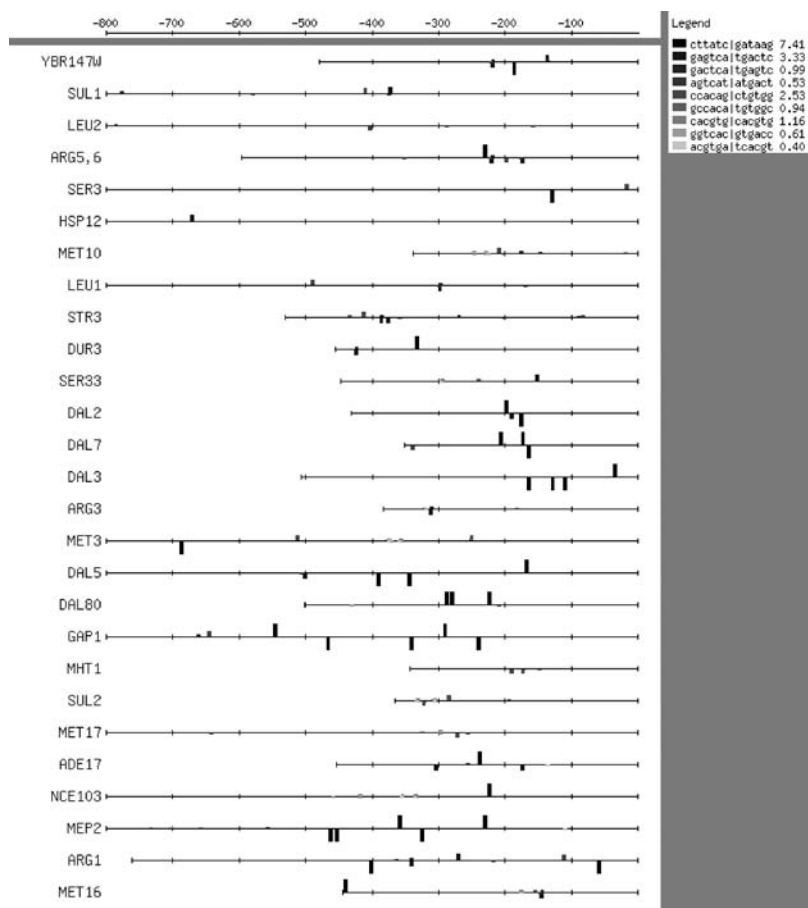


Fig. 3. Feature map of the patterns discovered with oligo-analysis in promoters of the 27 genes responding to nitrogen depletion.

3.4.1. Interpretation of the Feature-Map

The feature map gives an intuitive representation of the discovered motifs. Its interpretation deserves some comments.

1. The height of each bow is proportional to the significance of the corresponding pattern. On this map ([Fig. 3](#)), this emphasizes the predominant role of the GATA box.
2. Each box is not necessarily a regulatory element. For example, it is well known that an effective nitrogen regulation relies on multiple GATA boxes. On the map, the promoters with two or three GATA-boxes are thus probably regulated by GATA factors, but this is not obligatorily the case for those with a single GATA box.
3. As previously stated about the interpretation of the pattern assembly, several motifs (Gcn4p, Met4p, Met31p) are composed of a collection of over-represented hexanucleotides. On the feature-map, the corresponding binding sites appear as clumps of boxes, representing groups of overlapping fragments (hexanucleotides) of a larger motif.
4. Each of the discovered motifs is present in a subset of the promoters only. This illustrates the robustness of the method: it is able to detect relevant patterns even if the data set contains a mixture of sequences regulated by distinct factors.
5. The feature-map also illustrates the combinatorial aspect of regulation: the Gcn4p binding sites (GAGTCA) act in combination with other sites to regulate the various pathways involved in amino acid metabolism.

3.5. Dyad Analysis

The program oligo-analysis usually gives good results with yeast promoters, but it fails to detect a whole class of regulatory motifs: the spaced pairs.

Our second study case is a good illustration of this limitation. [Figure 4](#) shows the result of oligo-analysis with the 18 genes downregulated by glucose ([Table 2](#)). The program only returns one oligonucleotide (GGAAAA), detected with a low significance ($\text{sig} = 0.26$). A motif of this significance is expected by chance every two sequence sets ($e\text{-value} = 0.55$). This motif is thus likely to be a false-positive.

The reason why the program fails to detect any significant motif is that the regulator involved in this response binds DNA at two distant contact points, separated by a spacing of fixed width but variable content. The pattern discovery program dyad-analysis was specifically designed to detect this kind of patterns ([2](#)). It relies on the same statistics as oligo-analysis, but extends the analysis to dyads, i.e., pairs of short oligonucleotides (typically trinucleotides) separated by a spacing of fixed width, but variable content. We illustrate hereafter the use of dyad-analysis with the second study case.

```
; oligo-analysis -i tmp/retrieve-seq.2006_02_01.173533.res.purged -format fasta -sort -
lth occ_sig 0 -return occ,rank,proba -2str -noov -v -seqtype dna -l 6 -bg upstream-noorf
-org Saccharomyces_cerevisiae -pseudo 0.05
; Citation: van Helden et al. (1998). J Mol Biol 281(5), 827-42.
; Detection of over-represented words (right-tail test)
; Oligomer length 6
; Input file tmp/retrieve-seq.2006_02_01.173533.res.purged
; Input format fasta
; Discard overlapping matches
; Counted on both strands
; grouped by pairs of reverse complements
; Background model upstream-noorf
; Organism Saccharomyces_cerevisiae
; Method Frequency file
; Expected frequency file data/genomes/Saccharomyces_cerevisiae/oligo-
frequencies/6nt_upstream-noorf_Saccharomyces_cerevisiae-noov-2str.freq
; Pseudo weight 0.05
; Pseudo frequency 2.40384615384615e-05
; Sequence type DNA
; Nb of sequences 18
; Sum of sequence lengths 11576
; discarded residues 668 (other letters than ACGT)
; discarded occurrences 663 (contain discarded residues)
; nb possible positions 10823
; total oligo occurrences 10823
; total overlapping occurrences 245
; total non overlapping occ 10578
; alphabet size 4
; nb possible oligomers 2080
; oligomers tested for significance 2080
; Threshold values
; Parameter Lower Upper
; occ_sig 0 none
; occ_P none 1
;
; column headers
; 1 seq oligomer sequence
; 2 identifier oligomer identifier
; 3 exp_freq expected relative frequency
; 4 occ observed occurrences
; 5 exp_occ expected occurrences
; 6 occ_P occurrence probability (binomial)
; 7 occ_E E-value for occurrences (binomial)
; 8 occ_sig occurrence significance (binomial)
; 9 ovl_occ number of overlapping occurrences (discarded from the
count)
; 10 forbocc forbidden positions (to avoid self-overlap)
; 11 rank rank
; 12 test over- or under-representation test
```

seq	identifier	exp_freq	occ	exp_occ	occ_P	occ_E	occ_sig	ovl_occ	forbocc	rank
ggaaaa	ggaaaa ttttcc	0.001869	38	20.22	0.00026	5.5e-01	0.26	0	190	1

```
; Job started 2006_02_01.173624
; Job done 2006_02_01.173627
```

Fig. 4. Result of oligo-analysis on 18 yeast genes downregulated by glucose in the Gasch (2000) dataset. The only word returned by the program has a quite low significance, so that this result can be considered as a negative answer.

1. Select the 18 genes from [Table 2](#) and retrieve their upstream sequences as explained in [Subheading 3.1](#). At the bottom of the result page, click on the button labeled “dyad-analysis.”
2. Check that the “Spacing” is set to “from 0 to 20” (see [Note 22](#)).
3. Make sure the “prevent overlapping matches” option is checked (see [Note 23](#)).
4. For the “Expected frequency calibration” make sure that:
 - a. The option “Background model” is selected (see [Note 24](#)).
 - b. “Sequence type” is set to “upstream-noorf” (see [Note 25](#)).
 - c. The “Organism” is correctly specified (e.g., *S. cerevisiae* for the study case 2).

5. Check that the “Lower threshold” is set to “1” for “Occurrences” and to “0” for “Significance” (see **Note 20**).
6. Leave all other parameters unchanged and click on “GO.”
7. The computation takes more time than for oligo-analysis, because when we sample spacing sizes from 0 to 20, there are 21 times more possible dyads than hexanucleotides.
8. After having obtained the result, the locations of the significant dyads can be detected and a feature-map can be drawn in the same way as for oligo-analysis (**Subheadings 3.3. and 3.4.**).

```

; Sequence type                               DNA
; Nb of sequences                             18
; Sum of sequence lengths                     11576
; default return values                       proba,occ
; return values                               proba,occ,exp_freq,exp_occ,rank
; Monad parameters
;   monad size                               3
;   monad positions                           22414
;   valid                                     21748
;   discarded                                 666 (contain other letters than ACGT)
;   distinct monads                           64
; Dyad parameters
;   dyad type                                any dyad
;   minimal spacing                           0
;   maximal spacing                           20
;   dyad positions                            11486
;   valid                                     10820
;   discarded                                 666 (contain other letters than ACGT)
;   distinct dyads                           43680
;   dyads tested for significance             41881
; Threshold values
;   Parameter                                Lower Upper
;   occ                                       1      none
;   occ_sig                                  0      none
; Estimation of expected dyad frequencies
;   Background model
;   organism                                  Saccharomyces cerevisiae
;   sequence type                            upstream-noorf
; column headers
;   1 sequence
;   2 identifier
;   3 expected_freq
;   4 occ                                     observed occurrences
;   5 exp_occ                               expected occurrences
;   6 occ_P                                 occurrence probability (binomial)
;   7 occ_E                                 E-value for occurrences (binomial)
;   8 occ_sig                               occurrence significance (binomial)
;   9 ovl_occ                               number of overlapping occurrences
;   10 all_occ                              number of non-overlapping + overlapping occurrences
;   11 rank                                 rank
;   12 ov_coef                              overlap coefficient
;   13 remark                               remark

```

sequence	Exp freq	occ	exp_occ	occ_P	occ_E	occ_sig	ovl_occ	all_occ
cggn{10}tcc	0.00012960	14	1.34	3.5e-10	1.5e-05	4.83	3	17
cggn{11}ccg	0.00006980	8	0.72	1.3e-06	5.6e-02	1.25	1	9
ggan{1}aaa	0.00154442	39	16.27	2.2e-06	9.4e-02	1.03	0	39
ggan{5}gga	0.00029840	14	3.11	7.7e-06	3.2e-01	0.49	0	14
caan{1}gag	0.00043829	17	4.62	1e-05	4.3e-01	0.37	0	17
cacn{1}aac	0.00036189	15	3.81	1.6e-05	6.5e-01	0.19	0	15

```

;Job started 01/02/06 02:30:59 CET
;Job done    01/02/06 02:31:24 CET

```

Fig. 5. Result of dyad-analysis on 18 yeast genes downregulated by glucose in the Gasch (2000) dataset. Some rows and columns have been suppressed to improve readability.

```

; pattern-assembly -v 1 -subst 0 -2str -maxfl 1 -subst 0 -i public_html/tmp/dyad-
analysis.2006_02_01.023057.res
; Input file      public_html/tmp/dyad-analysis.2006_02_01.023057.res
; Input score column      8
; Output score column     0
; two strand assembly
; max flanking bases      1
; max substitutions       0
; max assembly size       50
; max number of patterns  100
; number of input patterns 6
;

;assembly # 1 seed: cggnnnnnnnnnntcc 3 words length
;      align      rev_cpl      score
cggnnnnnnnnnntcc.      .ggannnnnnnnnnccg      4.83
cggnnnnnnnnnnnccg      cggnnnnnnnnnnnccg      1.25
.ggannnnnnnnnnnnccg      cggnnnnnnnnnntcc.      4.83
cggnnnnnnnnnntccg      cggnnnnnnnnnntccg      4.83      best consensus

; Isolated patterns: 4
;align rev_cpl score
gganaaa ttntcc 1.03      isol
ggannnnnnga      tccnnnnntcc      0.49      isol
caangag ctcnttg 0.37      isol
cacnaac gttngtg 0.19      isol
;Job started 01/02/06 02:31:25 CET
;Job done    01/02/06 02:31:25 CET

```

Fig. 6. Assembly of the dyads discovered with dyad-analysis on 18 yeast genes responding negatively to galactose.

3.5.1. Interpretation of the Result of Dyad-Analysis

1. As for oligo-analysis, the leading comment lines (starting with “;”) summarize the parameters used for the analysis.
2. The result appears in a table where each row corresponds to one spaced dyad and each column to one statistical criterion (**Fig. 5**). Among the 41,881 dyads encountered in the input sequence set, no more than 6 passed the threshold of significance. In addition, the two most significant dyads can be assembled (**Fig. 6**) to form the larger motif CGGAn{9}TCCG, which corresponds to the Gal4p-binding motif.
3. The feature-map of the significant dyads is displayed in **Fig. 7**. It is interesting to note that the Gal4p motif is detected in no more than half of the promoters (9 among 18). This shows the robustness of the method, which does not require for the motif to be present in all the input sequences.
4. A closer analysis of the feature-map reveals that 3 of the 18 promoters contain multiple instances of the discovered motif. As suggested by their names (*GAL2*, *GAL1*, *GAL10*), these genes are all involved in galactose metabolism.
5. In summary, the analysis of our second study case showed that a fraction of the genes repressed by glucose contain the Gal4p-binding site, and are indeed involved in galactose utilization. It is well known that glucose is a preferred carbon source, and that in presence of glucose, the catabolism of other carbon sources is repressed.

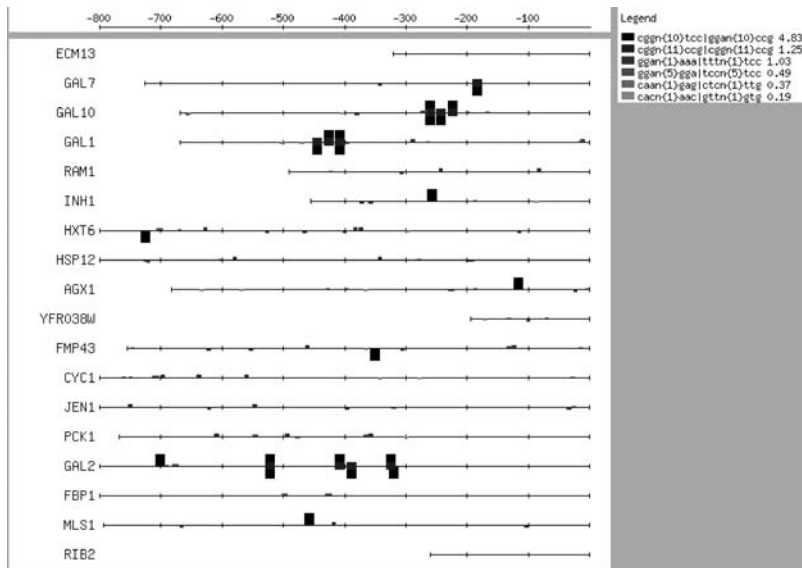


Fig. 7. Feature map of the patterns discovered with dyad-analysis in promoters of the 18 genes responding to glucose.

Acknowledgments

This project was partly supported by the BioSapiens Network of Excellence funded under the sixth Framework programme of the European Communities (LSHG-CT-2003-503265). Genome installation is done on a 40-node PC cluster contributed by various institutions, including the Belgian Fonds pour la Recherche Fondamentale Collective (FRFC grant 2005). We are grateful to Raphaël Leplae for his enthusiasm in obtaining, installing, and maintaining this cluster. JvH acknowledges Stéphane Vissers for an inspiring discussion on the importance of good quality protocols for teaching and practicing good science. We are thankful to Stephan Kurtz for making available his very efficient program vmatch, which is used to purge redundant fragments.

4. Notes

1. The alternative would be to analyze all the genes of a genome together. This can be done to detect general motifs involved in transcriptional regulation, but it requires more complex probabilistic models (Markov chains).
2. Gene selection. Each query gene must come as the first word of a new line. Additional text on the same line is ignored. The list of genes can also be uploaded from a text file using the option "Upload gene list from file."

3. Gene names. Gene names, synonyms, or gene identifiers can be used. They have to be separated by carriage returns. Query genes are case-insensitive.
4. Feature type. Ideally, the reference position for elements involved in transcriptional regulation is the transcription start site (TSS). Unfortunately, the location of TSS is generally not annotated because their computer-based detection is extremely difficult and inaccurate. For most genomes, annotations only indicate the boundaries of coding sequences (CDS). For this reason, the default feature type is set to CDS. For some genomes (e.g., *Drosophila*, Human, . . .), boundaries of transcription units are also annotated. In such cases, the option mRNA can be selected as feature type, and the TSS is used as reference position for the upstream (5') side of genes.
5. Sequence type. The terms "upstream" and "downstream" refer to 5' boundary and the 3' boundary of the selected feature type (CDS, mRNA, . . .), respectively.
6. Upstream region size. The efficiency of pattern discovery programs depends on the selected size. Default values have been determined in an organism-specific way (400 bp for bacteria, 800 bp for yeast, . . .).
7. Negative/positive coordinates. Negative (positive) coordinates return sequences located upstream (downstream) the reference point. Thus, sequences located on the 5' side of a gene are obtained with negative values and sequence type "upstream," whereas sequences located on the 3' side are obtained with positive coordinates and sequence type "downstream."
8. Clipping. For prokaryotic genomes, where some genes are part of an operon, it is very important to clip the sequence of upstream regions at the end of the preceding open reading frame (ORF). Indeed, the distribution of words in coding sequences is not the same as in noncoding sequences and therefore the calibration done on noncoding sequences is not appropriate for those.
9. Sequence label. "full identifier" is a concatenation of ORF identifier, gene name, sequence type, from, to, and strand. This option gives a full description of the conditions of sequence retrieval.
10. Output mode. The "server" mode will keep the data on the server for further usage without displaying it. By default, sequences are not displayed to minimize data transfer through the web browser, but if the connection is fast enough, the link can be clicked to see the sequence. The "display" mode will display the sequences in the results page. The "email" mode will prepare the sequences on the server, and send an email when the task is finished. This can be useful for large queries, but it is usually not necessary.
11. Actually, these sequences are not seen (for the same reason: to avoid transferring large data sets through the web), but they can be checked by clicking on the link "sequences."
12. Alternatively, instead of transferring sequences obtained with retrieve-seq, the user can paste his sequences. For this, the oligo-analysis needs to be clicked in the left frame (the menu) to obtain a fresh form.

13. Purge sequence. For pattern discovery, purging sequences is important to avoid statistical biases in case of repeated sequences. On the contrary, the pattern matching and feature-map drawing are done with nonpurged sequences, to highlight all the candidate *cis*-acting elements.
14. Oligonucleotide size. The default value is six because this value generally gives good results, at least with yeast promoters. However, depending on the data set, more significant patterns are sometimes detected with larger or smaller words. We suggest, thus, to test various oligonucleotide sizes (from five to eight) and to compare the results. An increase in word size generally reduces the noise but also the signal.
15. Overlapping matches. Counting all occurrences of overlapping oligonucleotides introduces a bias to most statistics (binomial, log-likelihood).
16. Count on mode. Counting on both strands allows detection of elements acting in an orientation-insensitive way. This model is appropriate for yeast *cis*-acting elements, but if the users want to use the same tool with RNA sequences (e.g., 3' UTR), single-strand analysis is of course recommended.
17. Expected calibration frequency. Various probabilistic models can be used to estimate prior oligonucleotide probabilities, i.e., the probability to observe each oligonucleotide at a given position of the sequence. This is one of the most important parameters of the analysis, and its choice can have drastic effects on the results. The default model (Predefined background frequencies) was calibrated by counting, for each supported organism, the frequency of each oligonucleotide in the whole set of upstream sequences. This model gives the best results in our evaluations (Sand and van Helden, in preparation). Other models are supported as well, which can be useful for specific types of analyzes, but for the analysis of co-expressed clusters, we definitely recommend the predefined background frequencies.
18. Predefined background frequencies. Compare oligonucleotide frequencies observed in the query sequence to those of a reference sequence (the background model).
19. In principle, these parameters have been selected automatically when the sequences are transferred from retrieve-seq to oligo-analysis.
20. Threshold on significance. The significance is a simple logarithmic conversion of the *e*-value: $\text{sig} = -\log_{10}(e\text{-value})$, which gives an intuitive perception of the reliability of a prediction: the higher is the significance, the more reliable is the motif. With a significance level of 0, the random expectation is around one false-positive per analysis. With a significance of 1, a false-positive is expected every 10 sequence sets. With a significance of *s*, a false-positive is expected every 10^s sequence sets.
21. The feature-map appearing on the screen may differ from the one published here for two reasons: (1) we selected a monochrome because of publishing constraints,

- and (2) we modified the parameter “thickness” and “spacing” to obtain a more compact map.
22. Dyad spacing. The spacing is the number of bases between the end of the first element and the beginning of the second one. This spacing is factor-specific. Because we assume we have no prior knowledge of the factor regulating our genes, we systematically test all the dyads, and select the most significant ones.
 23. Overlapping matches. Counting all occurrences of overlapping dyads introduces a statistical bias, leading to an over-estimation of the significance for self-overlapping dyads. This bias is circumvented by discarding self-overlapping occurrences from the count.
 24. Expected calibration frequency for dyads. For dyads, two models are proposed.
 - a. Predefined background frequencies: the prior probability of each dyad is estimated by computing its frequency in a reference sequence set, i.e., the whole set of upstream sequences for the considered organism.
 - b. Monad frequencies. The prior probability of each dyad is estimated by calculating the product of the monad (trinucleotides forming the dyad) frequencies in the input set.
 25. Background model. upstream: all upstream regions, allowing overlap with upstream ORFs; upstream-noorf: all upstream regions, preventing overlap with upstream ORFs (sequences are clipped to discard upstream ORF sequences); intergenic: all the intergenic regions, including upstream, and downstream sequences.

References

- 1 van Helden, J., Andre, B., and Collado-Vides, J. (1998) Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.* **281**, 827–842.
- 2 van Helden, J., Rios, A. F., and Collado-Vides, J. (2000) Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res* **28**, 1808–1818.
- 3 Gasch, A. P., Spellman, P. T., Kao, C. M., et al. (2000) Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell.* **11**, 4241–4257.
- 4 van Helden, J. (2003) Regulatory sequence analysis tools. *Nucleic Acids Res.* **31**, 3593–3596.
- 5 van Helden, J., Andre, B., and Collado-Vides, J. (2000) A web site for the computational analysis of yeast regulatory sequences. *Yeast* **16**, 177–187.

Fastcompare

A Nonalignment Approach for Genome-Scale Discovery of DNA and mRNA Regulatory Elements Using Network-Level Conservation

Olivier Elemento and Saeed Tavazoie

Summary

Here, we describe the usage of Fastcompare, a simple and efficient comparative approach for finding short noncoding DNA (e.g., transcription factor binding sites) and mRNA (e.g., microRNA target sites) sequences that are globally conserved between two genomes. Fastcompare is based on the network-level conservation principle, according to which the connectivity of transcriptional regulatory networks should be largely conserved between two closely related genomes. We describe here the procedure for applying Fastcompare to large genomes (with an emphasis on metazoan genomes), including scoring of exhaustive motif lists, determination of conservation threshold using sequence randomizations, and discovery of interactions between regulatory elements.

Key Words: Transcription factor binding sites; microRNA target sites; computational method; network-level conservation; comparative genomics; metazoan genomes.

1. Introduction

1.1. Motivation and Assumptions

The growing number of fully sequenced prokaryotic and eukaryotic genomes provides unique opportunities for annotating functional elements that reside within them. One of the challenges is to determine regulatory elements encoded within these genomes and to predict their biological function. We have described a simple and efficient comparative approach for finding short

noncoding DNA (e.g., transcription factor binding sites) and mRNA (e.g., microRNA target sites) sequences that are globally conserved between two genomes (*1-3*). Our method, which we have called Fastcompare, is based on the assumption that the set of genes regulated by the same transcription factor in two related genome should be approximately conserved. This implies that the wiring of transcriptional regulatory networks should also be largely conserved between two closely related genomes; we have termed this type of conservation “network-level conservation” (*2*) (see [Fig. 1](#)). Unlike other approaches, Fastcompare does not use global alignments. A candidate regulatory element is considered as conserved between two orthologous promoter regions if it is found in both promoters, without requiring its position to be conserved. Note, however, that conservation is not assessed from a single pair of orthologous promoter regions, but over all orthologous promoter regions between the two considered genomes (i.e., for all conserved genes).

It is important to emphasize that Fastcompare is not a tool for discovering conserved sites within multiple orthologous promoter regions for the same gene, as described elsewhere (*4,5*), but provides a genome-wide assessment of the conservation for a very large number of patterns. Indeed, Fastcompare exploits

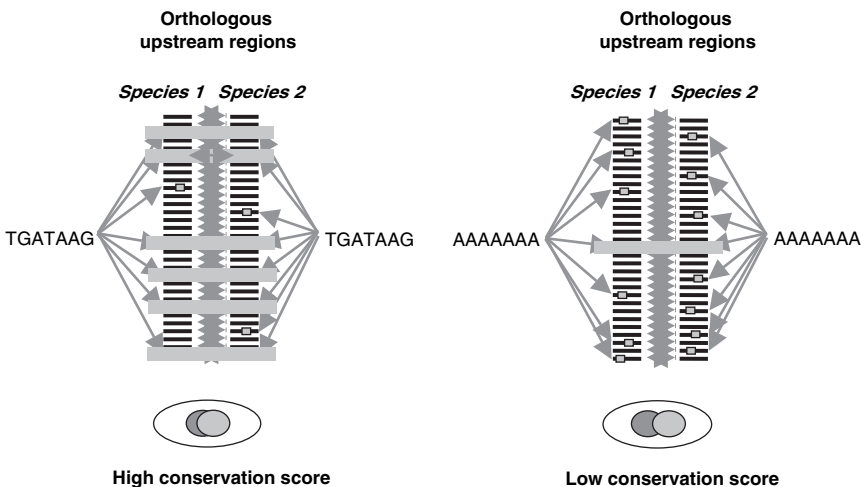


Fig. 1. Overview of motif discovery using network-level conservation. On the left, TGATAAG is given a high conservation score, because there are many pairs of orthologs that contain the k -mer in their upstream region. Consequently, TGATAAG is likely to be functional. On the other hand, AAAAAA is given a low conservation score as the sets of upstream regions that contain it do not significantly overlap.

the fact that transcription factors or mRNA-binding regulators often regulate many genes to discover their binding site (although without any knowledge about the regulating molecule). In fact, Fastcompare can be used even with unfinished genomic sequences, provided a sufficiently large number of orthologous genes are available (2).

We have used Fastcompare to build catalogues of globally conserved regulatory elements in yeasts, worms, flies, and mammals. We have used it for predicting both DNA and mRNA regulatory elements (1,3). In both cases, many of the highly conserved elements we have found had been previously experimentally verified. In ref. 3, we also found many highly conserved mRNA elements that are complementary to the 5' extremity of microRNAs. Similar to ref. 6, we have used the remaining predicted mRNA regulatory elements to predict novel microRNAs (3). Importantly, we have also predicted the involvements of many of the known and novel elements in many diverse cellular processes (3).

In this chapter, we describe in detail the application of Fastcompare to the identification of globally conserved elements in yeast and metazoan genomes. Alternative attempts at discovering conserved regulatory elements have essentially used multiple alignments of several genomes (7,8). These approaches have been very successful, but usually employ at least four genomes. Moreover they are not applicable to genomes that span a high degree of sequence divergence.

1.2. Motif Definition

For the sake of simplicity, we focus in what follows on 5' upstream regions. However, other classes of noncoding regions can be used almost interchangeably. Given two sets of orthologous upstream regions and a list of putative DNA motifs, Fastcompare calculates the network-level conservation score for all motifs, and outputs these motifs sorted by a conservation score. Motifs can be defined in various ways; in our initial attempt, we have used weight matrices returned by the AlignACE Gibbs sampler (2). In our latest work, we have focused on k -mers (sequences of k nucleotides); the use of k -mers allows for a more comprehensive search over sequence space, in addition to making the approach extremely fast. It, however, retains the ability to recover most of the known transcription factor binding sites in yeast (1), including the core of very degenerate motifs (e.g., RAP1). However, highly degenerate binding sites are in general not represented well by k -mers. Although the use of degenerate patterns may provide an attractive alternative, they are not supported in the present version of Fastcompare because exhaustive search over such

patterns is computationally expensive. In what follows, we only use exact k -mers, with or without gaps, as it is known that some binding sites are actually gapped (e.g., GAL4 in yeast). Our published analyses have used ungapped k -mers with k ranging from seven to nine (1,3).

1.3. Calculating Conservation Scores

Figure 1 depicts the principle behind using network-level conservation for motif discovery. The network level-conservation score for a given k -mer is simply defined as the hypergeometric p -value,

$$P(X \geq i) = \sum_{x=i}^{\min(s_1, s_2)} \frac{\binom{s_1}{x} \binom{N-s_1}{s_2-x}}{\binom{N}{s_2}},$$

where s_1 is the number of genes that have the k -mer in their upstream region in the first species, s_2 is the number of genes that have the k -mer in their upstream region in the second species, i is the overlap between the two sets (i.e., the number of orthologous genes that have the k -mer in both species). Finally, N is the total number of orthologous genes. It is important to note that the p -value calculated by the hypergeometric function previously described is not used in the traditional null hypothesis rejection scheme. Indeed, in cases of recent common ancestry, a large number of k -mers will have nominally significant p -values without being functional. However, our approach assumes that functional regulatory elements will be more conserved than the genomic background. Hypergeometric p -values are therefore only used to rank candidate k -mers according to their level of conservation. For convenience, we define the conservation score as the negative logarithm (base e) of the p -value returned by the cumulative hypergeometric function previously described.

1.4. Selecting the Most Conserved k -mers

As previously mentioned, we generally run Fastcompare with varying k -mer sizes ranging from $k = 7$ to 9 (1,3). Our strategy has been to use 7-mers as core conserved motifs; when possible, we extend the highest scoring 7-mers using 8- and 9-mers (see **Subheading 1.5.**). Thus, we first retain only the most conserved 7-mers. Several strategies can be used to do so. For example, when dealing with DNA regulatory elements in yeast and several other metazoan genomes, we have observed that many of the approx 400–500 most conserved 7-mers were validated by other types of independent data (1), such as gene expression,

functional categories (Gene Ontology, MIPS), in vivo measurement of DNA occupancy using ChIP (9), TRANSFAC curated motifs (10), or complementarily to known microRNAs (11).

A complementary strategy is to use sequence randomizations. Randomized upstream regions have been used to show that the largest conservation scores obtained on the actual upstream regions are unlikely to arise by chance (1,3). They can also be used to assess whether a k -mer is significantly more conserved than what one would expect from two genomes with the same degree of divergence, but neutrally evolving (i.e., without negative selection pressure on certain elements). Such a procedure is especially useful when conservation scores are very low, i.e., when genomes are very divergent, as few motifs are actually significantly conserved. However, randomizations are computationally costly.

We perform randomizations as follows. We generate many sets of randomized sequences (e.g., 100), run Fastcompare on each of these sets, then calculate a z-score for each k -mer. The z-score of a given k -mer is calculated as $z = (S - \overline{S_{rand}}) / \sigma_{rand}$ where S is the actual conservation score (obtained on real data), $\overline{S_{rand}}$ and σ_{rand} are the average score and standard-deviation calculated out of the 100 randomized runs for the same k -mer. Only k -mers with z-scores above a given threshold, defined in terms of number of standard deviations above the mean, are retained.

It is crucial that randomized orthologous upstream regions retain the same level of divergence as the original upstream regions. Therefore, we must estimate the divergence between each pair of orthologous upstream regions (as divergence often varies between distinct upstream regions). For each pair of orthologous upstream regions, we define a matrix of substitution rates, which we estimate as follows. First, we align the two orthologous regions using a global alignment algorithm (e.g., Needleman-Wunch [12]). We then use the alignment to calculate the transition rates for the considered pair of orthologous upstream regions. Then, we randomly pick one of the two orthologous regions, and mutate it using the transition rates previously defined, to generate a randomized orthologous sequence. We repeat the same procedure for all pairs of orthologous upstream regions.

1.5. Extending k -mers

Once a set of high-scoring 7-mers has been defined, either based on raw conservation scores or based on z-scores, we attempt to extend these 7-mers into longer k -mers, using conservation scores (or z-scores) calculated for all 8- and 9-mers. The procedure we use is as follows. First, we replace each of

the retained 7-mers by the 8-mer with highest conservation score (or z-score) for which the considered 7-mer is a substring. We also include within the final list the high-scoring 8-mers, which do not have any substrings within the initial set of 7-mers. Then, we repeat the same process for the 8-mers we have just added, replacing an 8-mer by a higher scoring 9-mer superstring (if there is one). Finally, we add the 9-mers that do not have any substrings among the 8-mers. This strategy thus produces the best length representation for candidate regulatory elements, starting from 7-mers.

1.6. Conserved Sets, Conserved k -mer Positions, and Orientations

We define the conserved set of a given k -mer as the set of genes that have the k -mer in their upstream region in both considered species. We have used conserved sets for predicting the function of high-scoring k -mers discovered by Fastcompare (1,3), using functional categories and gene expression data. We have also used conserved sets as the predicted target genes for microRNAs, when the considered k -mer was complementary to the 5' extremity of microRNAs.

In addition, we have used the positions and orientations of k -mers within their respective conserved sets to show that many of these k -mers have position and orientation biases (1,3). This provides both additional evidence for functionality of these predicted regulatory elements, and additional insights into the underlying regulatory mechanisms, possibly leading to more focused validation experiments.

1.7. Co-Conservation Among k -mers

As shown in ref. 1, the network-level conservation principle can be trivially extended to discover cases of co-conservation of two k -mers (possibly pointing at interactions between the factors that bind these sites). Indeed, significant simultaneous conservation (termed co-conservation) of two nonoverlapping k -mers within the same upstream regions is likely to predict such interactions. To discover such interactions, we proceed as previously described, except that instead of seeking two sets of upstream regions that contain a single k -mer, we seek the two sets of upstream regions that contain the two k -mers simultaneously. We evaluate conservation scores as previously described, and sort pairs according to their score. Once again, we have shown that this simple approach is capable of discovering known motif co-occurrences in yeast, such as the co-occurrence between PAC and RRPE sites (13).

2. Materials

2.1. Computer Program

1. Fastcompare is distributed as a set of console-based utilities. Linux and Windows binaries are freely available at <http://tavazoielab.princeton.edu/fastcompare/dist/>. Note however that the explanations below are largely geared toward Linux/Unix environments.
2. Although Fastcompare is an alignment-free method, the sequence randomization procedure previously described requires the application of global alignment. We have used ClustalW (14), available at <http://www.ebi.ac.uk/clustalw/>. ClustalW must be installed locally, before running the randomization scripts.
3. We provide on our website additional tools not described here, but that can be used for additional analyses, e.g., a script for determining which k -mers are complementary to the 5' extremity of microRNAs sequences, a script for determining orthologous genes using the reciprocal best BLAST hits approach, and so on. The procedure for using these tools is described on our website.

2.2. Sequences

1. Fastcompare uses as input two sets of orthologous sequences (upstream regions or 3' UTRs). Both upstream regions and 3' UTRs can be downloaded from Ensembl (15) for a large number of eukaryotic genomes. In other genomes, upstream regions have to be extracted from the whole genome sequence, using the gene annotation information (i.e., location of gene boundaries). We provide tools and explanations on our website for performing this task.
2. The use of Fastcompare sometimes requires the annotation of unpublished genomes to be done "in-house" by using the available annotation and protein sequences of a related genome. We have used a combination of BLAST (16) and Genewise (17) to perform this task. The corresponding script is available on our website. However, more sophisticated alternative strategies for gene annotation exist and should be considered when possible.
3. Masking repeats has not significantly affected the results of our analyses (1), however it should be considered when possible, to avoid discovering spurious motifs (e.g., motifs contained within *Alu* repeats). Masking exons, such that upstream regions do not overlap with other coding regions within the genome may also be considered.
4. For DNA analyses, we have used 1 kb upstream regions for yeasts and 2 kb upstream for metazoan genomes (1). The upstream region is defined from the transcriptional start site, when available, or from the start codon otherwise (18).
5. For mRNA analyses, real-length 3' UTRs should be used when available. When several alternate 3' UTRs exist for the same gene, we have retained only the longest one (3). When a gene has no annotated 3' UTR, we have chosen an arbitrary length corresponding to the 80th percentile of the lengths of the other annotated 3' UTRs (3). It is sometimes the case that one of the genomes has annotated 3'

UTRs, whereas the other has none. In that case, we use the 3' UTR lengths of the former for the latter. When a genome has no annotated 3' UTRs, a reasonable length estimate may be chosen for all 3' UTRs. It may be desirable to compare results obtained with several distinct lengths (e.g., 300, 500, and 1000 nt).

2.3. Orthology

1. Ensembl provides orthology relationships for the genomes it contains. For other genomes, such information has to be derived. We provide a script on our website for determining orthologs using the reciprocal best BLAST hits approach, from two sets of protein sequences (the script uses the NCBI standalone BLAST program [16], available at <ftp://ftp.ncbi.nih.gov/blast/executables/LATEST/>). For many genomes, orthology relationships can be obtained from the Inparanoid database (19). This orthology information is used in conjunction with upstream regions for all genes to generate two files of orthologous upstream regions, in the same order.

2.4. Candidate Regulatory Elements (*k*-mer Lists)

1. In addition to the sequences described above, Fastcompare takes as input a file containing the *k*-mers for which a conservation score will be calculated. Exhaustive lists of *k*-mers are available on our website, for $k = 5 - 10$, both for DNA (e.g., `7mers_dna.txt`) and mRNA analyses (e.g., `7mers_rna.txt`). In a DNA analysis, a *k*-mer and its reverse complement are considered as the same motif; therefore, reverse complements have been removed from lists of *k*-mers used in DNA analyses. On the other hand, in a mRNA analysis, all *k*-mers are considered.
2. The use of external lists of *k*-mers is meant to provide the user with flexibility about the patterns to be considered. For example, the user may prefer to remove sequences of low complexity, such as polyA- or polyT-tracts, if it is known that such sequences are over-abundant and unlikely to be regulatory elements in the considered genomes.

3. Methods

The steps in the Fastcompare procedure are depicted in the diagrams of Fig. 2., Subheadings 3.1., 3.2., and 3.3. describe the most commonly used options in Fastcompare. Note 1 describes the remaining options. In what follows, we seek to discover conserved motifs in the upstream regions of *Caenorhabditis elegans* and *C. briggsae* orthologous genes (20).

3.1. General Instructions for Running Fastcompare

1. Two files, `ce_u_2000.fa` and `cb_u_2000.fa`, contain 10,894 pairs of orthologous upstream regions for *C. elegans* and *C. briggsae*, respectively (see Note 2 for instructions regarding the format of these files). Another file, named

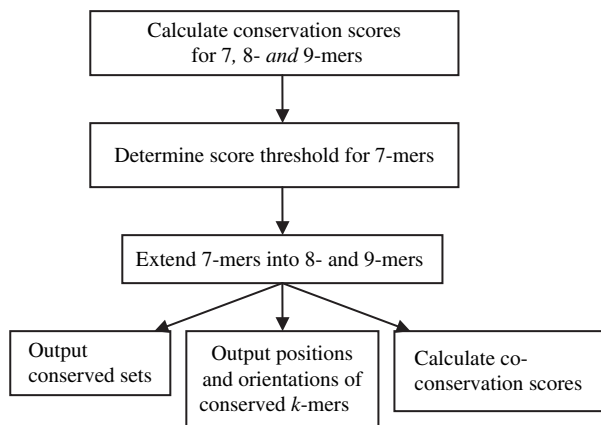


Fig. 2. Flowchart of a comprehensive Fastcompare analysis.

7mers_dna.txt, contains all 8192 7-mers. Given these files, the command-line for running Fastcompare for DNA analysis is the following:

```
fastcompare -fasta1 ce_u_2000.fa -fasta2 cb_u_2000.fa  
-kmers 7mers_dna.txt
```

After 1–2 min (on a modern desktop computer, *see* **Note 3**), Fastcompare outputs decreasing conservation scores for all 8192 7-mers, to the user’s screen (the output can be redirected to a file). This output will resemble the following:

CTGCGTC	1980	2184	813	295.13
AGACGCA	2196	2443	917	270.05
GAGACGC	1954	2255	785	246.73
CTTATCA	3599	3770	1752	234.36
AATCGAT	4036	3578	1831	228.18
TGACTCA	1970	2521	821	213.28
TCTTATC	3368	3800	1626	192.10
CACGTGG	1771	1249	457	179.49
TCCGCCC	1338	2155	533	169.49
CACGTGA	1515	814	311	167.59
CCGCCCA	1628	2107	595	162.43
CGAGACC	2023	1335	496	152.03
TGTTTGC	3477	3086	1358	143.99
AAGGTCA	1695	1832	537	141.26
ACGTCAT	2152	2181	713	134.52

ACGTGGC	1592	1410	424	133.30
ATCGATA	3041	2415	996	133.28
...				

Note 4 provides additional explanations on the Fastcompare output. **Note 5** describes possible reasons why the above listed command may fail to execute correctly.

3.2. Analysis of Gapped k -mers

1. Fastcompare also handles gapped k -mers, i.e., patterns such as TGGCNNNNNGCCA, where N can be any nucleotide. This is accomplished using the `-gap X` option, where X is an integer specifying the width of the gap. Note that, in a gapped analysis, the k -mer file is the same as when using Fastcompare without the `-gap` option (Fastcompare inserts gaps automatically). When using k -mers with even length, gaps will be inserted in the middle of the input k -mers. The corresponding command-line is:

```
fastcompare -fastal ce_u_2000.fa -fasta2 cb_u_2000.fa
-kmers 8mers_dna.txt -gap 4
```

which yields the following output:

TGGCNNNNNGCCA	433	321	121	204.45
AGAGNNNNNGAGA	2270	3134	985	144.99
CTTCNNNNNCTTC	2287	3281	974	105.58
CTCTNNNNNTCTC	2110	2857	820	104.84
CGTANNNNNACAC	643	465	116	98.26
...				

When using uneven length k -mers, the user has to specify the number of nucleotides after which the gap appears, using the `-l` option:

```
fastcompare -fastal ce_u_2000.fa -fasta2 cb_u_2000.fa
-kmers 8mers_dna.txt -gap 4 -l 5
```

which yields the following output:

AGAAGNNNNNAGA	2550	3597	1143	104.89
TGGCANNNNNCCA	731	503	125	91.49
TCGAGNNNNNACC	870	501	135	88.76
CTTCTNNNNNTTC	2338	3327	966	81.63
TTGGCNNNNNGCC	523	430	91	79.27
...				

3.3. Discovering mRNA Motifs Using Single-Stranded Analyses

1. The analysis of mRNA motifs is conducted using the `-singlestrand 1` option and using full lists of *k*-mers (see **Note 6** for instructions and common pitfalls of mRNA analyses). Note here that the sequence files (`ce_3utr.fa` and `cb_3utr.fa`) contain the *C. elegans* and *C. briggsae* orthologous 3' UTR sequences. The command-line for running the single strand Fastcompare analysis is:

```
fastcompare -fasta1 ce_3utr.fa -fasta2 cb_3utr.fa -kmers
7mers_rna.txt -singlestrand 1
```

which yields the following output:

CTGTGAT	387	415	187	405.70
ATTTATT	2580	2039	955	363.29
TGATCTC	466	490	197	355.54
TGTGATA	491	527	198	328.93
TATTTAT	1861	1444	590	293.06
ACGGGTT	285	240	115	288.17
TCTAGTC	236	273	112	284.54
TTGTGAT	679	680	228	270.72
TGTACAT	553	495	179	261.36
GATCTCT	388	452	148	254.45
...				

3.4. Sequence Randomizations (see **Note 7** for Comments)

1. As described in the introduction, the first step in the randomization procedure consists of estimating the substitution rates between each pair of orthologous upstream regions. The sequence files (`ce_u_2000.fa` and `cb_u_2000.fa`) must be the same as those used by Fastcompare in **Subheading 3.1**. The following command-line stores gene-specific substitution rates in the `mat.txt` file:

```
perl do_fastcompare_alignment -fasta1 ce_u_2000.fa
-fasta2 cb_u_2000.fa -outmat mat.txt
```

Note 8 contains instructions in case the above listed command-line fails to execute.

- Assuming that the Fastcompare output from **Subheading 3.1.** has been stored in a file named `7mer_scores.txt`, the command-line for running the rest of the randomization procedure and calculating a z-score for all k -mers is:

```
perl do_fastcompare_randomization -kmers 7mer_scores.txt
-inmat mat.txt -fasta1 ce_u_2000.fa -fasta2 cb_u_2000.fa
-nbrepeats 100 -outz 7mer_zscores.txt
```

The `-inmat` specifies the gene-specific substitution matrices to be used (here `mat.txt`, the output of `do_fastcompare_alignment`). The `-nbrepeats` option specifies the number of times the randomization procedure is to be repeated (here, 100). The `-outz` option specifies the file in which to store the z-scores, which will resemble the following (the second column contains the z-scores):

```
CTGCGTC 111.26
AGACGCA 87.70
GAGACGC 75.54
CTTATCA 74.05
AAATCGA 68.16
TACGTCA 66.46
AAGAAGA 63.92
TGA CTCA 63.67
AATCGAT 60.21
CCGCCCA 53.75
ACGCAGA 53.51
TGTTTGC 51.56
...
```

Note that it is possible to calculate z-scores for several lists of k -mers, e.g., for 7-, 8-, and 9-mers:

```
perl do_fastcompare_randomization -kmers
7mer_scores.txt,8mer_scores.txt,9mer_scores.txt -inmat
mat.txt -fasta1 ce_u_2000.fa -fasta2 cb_u_2000.fa -nbrepeats
100 -outz 7mer_zscores.txt,8mer_zscores.txt,9mer_zscores.txt
```

where the `-kmers` and `-outz` options are followed by file names separated by commas (without spaces).

Note that the script deduces from the input *k*-mer lists (e.g., `7mer_scores.txt`) whether gaps should be used or not. Also, mRNA analyses are conducted according to **Note 6**.

3.5. *k*-mer Extension Procedure

1. As indicated in the introduction, *k*-mers can be extended based on conservation scores (e.g., when using a raw conservation score threshold), or based on z-scores when available. In what follows, we extend 7-mers based on raw conservation scores. We assume here that the file named `best_7mer_scores.txt` contains the most conserved 7-mers, i.e., all 7-mers whose raw conservation score is above a chosen score threshold. Note that the file should also contain the conservation scores along with the *k*-mers. We attempt to extend these 7-mers into more conserved 8- or 9-mers that contain them, using the following command-line:

```
perl do_fastcompare_extension -kmers best_7mer_scores.txt
-otherkmers 8mers_scores.txt,9mers_scores.txt
```

where the `-otherkmers` option is followed by the names of the files containing the conservation scores for 8- and 9-mers, separated by commas (without spaces). The output (to screen) is the following:

AGACGCAG	1496	1722	655	407.43
CTGCGTCTC	1198	1468	518	381.94
AGACGCAGA	1145	1367	455	320.01
CGACACTCC	239	312	107	241.46
CTTATCA	3601	3771	1753	234.25
AATCGAT	4037	3578	1832	228.76
TCTTATCA	1941	2089	720	218.57
ATGAGTCA	827	1042	282	214.50
GCAAACAC	1035	948	302	213.58
...				

Note 9 provides additional explanations for the previously listed output. The command-line for extending 7-mers based on z-scores is identical, except that conservation score files are replaced by z-score files:

```
perl do_fastcompare_extension -kmers best_7mer_zscores.txt
-otherkmers 8mers_zscores.txt,9mers_zscores.txt
```

Finally, mRNA analyses are conducted according to **Note 6**, i.e., using the `-singlestrand 1` option.

3.6. Generating Conserved Sets, *k*-mer Distances, and Orientations

1. The conserved set of a given *k*-mer, e.g., AGACGCAG, can be generated by the following command (gene names will be stored in text file g.txt):

```
do_fastcompare_conserved_set -fasta1 ce_u_2000.fa -fasta2
cb_u_2000.fa -kmer AGACGCAG -cons g.txt
```

2. If desired, the same program also returns the positions and orientations of all occurrences of the input *k*-mer within the sequences from the conserved set (in the first species only, i.e., in `ce_u_2000.fa` here). This is accomplished by the `-pos` and `-ori` options, respectively (each followed by a file name in which the program will store the positions and orientations).

```
do_fastcompare_conserved_set -fasta1 ce_u_2000.fa -fasta2
cb_u_2000.fa -kmer AGACGCAG -pos p.txt -ori o.txt
```

Note 10 contains additional information on the positions and orientations outputted by the previously listed command. As usual, refer to **Note 6** for mRNA analyses.

3.7. Assessing *k*-mer Co-Conservation

1. The `do_fastcompare_coconservation` utility calculates conservation scores for pairs of *k*-mers, in general the high-scoring *k*-mers obtained in **Subheadings 3.1., 3.2., 3.3., or 3.6.** To focus on heterotypic interactions, it is recommended to examine pairs of *k*-mers that do not overlap by more than a few nucleotides (determined by the `-maxov` option, e.g., four below). The following command-line determines the conservation scores of all such pairs (with the input *k*-mers stored in the `best_kmers.txt` file):

```
do_fastcompare_coconservation -fasta1 ce_u_2000.fa -fasta2
cb_u_2000.fa -kmers best_kmers.txt -maxov 4
```

which produces the following output:

```
CTGCGTCTC CTCTCTC 277 453 86 119.72
CTTATCA CTCTCTC 439 673 104 78.47
CTGCGTCTC TCTTCTTC 157 334 44 68.73
TCTTCTTC TCATCATC 281 427 62 67.33
TCTTCTTC CTCTCTC 468 1097 131 65.68
CTTATCA TGATAAC 407 475 74 60.15
```



```

CTGCGTCTC  CTTATCA  151  225  34  58.53
CTGCGTCTC  AATCGAT  197  230  35  51.00
CTTATCA    ACAAACA  461  576  78  46.03
...

```

Note 11 provides additional explanations on the previously described output. As usual, refer to **Note 6** for mRNA analyses.

4. Notes

1. For all types of analyses, a minimum number of copies of k -mers within the same regulatory region, greater than one, can also be specified using the `-nbcopies` option. The number of k -mers shown in the output can also be limited to the best m k -mers using the option `-limit m`.
2. The two sequence files are required to be in Fasta format and should contain exactly the same number of sequences, in the exact same order (i.e., ordered by orthologous pairs). However, orthologous upstream regions need not have the same name.
3. Running time depends on the number of genes, the length of the sequences and the number of k -mers for which a conservation score is to be calculated. In general, it takes up to a few minutes. If it takes longer, verify the format and number of sequences (`grep ">" ce_u_2000.fa | wc -l` provides a quick way to count the number of sequences in a Fasta file in Unix). Also, check the file that contains the k -mers to make sure that k -mers have the right length. Also make sure that the file is in the right format, i.e., one k -mer per line (nothing else).
4. Each line in the output represents (from left to right) a k -mer, the number of genes having that k -mer in their upstream region in the first species, the number of genes having the same k -mer in their upstream region in the second species, the number of genes that have the k -mer in both species, and the corresponding conservation score.

Consider the highest conserved 7-mer, CTGCGTC. It is present at least once in 1980 and 2184 upstream regions, in *C. elegans* and *C. briggsae*, respectively. The number of upstream regions for which CTGCGTC is present in both species is 813. Given that the total number of orthologous upstream regions considered here is 10,894, the negative logarithm of the hypergeometric p -value is 295.14.

5. In a Unix or Mac OS X environment, if the Fastcompare executable is located within the current directory, it is in general required to specify that the executable has to be executed from that current directory, using `./fastcompare` instead of `fastcompare`. Also, make sure that the program has executable permissions (otherwise use `chmod +x fastcompare` to set them). Finally, it may be necessary to recompile the program to the specific platform (see README file in the distribution for instructions).

6. Common errors for mRNA analyses include using erroneous k -mer lists and omitting the `—singlestrand 1` option. Make sure to use `7mers_rna.txt` and not `7mers_dna.txt` (or equivalent files for other k -mer sizes)
7. Note that randomizations are highly recommended, but not always necessary. As described in the introduction, the number of conserved k -mers to retain for further investigation can be selected on the basis of independent validation data, such as gene expression, functional annotation enrichment, or complementarily to microRNAs (**1,3**).
8. The script uses Perl and ClustalW. If it fails to execute correctly, make sure these packages are installed on the machine, and are accessible from the command-line. In Unix, this may involve adding the directories that contain the ClustalW and Perl executables to the `$PATH` environment variable. In Windows, it is advised to place the ClustalW executable in the current directory. Also, this scripts involves computationally costly alignments and may take several hours to complete.
9. Here is an illustration of the behavior of the extension procedure on an example. The most conserved 7-mers in the initial list of 7-mers in **Subheading 3.1.**, CTGCGTC (reverse complement is GACGCAG), has a conservation score of 295.14. However, one of the 8-mers, AGACGCAG, contains CTGCGTC and has a higher score of 407.4. Therefore, the 7-mer is replaced by its better scoring 8-mer. Because the second k -mer in the initial list, AGACGCA, is a substring of the new 8-mer, it is removed from the list.
10. Note here that `do_fastcompare_conserved_set` used with the `—pos` and `—ori` options reports the position and orientation of all k -mer occurrences from the conserved set, as some upstream region may contain more than one occurrence.
11. The output format is similar to the format described in **Note 4**. Each line in the output represents (from left to right) the first k -mer and second k -mer for which co-conservation is to be assessed, the number of genes having both k -mers in their upstream region in the first species, the number of genes having both k -mers in their upstream region in the second species, the number of genes that have both k -mers in both species, and the corresponding conservation score.

Acknowledgments

The authors are grateful to Chang S. Chan and Kellen Olszewski for critical reading of preliminary versions of this document. The authors are also grateful to members of the Tavazoie group for insightful discussions. Saeed Tavazoie is supported by National Institutes of Health, National Science Foundation, and Defense Advanced Research Projects Agency.

References

- 1 Elemento, O. and Tavazoie, S. (2005) Fast and systematic genome-wide discovery of conserved regulatory elements using a non-alignment based approach. *Genome Biol* **6**, R18.

- 2 Pritsker, M., Liu, Y., Beer, M., and Tavazoie, S. (2004) Whole-genome discovery of transcription factor binding sites by network-level conservation. *Genome Res.* **14**, 99–108.
- 3 Chan, C. S., Elemento, O., and Tavazoie, S. (2005) Revealing posttranscriptional regulatory elements through network-level conservation. *PLoS Computational Biology* **1**, e69.
- 4 Wasserman, W. W., Palumbo, M., Thompson, W., Fickett, J. W., and Lawrence, C. E. (2000) Human-mouse genome comparisons to locate regulatory sites. *Nat. Genet.* **26**, 225–228.
- 5 Blanchette, M., and Tompa, M. (2002) Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res.* **12**, 739–748.
- 6 Xie, X., Lu, J., Kulbokas, E., et al. (2005) Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. *Nature* **434**, 338–345.
- 7 Kellis, M., Patterson, N., Endrizzi, M., Birren, B., and Lander, E. S. (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature* **423**, 241–254.
- 8 Cliften, P., Sudarsanam, P., Desikan, A., et al. (2003) Finding functional features in *Saccharomyces* genomes by phylogenetic footprinting. *Science* **301**, 71–76.
- 9 Lee, T. I., Rinaldi, N. J., Robert, F., et al. (2002) Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science* **298**, 799–804.
- 10 Matys, V., Kel-Margoulis, O. V., Fricke, E., et al. (2006) TRANSFAC and its module TRANSCmpel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**, D108–D110.
- 11 Griffiths-Jones, S. (2004) The microRNA Registry. *Nucleic Acids Res.* **32**, D109–D111.
- 12 Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.
- 13 Tavazoie, S., Hughes, J., Campbell, M., Cho, R., and Church, G. (1999) Systematic determination of genetic network architecture. *Nat. Genet.* **22**, 281–285.
- 14 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 15 Birney, E., Andrews, D., Caccamo, M., et al. (2006) Ensembl 2006. *Nucl. Acids Res.* **34**, D556–D561.
- 16 Altschul, S., Madden, T., Schaffer, A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25**, 3389–3402.

- 17 Birney, E., Clamp, M., and Durbin, R. (2004) GeneWise and Genomewise. *Genome Res* **14**, 988–995.
- 18 Jacobs Anderson, J. S., and Parker, R. (2000) Computational identification of cis-acting elements affecting post-transcriptional control of gene expression in *Saccharomyces cerevisiae*. *Nucleic Acids Res.* **28**, 1604–1617.
- 19 O'Brien, K. P., Remm, M., and Sonnhammer, E. L. (2005) Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucl. Acids Res.* **33**, D476–D480.
- 20 Stein, L., Bao, Z., Blasiar, D., et al. (2003) The genome sequence of *Caenorhabditis briggsae*: a platform for comparative genomics. *PLoS Biol* **1**, E45.

Phylogenetic Footprinting to Find Functional DNA Elements

Austen R. D. Ganley and Takehiko Kobayashi

Summary

Phylogenetic footprinting is a powerful technique for finding functional elements from sequence data. Functional elements are thought to have greater sequence constraint than nonfunctional elements, and, thus, undergo a slower rate of sequence change through time. Phylogenetic footprinting uses comparisons of homologous sequences from closely related organisms to identify “phylogenetic footprints,” regions with slower rates of sequence change than background. This does not require prior characterization of the sequence in question, therefore, it can be used in a wide range of applications. In particular, it is useful for the identification of functional elements in noncoding DNA, which are traditionally difficult to detect. Here, we describe in detail how to perform a simple yet powerful phylogenetic footprinting analysis. As an example, we use ribosomal DNA repeat sequences from various *Saccharomyces* yeasts to find functional noncoding DNA elements in the intergenic spacer, and explain critical considerations in performing phylogenetic footprinting analyses, including the number of species and species range, and some of the available software. Our methods are broadly applicable and should appeal to molecular biologists with little experience in bioinformatics.

Key Words: Phylogenetic footprinting; noncoding functional DNA element; *Saccharomyces*; ribosomal DNA.

1. Introduction

The falling costs of DNA sequencing and the increasing abundance of genomic sequence data means that the ability to find functional elements from DNA sequences is an increasingly important part of molecular biology. This

is especially true for functional noncoding DNA elements, as there are not clear rules for their sequence patterns, unlike protein-coding genes. One of the most powerful methods for finding functional elements from sequence data is phylogenetic footprinting (1,2). The concept behind phylogenetic footprinting is simple: through time functional elements will undergo a slower rate of sequence change than nonfunctional elements, as they are functionally constrained (3). Therefore, if we compare sequence data between related species, we expect functional elements to stand out as “footprints” of sequence conservation against the background sequence variation of the nonfunctional regions (4). However, therein lies the trick: the key to successful phylogenetic footprinting is choosing species for comparison that have the appropriate level of relatedness to distinguish functional elements from the background.

The number of studies employing phylogenetic footprinting has increased rapidly in the last few years, and these studies have used phylogenetic footprinting to reveal a variety of functional elements. Most commonly, it has been used to identify elements involved in regulation of gene expression (5,6). However, it has also proved useful for detecting other functional elements, such as: finding genes that are difficult to detect by conventional gene-prediction tools (e.g., ref. 7); detecting functional elements not associated with gene coding (called NOCs), such as origins of replication and replication fork barrier (RFB) sites (8); and identifying elements that clearly must be functional but for which the functions are not known (e.g., ref. 9). Furthermore, the availability of whole genome sequences for related organisms means these analyses can be performed on a genomic scale, without explicit regard as to the function (e.g., ref. 10). Finally, increasingly more sophisticated tools for phylogenetic footprinting are becoming available (e.g., refs. 11–15), often as web-based software tailored for specific purposes. Here, we demonstrate how to perform a basic phylogenetic footprinting analysis, involving making multiple sequence alignments, producing visual plots of the level of sequence conservation across these alignments, and some subsequent analyses. This phylogenetic footprinting methodology is applicable to researchers wanting to identify functional elements in their sequences of interest. However, phylogenetic footprinting is also a powerful confirmatory tool: if the presence of a functional element is inferred through other means, phylogenetic footprinting can be used to see if this element is indeed conserved, and, thus, whether or not it is likely to be biologically relevant.

We provide a practical example of how phylogenetic footprinting can reveal functional elements by using the ribosomal RNA gene repeats (rDNA) from a number of *Saccharomyces* yeast species. The rDNA repeats are maintained in most eukaryotes as multiple tandem repeats (see Fig. 1). Each repeat unit

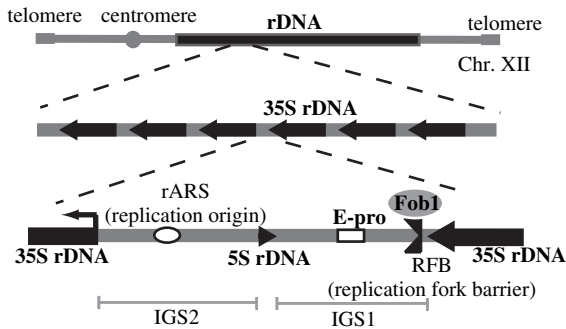


Fig. 1. Structure of the ribosomal RNA gene repeats (rDNA) in *Saccharomyces cerevisiae*. *S. cerevisiae* has approx 150 tandem rDNA units on chromosome XII. Each unit is 9.1 kb, and includes two rRNA genes (35S and 5S) and two intergenic spacers (IGS1 and 2). The IGS region is thought to be a hotspot for noncoding functional elements, and some are shown here: a replication origin (*rARS*) in IGS1, and a replication fork blocking (RFB) site and a noncoding promoter (*E-pro*) in IGS2. The RFB works as a recombinational hotspot to maintain rDNA copy number by amplification. This recombination is mediated by the Fob1 protein that associates with the RFB (27), and is regulated by *E-pro* (28).

consists of the 35S and 5S ribosomal RNA genes (the 5S rRNA gene is only variably present), and these are separated by spacer elements (16). Here, we analyze the intergenic spacer (IGS), located between adjacent 35S rRNA genes. In the *Saccharomyces* yeasts, this spacer is divided into two regions (IGS1 and IGS2) by the presence of the 5S rRNA gene (see Fig. 1). The IGS contains a variety of functional elements, such as promoters, terminators, an origin of replication, RFB sites, and other protein binding sites. Phylogenetic footprinting of the IGS identifies these previously identified elements as well as several other uncharacterized functional elements.

2. Materials

1. The first consideration for a phylogenetic footprinting study is the species to use (see Note 1). To aid in species selection, a phylogeny showing the relationship of species related to the species of interest is useful. We used the phylogeny of the *Saccharomyces* yeasts presented in Kurtzman and Robnett (17) to select a number of species for analysis. The relationships of the species we used here are shown in Fig. 2.
2. DNA sequences for the region of interest are obtained either through sequencing or from DNA databases or genome projects if available. It is recommended that a known conserved feature (e.g., part of a gene) be included in the region analyzed

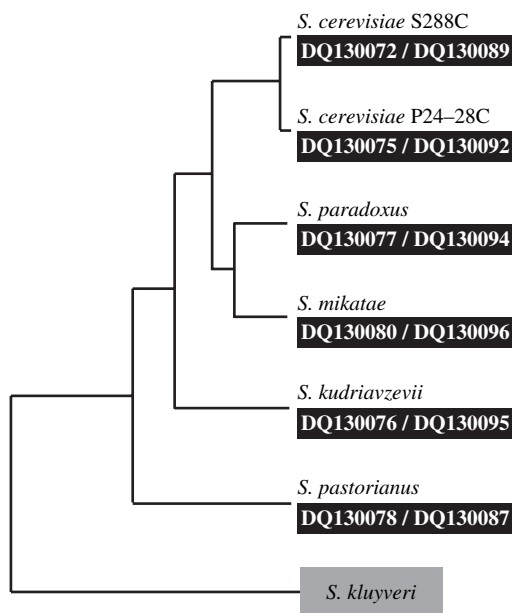


Fig. 2. Phylogeny of the *Saccharomyces* yeast species used in this study. The species/strain names are indicated, as are the accession numbers of the rDNA IGS1/IGS2 sequences (in boxes). For reference, an outgroup species (*Saccharomyces kluyveri*) is also shown in gray. The figure is adapted from Kurtzman and Robnett (17).

to act as a “landmark.” Make sure all sequences are in the same complement. We previously sequenced the IGS in two parts (IGS1 and IGS2, divided by the 5S rRNA gene) (8) for our chosen species, and the Genbank accession numbers of the sequences used here are shown in Fig. 2. All sequence manipulations were performed separately for the IGS1 and IGS2 sequences.

3. Software used in this analysis (see Note 2 for alternative software available):

- DNA sequence alignment: ClustalW ([18]; <http://align.genome.jp/>).
- Multiple alignment editing: GCG 11.0 Package (Accelrys, San Diego, CA).
- Similarity plots: SimilarityPlot from the GCG 11.0 Package (Accelrys).
- Conserved peak sequence analyses: TESS ([19]; <http://www.cbil.upenn.edu/cgi-bin/tess/tess?RQ=WELCOME>).

3. Methods

- The first step is the creation of multiple alignments of the sequences. We used FASTA formatted sequences in a single file, separately for IGS1 and IGS2 (Fig. 2) (see Note 3). The alignments should contain at least three sequences to prevent spurious alignment of regions that can occur in pairwise alignments.

- (a) Enter these FASTA files (separately for IGS1 and IGS2) into the web interface of the ClustalW multiple alignment program.
 - (b) Execute the multiple alignments using “DNA alignments” and the “slow/accurate” options. The output is set to “GCG(MSF)” format (*see Note 4*). Defaults are used for the remaining parameters (*see Note 5*).
2. The alignments are imported into the GCG 11.0 software package, and corrected manually for alignment error in the “Editor” mode (*see Note 6*).
3. To visualize the level of conservation across the alignment, similarity plots are created that graph the level of conservation in sliding windows across the alignment (*see Note 7*).
 - (a) Perform the “PlotSimilarity” function (under “Multiple Comparison”) in the GCG package on each of these alignments after selecting all the sequences. The relevant parameter is the size of the sliding window, with 15 bp being used in these examples (*see Note 8*). Defaults were used for the others.
 - (b) A graphical representation is then automatically created (*see Note 9*).
4. Once the similarity plots have been produced, known features can then superimposed onto these graphs (*see Note 10*). This produces visual representations of the alignments referenced to known features from one or more of the sequences (*see Note 11*) from which to base subsequent analyses, as shown in [Fig. 3](#).
5. Obviously, peaks of conservation are the interesting points. Reference to [Fig. 3](#) reveals several relevant points:
 - (a) Promoter elements, terminator elements, and particularly the gene coding regions themselves form conserved peaks.
 - (b) Most NOC elements also show remarkable coincidence with peaks of conservation, and the example of the RFB site is highlighted in [Fig. 4A](#).
 - (c) Other conserved peaks without known functions are also present.
 - (d) Some features that were previously predicted are not conserved, calling into question their biological meaning. These include the Abf1p binding site in IGS1 and one of the two Top1p binding sites predicted in the RFB.
 - (e) It is important to refer back to the original alignment to look for features that are masked in the similarity plots. For instance, the Reb1p binding site in IGS2 shows only a very small peak of conservation. However, reference to the alignment shows that this is a highly conserved motif embedded in a very variable region (*see Fig. 4B*).
 - (f) An important question is “how high should a peak of conservation be to imply biological relevance?” However, there is no way to know this *a priori*. We have used the average level of similarity as a proxy for this, but in reality this is arbitrary (*see Note 12*). Other evidence, including ultimately experimental evidence, is required to show biological function.
6. The real object of these studies is to determine what roles biologically functional sequence elements play. To help determine whether or not conserved peaks are

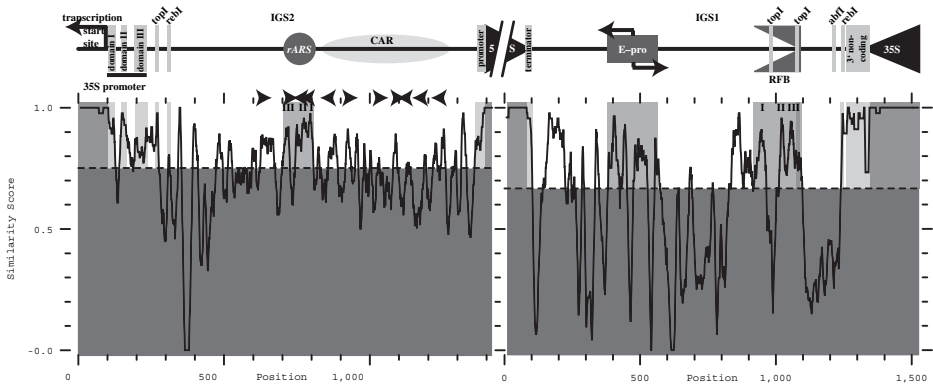


Fig. 3. Similarity plots of the IGS2 and IGS1 regions. Previously identified features from the IGS are indicated stylistically above the plots. These are the 35S and 5S rRNA genes, 35S promoter (29) and 3' noncoding region, 5S promoter (30) and terminator (31), origin of replication (*rARS*; [32]), replication fork barrier site (RFB; [27]) including two Top1p binding sites (33), expansion sequence bidirectional promoter (E-pro; [28]), cohesin associating region (CAR; [34]), and two Reb1p binding sites and an Abf1p binding site (35). Conserved peaks that match these elements are boxed in the similarity plots. Sequence matches to the origin of replication core consensus sequence found in conserved peaks (8) are indicated by small arrows above the plot. The dotted line represents the average level of similarity across the alignments. These are different between IGS1 and IGS2 because the average level of similarity is different between the two regions (largely as a result of IGS1 containing more indels). The region of the similarity plots below this line is shaded to emphasize the conserved peaks.

biologically functional and to gain clues as to their potential function, it is useful to examine the sequences of the conserved peaks in detail. Examples of some of the *in silico* examinations that can be performed include:

- Comparison of the conserved peaks with each other. The sequences of any conserved peaks can be used to search across the whole sequence to identify multiple copies of conserved peaks. We have performed this using the TESS website. This has the advantage of searching both directions and both strands, and can accept degenerate sequences. Using the “String-based Search Query” page, enter the sequence of the whole region in the “DNA Sequence(s)” Input option, and enter the conserved peak sequence(s) in the “Search My Site Strings” String Database option (see Note 13). The search criteria can be modified in the “String Scoring” option (see Note 14).
- Matches of conserved peaks with previously identified functional motifs. In many cases the functional elements detected by phylogenetic footprinting will

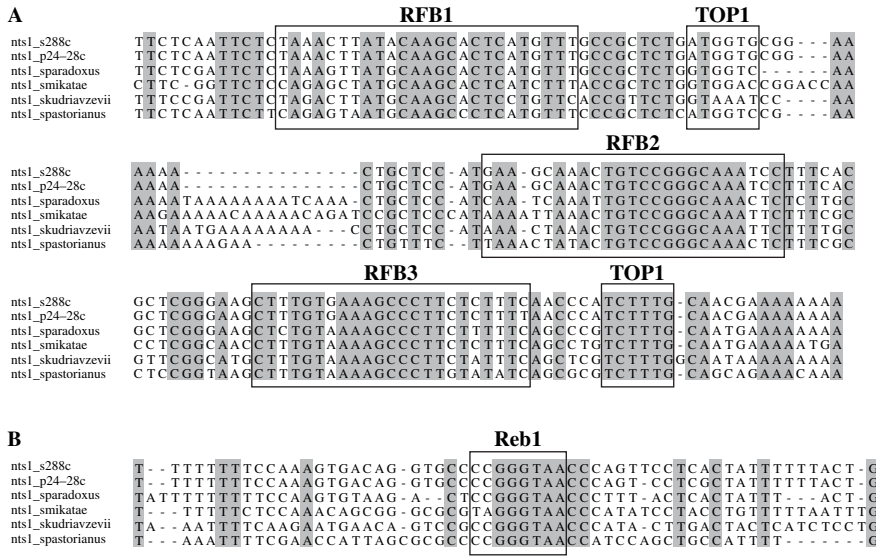


Fig. 4. Features of the plots. The actual alignments of two regions of interest are shown. Sites completely conserved in all six isolates are boxed in gray. (A) Conservation of the RFB site. The individual RFB elements and predicted Top1p binding sites are boxed. The three RFB elements all overlap with the regions of highest conservation. As can be seen, only one of the two predicted Top1p binding sites is conserved, calling into question the biological function of the other site. (B) Region surrounding the IGS2 Reb1p binding site. The Reb1p binding site is boxed. Although the peak in Fig. 3 is small, the alignment shows this Reb1p binding site is highly conserved, but is embedded in a background of variable sequence.

be protein binding sites, and some of these will have already been characterized. The easiest way to find such matches is to search a database of functional motifs. We have also used the TESS website for this analysis. In this case, the search was performed exactly as previously described, except that the Transfac database (a database containing protein binding site sequence motifs) was searched by checking the “Search TRANSFAC Strings” String Database option. Binding sites that fall on conserved peaks can then be identified. Alternatively, the sequence(s) of any conserved peaks found can be entered in the “DNA Sequence(s)” Input option, and the Transfac database searched using these. A combination of these methods showed that the series of conserved peaks between the origin of replication and the 5S rRNA gene all had matches to the origin of replication core consensus sequence [8]; see Fig. 3).

- (b) Again, there are a number of alignment editor software programs available. Many labs will already have software in use. Features that interactively indicate when bases are matched or color-code the different bases are useful for editing alignments.
 - (c) We have also used two freely available software programs to produce similarity plots. These are SynPlot ([21]; <http://www.sanger.ac.uk/Users/jgrg/SynPlot/>) for Unix systems and SWAAP ([22]; <http://www.bacteriamuseum.org/SWAAP/SwaapPage.htm>) for Windows systems (see also **Note 7**).
3. We recommend making several alignments using various subsets of the sequences. This allows the user to determine what level of phylogenetic relatedness is the most appropriate for the sequence of interest. As the level of relatedness between sequences decreases, the conserved peaks stand out more from the background noise. However, when the level of relatedness drops off too far, the conserved peaks will disappear. Furthermore, this also allows the user to detect different types of elements. Functional elements, for a number of reasons, are expected to vary in their rate of evolution. Therefore, some will only be seen when closely related species are used, whereas others will persist longer through evolutionary time.
 4. The output format selected will depend on the software used to make the similarity plots. Be sure to check what format the multiple alignments should be in for making the similarity plots. The sreformat program from the HMMER package (<http://hmmer.wustl.edu/>), available as a web-based service (<http://bioweb.pasteur.fr/seqanal/interfaces/sreformat.htm>), is useful for alignment format interconversions. Furthermore, be careful when switching between different computer systems (e.g., Mac/Unix/Windows) as line breaks and other characters are often coded differently. Opening and resaving text files in the native system is sometimes required.
 5. Usually the default parameters produce reasonable alignments. However, if there are many gaps the alignment can be poor, and adjusting the parameters (e.g., decreasing the gap penalty options) can help. In such cases, other alignment programs that use different algorithms (e.g., DIALIGN [23] and multi-LAGAN [24]) may also be useful.
 6. The alignment is the basis for phylogenetic footprinting; subsequent operations are just a way to visualize more easily the information from the alignment. Therefore, it is important to check the alignment for accuracy. However, there is no need to take this too far. Editing the alignment to produce a 1-bp match in a region of low similarity will not affect the overall results. It is more major errors that should be checked for. In our experience, most relevant alignment errors occur at sites where there are large insertions/deletions in some of the sequences.
 7. Here are some basic instructions for using freely available software (see **Note 2**) for making similarity plots:

- (a) SynPlot 0.5.3. This is a command-line program. The input file should be in aligned FASTA format (a2m format). Make sure the format is correct; this is not a simple FASTA format, but instead has the alignment gaps included in the FASTA sequences. The sreformat program (*see* **Note 4**) can convert to this format. Some of the SynPlot defaults will probably need changing. Change the sliding window size (-window) to 15 and the slide length (-increment) to 1 to give the same results as the GCG example. The -nuc_width setting may also be changed for viewing convenience. Specify the output file name (-out). The user can also include annotations for the sequences in GFF format if they exist.
 - (b) SWAAP 1.0.2. This Windows-based program can accept various alignment formats, although we have used the aligned FASTA format (as in SynPlot) most easily. The alignment should be opened (and reading frame arbitrarily selected even though this plays no role). At this stage, change the defaults under the “Parameters” menu if necessary, e.g., Window Size to 15 and Step Size to 1 to give the same results as the GCG example. To make the output similar to that made by GCG or SynPlot, the “Indel” option should be changed at least to “Treat Pairwise Indels as 5th Nucleotide,” or even “Remove All Indels” which removes all positions in the alignment that contain an indel. This is because indels seem to be treated as homologous characters in this software, thus giving somewhat different outputs. Then, create the similarity plot by implementing “Calculate Percent Identity Over Sliding Window” under the “Analyses” menu, using of course the “Nucleotide” option. This creates a table of the values from the sliding windows, which can be plotted directly in the program, or transferred to other software (e.g., Microsoft Excel) for plotting.
8. As the similarity plots are simple to produce at this stage, it is worth experimenting with the sliding window size. Remember, these plots are merely graphical representations of the underlying alignments for ease of visualization, so the window size can be varied for maximum utility. Smaller window sizes will better reveal small conserved motifs, however the differentiation from the background is less.
 9. It will probably be necessary to save the output in a format that can be used by subsequent programs for making figures. We have found that EPS format gives the best results in our situation. To save the file in this format, press “Print” once PlotSimilarity produces the image, and select the EPSF option from the “Output Device” item. Change the name in the “Port or File” item to the output name you want.
 10. Often, only the region of the plots above a certain threshold (e.g., average identity across the alignment) is shown in figures, highlighting the conserved peaks. This is for ease of reference, and also because the depth of the troughs are determined mainly by the number of indels, rather than the absolute level of variation per

se, and therefore are not necessarily biologically meaningful. We have shaded the lower part of the plots in **Fig. 3**.

11. In most cases these features will come from just one of the species in the alignment (in our example the features all come from studies on *Saccharomyces cerevisiae*). However, if information from several species in the alignment is available, obviously this should be used. Remember to use the alignment, not the original reference sequence, as the reference template to plot features, as it is likely gaps will have been introduced into the alignment.
12. The presence of large “valleys” in the similarity plot is usually the result of gaps in the alignment, and they are not necessarily the least-conserved regions, per se. The effect of gaps on the similarity plots can be checked by removing the gapped regions from the alignment, e.g., by using the Gapstrip tool at the Los Alamos National Laboratory HIV website (<http://www.hiv.lanl.gov/content/hiv-db/GAPSTRIP/gapstrip.html>).
13. Two methods can be used. In one, degenerate sequences of the conserved peaks can be made from the alignment. In the other, the sequences of each conserved peak from one reference species can be used as queries to the entire sequence from the same species. A useful way to represent degenerate sequences of conserved peaks is as sequence logos (**25**), and a tool for making these is available online (**[26]**; <http://weblogo.berkeley.edu>).
14. When searching for matches to the conserved peaks, the searches should allow for some mismatch, as conserved motifs are unlikely to be 100 % conserved.

Acknowledgments

This work was supported by grants 13141205, 17080010, and 17370065 from the Ministry of Education, Science and Culture, Japan, and by a Human Frontier Science Program grant.

References

- 1 Frazer, K. A., Elnitski, L., Church, D. M., Dubchak, I., and Hardison, R. C. (2003) Cross-species sequence comparisons: a review of methods and available resources. *Genome Res.* **13**, 1–12.
- 2 Hardison, R. C. (2003) Comparative genomics. *PLoS Biol.* **1**, 156–160.
- 3 Moses, A. M., Chiang, D. Y., Kellis, M., Lander, E. S., and Eisen, M. B. (2003) Position specific variation in the rate of evolution in transcription factor binding sites. *BMC Evol. Biol.* **3**, 19.
- 4 Hardison, R. C. (2000) Conserved noncoding sequences are reliable guides to regulatory elements. *Trends Genet.* **16**, 369–372.
- 5 Gumucio, D. L., Shelton, D. A., Bailey, W. J., Slightom, J. L., and Goodman, M. (1993) Phylogenetic footprinting reveals unexpected complexity in trans factor binding upstream from the ϵ -globin gene. *Proc. Natl. Acad. Sci. USA* **90**, 6018–6022.

- (c) Other things to look for in the conserved peak sequences include the presence of repeats, palindromic sequences, and regions of unusual sequence composition.
7. Our analyses of the rDNA IGS region from these *Saccharomyces* yeast species reveals that phylogenetic footprinting is a simple, yet extremely powerful method for detecting a variety of noncoding functional elements. Most previously characterized elements in the IGS were obvious as conserved peaks, and for some of those that do not form conserved peaks we have independent reasons to doubt their functionality. Furthermore, there are a variety of conserved peaks for which functions have not been previously described, and we have confirmed that some of these are indeed functional (8). Indeed, all the conserved peaks identified in the IGS by this method that we have tested to date are functional (8), attesting to the great power of phylogenetic footprinting. These IGS conserved peaks also represent elements with a variety of functions, therefore this technique is likely to be applicable to studies investigating many different kinds of functional elements. Finally, the method seems to be very specific: both the origin of replication and RFB sites are believed to contain three subelements, and in each case each of these subelements can be identified as its own conserved peak. Single protein binding sites also appear as conserved peaks. Given the simplicity of performing the analyses (once the sequences are obtained), phylogenetic footprinting promises to be an important component of molecular biology studies that look to relate noncoding DNA to biological action.

4. Notes

1. Choosing species with the appropriate level of evolutionary relatedness is the most critical part of phylogenetic footprinting. However, this is easier said than done; the most appropriate species can even change depending on what region is being examined. We recommend choosing species encompassing a range of relatedness, including very closely related species, and more distantly related species. A phylogeny is, therefore, very useful for this species selection. The total phylogenetic distance separating all species is likely to be the important parameter (20). In practice this means choosing a good phylogenetic range, rather than a lot of species; three species with appropriate levels of relatedness are likely to give better results than six species from a more restricted range. However, if the species are too distantly related, some or all of the conserved peaks will disappear. It may be necessary to return and choose a species with an intermediate level of relatedness if the results from the first pass are not satisfactory.
2. In most cases, there are alternatives available to the software we have used.
 - (a) There are a number of multiple alignment programs available, both as web-based services and as installed programs for personal computers. Any of these should be suitable for making the multiple alignments.

- 6 Hong, R. L., Hamaguchi, L., Busch, M. A., and Weigel, D. (2003) Regulatory elements of the floral homeotic gene *AGAMOUS* identified by phylogenetic footprinting and shadowing. *Plant Cell* **15**, 1296–1309.
- 7 Brachat, S., Dietrich, F. S., Voegeli, S., et al. (2003) Reinvestigation of the *Saccharomyces cerevisiae* genome annotation by comparison to the genome of a related fungus: *Ashbya gossypii*. *Genome Biol.* **4**, R45.
- 8 Ganley, A. R. D., Hayashi, K., Horiuchi, T., and Kobayashi, T. (2005) Identifying gene-independent noncoding functional elements in the yeast ribosomal DNA by phylogenetic footprinting. *Proc. Natl. Acad. Sci. USA* **102**, 11,787–11,792.
- 9 Dermitzakis, E. T., Reymond, A., Scamuffa, N., et al. (2003) Evolutionary discrimination of mammalian conserved non-genic sequences (CNGs). *Science* **302**, 1033–1035.
- 10 Kellis, M., Patterson, N., Endrizzi, M., Birren, B., and Lander, E. S. (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature* **423**, 241–254.
- 11 Frazer, K. A., Pachter, L., Poliakov, A., Rubin, E. M., and Dubchak, I. (2004) VISTA: computational tools for comparative genomics. *Nucl. Acids Res.* **32**, W273–W279.
- 12 Ovcharenko, I., Loots, G. G., Hardison, R. C., Miller, W., and Stubbs, L. (2004) zPicture: dynamic alignment and visualization tool for analyzing conservation profiles. *Genome Res.* **14**, 472–477.
- 13 Aerts, S., Van Loo, P., Thijs, G., et al. (2005) TOUCAN 2: the all-inclusive open source workbench for regulatory sequence analysis. *Nucl. Acids Res.* **33**, W393–W396.
- 14 Sinha, S., Blanchette, M., and Tompa, M. (2004) PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinform.* **5**, 170.
- 15 Nix, D. A. and Eisen, M. B. (2005) GATA: a graphic alignment tool for comparative sequence analysis. *BMC Bioinform.* **6**, 9.
- 16 Long, E. O. and Dawid, I. B. (1980) Repeated genes in eukaryotes. *Ann. Rev. Biochem.* **49**, 727–764.
- 17 Kurtzman, C. P. and Robnett, C. J. (2003) Phylogenetic relationships among yeasts of the ‘*Saccharomyces* complex’ determined from multigene sequence analyses. *FEMS Yeast Res.* **3**, 417–432.
- 18 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucl. Acids Res.* **22**, 4673–4680.
- 19 Schug, J. and Overton, G. C. (1997) *TESS: Transcription Element Search Software on the WWW*. University of Pennsylvania, Philadelphia, PA.
- 20 Moses, A. M., Chiang, D. Y., Pollard, D. A., Iyer, V. N., and Eisen, M. B. (2004) MONKEY: identifying conserved transcription-factor binding sites in multiple alignments using a binding site-specific evolutionary model. *Genome Biol.* **5**, R98.

- 21 Göttgens, B., Gilbert, J. G. R., Barton, L. M., et al. (2001) Long-range comparison of human and mouse *SCL* loci: localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences. *Genome Res.* **11**, 87–97.
- 22 Pride, D. T. and Blaser, M. J. (2002) Concerted evolution between duplicated genetic elements in *Helicobacter pylori*. *J. Mol. Biol.* **316**, 629–642.
- 23 Morgenstern, B. (2004) DIALIGN: multiple DNA and protein sequence alignment at BiBiServ. *Nucl. Acids Res.* **32**, W33–W36.
- 24 Brudno, M., Do, C. B., Cooper, G. M., et al. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 25 Schneider, T. D. and Stephens, R. M. (1990) Sequence logos: a new way to display consensus sequences. *Nucl. Acids Res.* **18**, 6097–6100.
- 26 Crooks, G. E., Hon, G., Chandonia, J. -M., and Brenner, S. E. (2004) WebLogo: a sequence logo generator. *Genome Res.* **14**, 1188–1190.
- 27 Kobayashi, T. (2003) The replication fork barrier site forms a unique structure with Fob1p and inhibits the replication fork. *Mol. Cell. Biol.* **23**, 9178–9188.
- 28 Kobayashi, T. and Ganley, A. R. D. (2005) Recombination regulation by transcription-induced cohesin dissociation in rDNA repeats. *Science* **309**, 1581–1584.
- 29 Musters, W., Knol, J., Maas, P., Dekker, A. F., van Heerikhuizen, H., and Planta, R. J. (1989) Linker scanning of the yeast RNA polymerase I promoter. *Nucl. Acids Res.* **17**, 9661–9678.
- 30 Challice, J. M. and Segall, J. (1989) Transcription of the 5S rRNA gene of *Saccharomyces cerevisiae* requires a promoter element at +1 and a 14-base pair internal control region. *J. Biol. Chem.* **264**, 20,060–20,067.
- 31 Brown, B. R., Bartholomew, B., Kassavetis, G. A., and Geiduschek, E. P. (1992) Topography of transcription factor complexes on the *Saccharomyces cerevisiae* 5S RNA gene. *J. Mol. Biol.* **228**, 1063–1077.
- 32 Miller, C. A. and Kowalski, D. (1993) *cis*-Acting components in the replication origin from ribosomal DNA of *Saccharomyces cerevisiae*. *Mol. Cell. Biol.* **13**, 5360–5369.
- 33 Burkhalter, M. D. and Sogo, J. M. (2004) rDNA enhancer affects replication initiation and mitotic recombination: Fob1 mediates nucleolytic processing independently of replication. *Mol. Cell.* **15**, 409–421.
- 34 Laloraya, S., Guacci, V., and Koshland, D. (2000) Chromosomal addresses of the cohesin component Mcd1p. *J. Cell Biol.* **151**, 1047–1056.
- 35 Morrow, B. E., Johnson, S. P., and Warner, J. R. (1989) Proteins that bind to the yeast rDNA enhancer. *J. Biol. Chem.* **264**, 9061–9068.

Detecting Regulatory Sites Using PhyloGibbs

Rahul Siddharthan and Erik van Nimwegen

Summary

PhyloGibbs is a program that uses Gibbs sampling to predict putative binding sites for transcription factors in DNA. It has two notable advances over previous algorithms for this task: it handles phylogenetically related sequence systematically, and it evaluates the significance of each predicted site via statistical sampling. In this article, we explain how to use PhyloGibbs effectively. We describe the essential command-line options in detail, and discuss other considerations that arise in practical situations.

Key Words: Gene regulation; binding sites; motif finding.

1. Introduction

Genes are stretches of DNA that code for biologically important molecules: proteins or RNA. They are transcribed to RNA by a special enzyme called RNA polymerase, and in the case of protein-coding genes, this RNA (mRNA) is translated by the ribosomal machinery into a protein. The recruitment of the RNA polymerase and the initiation of its transcription is generally regulated by additional proteins called transcription factors (TFs): these bind to the DNA, usually upstream of the start site of the gene, by recognizing patterns or “motifs” in the DNA sequence, and interact with the RNA polymerase (and often with one another) to either enable or inhibit the transcription of the gene. In general, the combinatorial regulation of a gene by several TFs is a complex process that can only partially be understood by analyzing the DNA sequence itself. However, understanding the process is a hugely important problem.

From: *Methods in Molecular Biology*, vol. 395: *Comparative Genomics*, Volume 1
Edited by: N. H. Bergman © Humana Press Inc., Totowa, NJ

The development of an organism from an embryo to a fully grown adult, the differentiation of cells into different tissue types, the internal functioning of the cells, their response to external stimuli and stresses, are all results of carefully orchestrated sequences of gene regulation events.

This article discusses the use of the motif-finding algorithm Phylogibbs ([1](#)) as a tool for detecting regulatory sites in DNA or RNA sequences. There are many motif finding tools available, but most of these are variants of two approaches: Gibbs sampling, first introduced in this context by Lawrence et al. ([2](#)), and expectation maximization of mixture models, as in the MEME algorithm of Bailey and Elkan ([3](#)). Our algorithm PhyloGibbs is a Gibbs sampling algorithm that incorporates several enhancements. In particular, it can operate on DNA sequences from related species, taking the phylogenetic relationships between the species into account.

The organization of this article is as follows. First, we discuss general issues of how genes and groups of genes are regulated and how regulatory regions in various genomes are organized. Next, we discuss usage of PhyloGibbs on isolated stretches of DNA, on sets of promoters of coregulated genes, and last, on phylogenetically related sets of DNA sequences. We will describe in detail how to prepare the input files, how to set command-line options and how to read the output files. Finally, we have set up a web-interface to the program, as well as a database of genome-wide binding site predictions made with PhyloGibbs and we will briefly describe both of these. The article is aimed at end-users of the code, who are not necessarily computational biologists. The mathematical underpinnings and internal workings of PhyloGibbs are discussed only minimally here: interested readers are referred to our main paper on the subject ([1](#)).

As with most tools in computational biology, the performance of PhyloGibbs on any particular data-set is highly dependent on the amount of prior information that the user provides about the data-set. The more specific the prior information provided to the algorithm, the better it knows what to look for and where to look for it. PhyloGibbs has a large number of command-line options that allow the user to specify the expected number of TFs regulating the gene(s), the number of binding sites that each TF is expected to have, the estimated length of an individual binding site, the base composition biases in the parts of the DNA not corresponding to binding sites (i.e., a background model), and—in case of phylogenetically related sequence—the phylogenetic tree relating the species from which the input sequences derive. There are many further options for fine-tuning the running of the program. In most cases, the performance of the program is robust against changes in the parameters, whose default values

are meant to be “reasonable” for typical data, but wildly inaccurate choices of parameters may well lead to nonsensical results.

Except where indicated, this article applies to v1.0 of PhyloGibbs, which was released with the paper in December, 2005. Some small changes and additional changes have been implemented since, but all information in this paper applies to the latest released versions. Versions with more significant changes and extensions are currently being developed and these will be released at a later date. Users of these future versions should carefully consult the corresponding manual.

2. Organization of Regulatory DNA

In the simplest unicellular organisms, the prokaryotes, most of the genome is protein-coding and there is relatively little intergenic region (ranging from tens to hundreds basepairs between genes). Given the difficulty (though not impossibility) of evolving a regulatory site amid the constraints of a coding region, it is believed that most regulatory sites occur in intergenic regions. Often a single regulatory region controls a whole set of genes, called an operon, that are transcribed in one go. These genes are placed successively on the DNA with very little spacing, and typically cooperate in a common pathway. Bacterial genes are often regulated by only a few factors, with one or two binding sites for each factor. These sites tend to be large, often more than 20 nucleotides wide, and often exhibit near reverse-complement symmetry. The latter is a result of the fact that the TF that binds to the sites is a homodimer that contacts the DNA using two identical but oppositely oriented domains on opposite sites of the DNA.

In simple unicellular eukaryotic organisms such as yeast there are typically a few hundred basepairs of intergenic region between each pair of genes, all of which could be regulatory. Binding sites might be much more numerous in these regions than in prokaryotes, and the sites themselves are smaller than in bacteria, typically on the order of about 10 nucleotides wide.

In higher eukaryotes there are many kilobases of intergenic region between genes, but it is generally assumed that only small portions of these regions have regulatory function. Regulatory modules are of the order of a few hundred basepairs to a kilobase or two in length, and a given gene may be regulated by several such modules, some of which may be many kilobases upstream or downstream of the gene or in introns. In addition, it is believed that the “proximal promoter” of roughly the first kilobase upstream of transcription start also contains a high density of regulatory sites.

When applying motif-finders to data from higher eukaryotic organisms it is generally advisable not to run on entire intergenic regions but to somehow focus on regions that are more likely to contain a high density of binding sites. This could for instance be done by identifying the transcription start sites of the genes of interest and running only on one or two kilobases immediately upstream. When applying motif finding to *cis*-regulatory modules it is necessary to locate these modules as well as possible, either experimentally or via a module-prediction program. Several computational approaches to module prediction exist, all based on clustering of predicted binding sites for TFs whose binding specificity is known: for some recent examples, *see* **refs. 4–9**. Finally, even within a given module, the gene may be under complex combinatorial control of several TFs. Some of these may act as cofactors to other factors, and, therefore, not directly contact the DNA, or contact it only partially.

3. PhyloGibbs: General Ideas

PhyloGibbs, like other motif finders, reads in DNA sequences, and assumes that certain small stretches are binding sites for TFs (or regulatory sites of another nature), whereas other regions are generic DNA that can be described by a “background model.” Its task is to predict where the regulatory sites are and which of these are sites for the same factor, and to assess the significance of these predictions.

Given a set of DNA sequences, every short sequence segment is a potential binding site for a TF. Thus, all potential answers to the question “where are the binding sites?” consist of configurations of an arbitrary number of short sequence segments (the regulatory sites) embedded in the “background” of the rest of the DNA. In addition, specifying which sites are binding sites for the same factor consists of partitioning the binding sites into groups, with each group corresponding to a regulatory factor. The PhyloGibbs algorithm assigns a Bayesian “posterior probability” to every possible such binding site configuration based on how likely it is to observe the sets of hypothesized binding sites under the assumption that each set contains regulatory sites for a common factor, and how likely it is to observe the rest of the DNA under a given background model.

Given the assumptions of PhyloGibbs’ model the best binding site configuration is thus the one that has maximal posterior probability. However, as can be easily imagined, the space of all binding site configurations is too large to be searched exhaustively. We search the space using a Gibbs sampling strategy, first introduced in the context biological motif-finding by Lawrence et al. (2).

This is an example of a more general technique called “Markov chain Monte-Carlo sampling” where, instead of exhaustively searching a state space, one starts from a random state and moves through the space in a stochastic fashion such that, in the limit of long time, each state is visited in proportion to its posterior probability.

The algorithm operates in two phases. In the first phase, the best configuration is found by a procedure called “simulated annealing.” A fictitious temperature is introduced and the system is slowly “cooled down,” which, in this context, means that as time increases, more and more weight is given to the configurations with highest posterior probability. At the end of this procedure the system will be “frozen” into the configuration with the highest posterior probability that it could find during its search. Note that this procedure always yields an answer, which may or may not be significant. In the second phase of the algorithm the significance of the best configuration found is assessed by performing another sampling run (without cooling) and comparing the best configuration with the configurations that are visited during this sampling run. During this second phase “tracking” statistics are gathered on how often any given site is coclustered with one of the groups of sites in the “best configuration.”

In its default mode of operation, which is adequate for most purposes in our experience, PhyloGibbs is asked to search for a fixed number of binding sites for a given maximum number of different factors. For example, PhyloGibbs may be asked to search for a total of nine binding sites for at most three different factors. It will then only consider binding sites configurations with a total of nine sites for one, two, or three different factors. (In v1.0, it would only search configurations for precisely three different factors; this has been relaxed in the current release.) For simplicity, we restrict ourselves to this usage in the examples next. Alternatively, however, one may allow PhyloGibbs the freedom to vary the total number of sites and the number of factors by using the `-c` option (see **Section 7**). In this mode the user must provide the program with the expected total number of sites and the expected total number of factors.

4. Running PhyloGibbs on a Single Sequence of DNA

Two issues that apply to all motif finders must be kept in mind, here and in more complex cases. The first point is that it is impossible to detect a single isolated binding site in a single sequence: to all computational motif finders binding sites are instances of “patterns” in the sequence, that is, they represent surprising similarities among sequence segments. A single example does not

constitute a “pattern” (mathematically, PhyloGibbs gives it nearly the same score as background) except in extreme cases, such as an island of C’s and G’s in a sea of A’s and T’s. It is thus important to provide the algorithm with input data that contains enough examples of the “pattern” for the algorithm to be able to discover it. Although a single promoter or enhancer sequence in eukaryotes typically contains multiple binding sites for each TF, this is not guaranteed. When running on a single sequence there is thus always the danger that there are too few examples of each binding site for the algorithm to discover it. But sometimes this is the best one can do.

The second point to note is that the input sequence should contain as much “signal” (actual regulatory sequence) and as little “noise” (background sequence, “junk DNA”) as possible. This is because, given enough sequence, copies of any pattern will be found by chance: for example, in completely random sequence, the pattern “ACATT” will occur every 1024 bases, and in nonrandom sequence such as actual DNA sequence, it may well occur much more often. So in order for the algorithm to discover the binding sites they need to occur significantly more often than one would expect by chance from a sequence with the same length and overall base composition.

As previously discussed, in bacteria, and even in many single-celled eukaryotes such as yeast (*Saccharomyces cerevisiae*), there is not very much intergenic sequence and one may assume, without much harm, that all of it is regulatory. But in higher organisms there are many kilobases of intergenic sequence, and one needs to locate regulatory modules as precisely as possible, either experimentally or via module-prediction programs, before running a motif finder.

The following are the parameters that must be understood to use PhyloGibbs. They are also used when running on multiple genes or multiple species, as discussed in the following two sections.

1. **-f filename**: the name of the file from which the input sequence is read. It is a fasta-format file, where headers (names or identifiers of the sequence) that begin with the “>” character precede raw sequence. When running on a single sequence, or (as in the next section) on regulatory sequences for multiple genes in the same species, the content of the header is reproduced verbatim in output files but not otherwise used. For phylogenetically related sequences these headers are important (see **Section 6.3**).
2. **-m motif_width**: the width (integer) of the motif being searched for: the default is 10. For most eukaryotic motifs the width ranges roughly from 6 to 16 and for prokaryotes the widths range roughly from 16 to 26. It is better to slightly overestimate the width of the motif than to underestimate it.

3. `-o output_file`: the name of the file into which the simulated-anneal results are written.
4. `-t tracked_output_file`: the name of the file into which the tracking results are written. Usually, of the two output files, this is the file that the user will be most interested in.
5. `-F bgfile`: the name of an optional auxiliary file (in fasta format) to be used for estimating the base composition of background sequences. For example, this file could contain large quantities of intergenic DNA sequence from the species being studied. If not supplied, the background model is estimated from the input sequence itself. This can hurt performance especially when the input sequence contains a high density of true sites.
6. `-N ncorrel`: the number of preceding bases (integer) that a given background base is assumed to correlate with. The default is 1. As special cases, 0 means use uncorrelated background with base counts estimated from the input file or auxiliary background file, and -1 means use uncorrelated background with probabilities of 0.25 for each base (totally random background model). As a further special case, one may supply, instead of integers, a list of four floats separated by commas (no spaces), which indicate the background probabilities of A,C,G,T, respectively (they will be automatically normalized). For example, `-N 0.3,0.2,0.2,0.3` would be roughly suitable for yeast.
7. `-I numlist` or (in newer versions) `-y sites -z factors`: these options are used to specify the number of binding sites and the number of different factors that the algorithm should assume exist. When using `-I` the variable, `numlist` should be a comma-separated list of integers, without spaces, with one entry for each factor. For example, `-I 3,5,4` tells the algorithm to start with an initial configuration that has three motifs, with three, five, and four binding sites, respectively. During sampling the total number of sites (12 in this case) will remain constant and the number of factors will remain three or less. However, the algorithm may choose to redistribute the number of sites per factor, so that one could obtain configurations with 4,4,4 sites, or it may even reduce the total number of factors and evolve to a configuration such as 8,4. In particular, the latter could happen if there happen to be only two strong motifs in the input data that together have 12 or more sites. Of course one in general does not accurately know the total number of sites and factors beforehand. Typically it is a good idea to slightly overestimate the number of factors (keeping in mind that although one might be looking for sites for a single factor only, there may be other factors that are also represented with multiple sites in the input data). For example, if there are only two factors with significant numbers of binding sites, `3,3,3,3,3` could well evolve into `8,7`, whereas conversely, if there really were five factors with only three sites each, `8,7` will give poor results. Inaccurate estimates of the total number of sites will generally not hurt the results. Through the tracking phase a posterior probability is assigned to each binding site and, for reasons explained in the paper ([1](#)), the “prob” (probability) numbers reported in the

tracked-output file represent the best possible estimate of the posterior probabilities, given the prior assumptions, that these are binding sites. So binding sites with weak evidence will still be easily distinguishable from reliable predictions. Instead of the `-I` option the total number of binding sites and the maximal number of allowed factors can also be specified using `-y` sites `-z` factors, i.e., `-y 12 -z 3` for the previously described example.

8. `-S nsteps`: this is the total number of steps in the tracking phase, each step consisting of a predetermined number of Gibbs-sampling moves of each type. Unless overruled, this parameter also controls the length of the initial simulated-anneal phase. Shorten it for quicker results, or increase it for more accurate results (in the infinite-time limit the program should give the best-possible answer given the assumptions).

5. Running PhyloGibbs on “Regulons,” Sets of Genes That are Believed to be Coregulated

Most TFs regulate multiple genes. In particular, responses to many events—stages in development and differentiation, checkpoints in the cell cycle, responses to different stimuli or stresses—involve the concerted activation or deactivation of sets of genes by common TFs. In some cases, there may be clear experimental evidence that indicates that a particular TF regulates a certain set of genes. For example, chromatin immunoprecipitation experiments can be used to identify intergenic regions that are bound by a common TF. Gene expression data from microarray experiments can also be used to identify genes that are under control of a common TF, e.g., by measuring the changes in gene expression upon activation or inactivation of the TF. When such genome-wide data are combined with more direct biological knowledge obtained from other experiments, they can provide a relatively reliable estimate of the set of genes regulated by a common TF. In other cases, the identity of the regulating TF or TFs may not be known, but the microarray may simply record the genome-wide changes in expression under a certain set of perturbations or conditions. These data can be “clustered” (for an overview of several general techniques, *see*, e.g., [ref. 10](#)) to determine which sets of genes show notable correlation in expression. It is reasonable to assume that the genes in such gene-expression clusters are regulated by common TFs. Finally, even if one is only interested in the regulation of one particular gene, running PhyloGibbs on a larger set of sequences that includes upstream regions of genes that are coregulated with the gene of interest may considerably improve the performance of the algorithm. This is because it ensures that the input sequences will contain multiple binding sites for the motif(s) of interest, and it will also generally improve the “signal-to-noise” ratio of the data.

Having arrived at a set of likely coregulated genes, one can extract likely regulatory regions for them, (either regions immediately upstream of transcription start or regions around regulatory modules), and then run *PhyloGibbs* on these sets of sequences with the aim of discovering the regulatory sites within them. The input file, supplied as before with the `-f` option, must be a single fasta file containing all sequences. The headers on the sequences do not matter for the functioning of *PhyloGibbs* and are simply used to name the sequences (i.e., given informative names will make the output files easier to read).

The main parameters that require some thought in this setting are the total number of sites and the number of factors that are specified either through the “initial conditions” parameter `-I` or through `-y` and `-z`. For example if one expects three binding sites per factor in a promoter one may expect nine binding sites in three promoters. However, it may be better to specify a slightly larger number of factors and reduce the number of sites per factor. For example, if there are really three factors each with six well-defined binding sites, the choice `-I 6,6,6` (or `-y 18 -z 3`) should capture them, but one cannot be expected to know the number of sites and factors in such detail in advance; however, the choice `-I 3,3,3,3,3,3` (or `-y 18 -z 6`) will typically do almost equally well. If the 18 sites for the three factors are distinguishable enough the algorithm will simply choose to populate only three of the six allowed motifs with sites.

This suggests that one may as well specify `-y 18 -z 18`, that is, allow as many factors as there are sites. However, by doing this one would significantly increase the number of configurations that *PhyloGibbs* has to search through (thus incurring a major performance penalty and possibly reducing the significance of predictions). Moreover, most of the configurations that would be added correspond to configurations with a very high number of colors, which we know in advance are very unlikely to be correct. It is generally a good idea to set the number of sites to a reasonable guess of the total number of sites in the data, and to set the number of factors to a number that is at the upper bound of the range of factors that one expects.

6. Running on Phylogenetically Related Sequence

The original motivation for developing *PhyloGibbs* was the wish to run on sets of orthologous sequences from related species and to incorporate information on evolutionary conservation of the sites into the scoring of binding site configurations. Binding sites on related sequences may be orthologous, i.e., they evolved from a common ancestor site, and in that case it would be

inappropriate to treat them as independent occurrences (this also applies to the background sequence, which may show spurious similarities because of their common evolutionary origin). PhyloGibbs handles the situation by requiring preprocessing of the input sequence by a multiple-alignment program to identify conserved regions; it then treats unconserved sequence as usual, but treats sites in conserved sequence as descendants of a single ancestral site. To score these descendants, phylogenetic parameters need to be supplied. PhyloGibbs then searches through parses consistent with the alignment, scoring them using the phylogenetic parameters, and as before, finds a “best parse” via a simulated anneal, and assesses the significance via tracking. Only the internal definition of a “site” changes, so in the output files individual “sites” will now often consist of alignments of orthologous sites from multiple species.

An alternative approach is “phylogenetic footprinting” (e.g., refs. 11 and 12) which identifies significantly conserved segments in multiple alignments of orthologous intergenic sequences. One of the drawbacks of this approach is that it assumes that only conserved sequence is functional, which is often not a safe assumption (13,14).

6.1. Specifying the Phylogeny: Using Preconstructed Trees

The phylogenetic tree relating all species from which the input sequences derive has to be specified to the program. Generally this is done via the command-line option `-L` described next but, in simpler cases, options `-H` or `-G` can also be used (*see* the program manual).

With the `-L` option the tree is specified in the standard Newick format but with so-called proximities in the place of distances. The proximity of a species to its ancestor is defined as the probability q that any given base that is not under selection has not mutated in the time separating the ancestor and the descendant. Note that proximities only apply to bases that are aligned with orthologous bases in other species, i.e., the bases in later insertions and deletions are not considered phylogenetically related to other sequences in the input. Note also that the proximity is multiplicative: if a species s_1 has proximity q_1 to ancestor a_1 and a_1 has proximity q_2 to an earlier ancestor a_2 , the proximity of s_1 to a_2 is q_1q_2 .

The easiest way in practice for determining the phylogenetic tree of the input species is to obtain a species tree for the species in question externally. For almost all sequenced organisms, approximate phylogenetic trees constructed using different algorithms on different sets of orthologous sequence are generally available. Once such a tree is obtained externally, the main task will be to replace the “branch lengths” with proximities. Here, the simple rule

is that the probability q that no mutations have taken place along a branch is related to the expected number of mutations m along the branch by: $q = e^{-m}$. Thus, if the external tree specifies the number of synonymous substitutions per site s then the proximity may be reasonably approximated as $q = e^{-s}$.

6.2. Specifying the Phylogeny: Calculating a Tree

If a preconstructed phylogenetic tree is not available then the user will have to construct one. If this seem daunting, it is important to keep a few things in mind. First, the behavior of PhyloGibbs is highly robust against changes in the proximities that are specified. Therefore, one only needs to get the tree very roughly correct to get close to optimal performance: one significant digit in the proximities will generally suffice. In some cases, a reasonable guess might already give performance that is hardly distinguishable from the performance with the true tree (*see Note 1*).

To reconstruct the phylogenetic tree one would generally start by estimating proximities between all pairs of species. There are several way of doing this, including:

1. Given a set of orthologous protein-coding genes between the two species one may use standard methods to align them and estimate the synonymous substitution rate in aligned regions. Synonymous mutations may not be entirely free of selection but are sometimes the closest available. As mentioned already, the proximity q is related to the number of synonymous substitutions s per synonymous site by $q = e^{-s}$.
2. Alternatively, one can estimate mutation rates in aligned regions of noncoding DNA. Some of these aligned regions will be binding sites, but if we assume that the binding sites are few in number compared to the background, the result will be a good approximation.

Having estimated pairwise proximities, one needs to combine these into a phylogenetic tree. In general one can use a UPGMA-like algorithm and get the best-fit proximities to intermediate ancestors. However, if there are only two or three species the tree can be estimated more directly:

1. If there are only two species, we can place a common ancestor halfway between them: if the proximity of the two sequences is q , their proximity from the common ancestor is $\sqrt{q}$. (This assumes both sequences evolved at the same rate.)
2. If there are three species, we can use a star topology in which all three species are directly connected to the common ancestor without any internal nodes. Let the three species be numbered 1, 2, and 3, and the common ancestor be A . Knowing their pairwise proximities q_{12} , q_{23} , q_{13} , we can calculate each species' proximity to the ancestor using

$$q_{12} = q_{1A}q_{2A}, \quad q_{23} = q_{2A}q_{3A}, \quad q_{13} = q_{1A}q_{3A}$$

which have the unique solution

$$q_{1A} = \sqrt{\frac{q_{12}q_{13}}{q_{23}}}, \quad q_{2A} = \sqrt{\frac{q_{12}q_{23}}{q_{13}}}, \quad q_{3A} = \sqrt{\frac{q_{13}q_{23}}{q_{12}}}$$

Even with more than three species the overall tree can often be well-approximated by a star phylogeny. In these cases, the phylogenetic tree consists of one ancestor and many leaves, each labeled by their proximity to the root, and the proximities can be set to approximately match the proximities between all pairs of species.

Finally, it should be noted that the time that the PhyloGibbs program takes to parse the tree rises sharply (exponentially) with the number of internal nodes. Therefore, it will improve running time, and generally not greatly hurt the results, to keep the number of internal nodes small, by removing internal nodes that are reasonably proximate (e.g., proximities greater than 0.8 or 0.9 to their parents).

6.3. Preparing the Input Multi-Fasta File

Prealigned input must be provided to PhyloGibbs in multi-fasta format (described below, *see* [Fig.1](#)). Any alignment program that provides output in multi-fasta format may be used. Because noncoding DNA tends to be “piecewise conserved” with orthologous blocks interspersed with unrelated sequence, we recommend a program such as Dialign ([15](#)) that does not use insertion/deletion penalties but builds up global alignments from local alignments of conserved blocks. Recently, one of us has written another program, Sigma ([16](#)), that uses a similar approach but is aimed specifically at noncoding DNA.

In the multi-fasta format, each sequence line looks like a line in a standard fasta-format file, except that gaps (represented by dashes, “–”) are inserted to ensure that only bases that are orthologous, i.e., derive from a common ancestor base, appear in the same vertical column. The variant of multi-fasta used by Dialign and Sigma adopts the additional convention that only uppercase bases are considered to be aligned; a column may also contain lowercase bases, which are assumed to be phylogenetically independent of the other sequences in the input. PhyloGibbs assumes this convention too; it can work either with such mixed-case multi-fasta ([Fig. 1](#)) or uppercase-only multi-fasta output, created by programs such as ClustalW ([17](#)) and Mlagan ([18](#)), where each column contains only orthologous bases.

To correctly take the phylogenetic relationships between the species into account PhyloGibbs must in general be able to identify which sets of sequences

```
>Scer_YHR124W NDT80 SGDID:S0001166 5' untranslated region, Chr VIII:3555
63..356562, 1000 bp length=999
atcgcacttttgatatctacttttttttatttcgaaaacaaggcacaacaatgaa-----TCTAT
CGCCCTGTGAGATTTTCAATCTCAAGTTTGTGTAATAGATAGCGTTATATATAGAactataaaggctccttg
aatatacatagtggttttcattcctattactgtTATATGTGACTTTACATTGTTACTTCCGCGGCTATTTGACGT
TTTctgctTCAGGTGCGGCTTGGAGGGCAAAGTGTGAGAAAATCGGCCAGGCGGTATGACACAAAAGAGTAG
AAAACGAGATCTCAAAATATCTCGAGGCTGTCTCTATAC-AACCGCCAGCTCTCTGACAAAAGCTCCAGAA
CGGTTGTCTTTTGTGTTTCGAAAAGCCAAGGTCCCTTATAATTGCCCTCCATTTTGTGTACctattTAAGCAA
AAAAATTGAAAGTTTACTAACCTTTTCATTAAGAGAAAATAACAATATTATAAAAA-GCGCTTAAA
>Sbay_Contig514.9 NDT80 YHR124W 5' untranslated region, Contig c514 1530
5 - 16304 (revcom), 1000 bp length=999
aacgcgcactttgttcacacgcttttctggtttgtgttcttccctttatTTAAATAAAACCCAATTTTCTCTAT
TGCCCTGCGGGACAACCGGCTCTAGTCTGTGTAATAGATAACATTATATTATAGAATGATAGAACTATCG
ATATGCATAGTGCCTTTTATCGCTGTGCGAGATATATCTGGCCTCACCTTATCACTTCCGCGGCTATTTGACGT
TTTTTGT-TCAAGCGCGGCTTGGACGGCAAAGTGTGAGAAAATCGGCCAGGCTGTATGACACAAAAGGGCaa
aaagagatctcaaaagccctctcgagacaagtcctcttctgAACCGCCGAGCTCTCTGCAAACTCTATTGGA
CAATCATCTTTTGTGTTGAAGAGGTAACCTCCGTTACAGTTGTCCCCCATTTTGTGTCActTAC-TAAAGTA
GAAATTAAGAGTTTAATAAACATTCAATAAGAGGGGAAAACGGTAATATAAAAAGCACTTAAA
>Smik_Contig2829.6 NDT80 YHR124W 5' untranslated region, Contig c2829 69
67 - 7966, 1000 bp length=999
aaatcatggtttgtgtttacgcttctctcttttttttctta-----TTAAACAAGGTACAAAGCACTCTAT
TGCTCGGTGAGATTATCAATCTCAACTTTGTGTAATAGATACCGTTATATTATAGAGTTATAGAAATCCGTTT
GATGTACATAGTGCCTTCAATTGCTGTTGCAGTATATGTAGTTTACATTGTAACCTCCGCGGCTATTTGACGT
TTTTTTG-TCCAGTGCGGCTTAAAGGCCAAAGTGTGAGAAAATCGGCCATGCCGAATGACACAAAAGAGTGG
CAACCGATATCTCAAGGTTCTCGAGGTCTATTCTATTCTG-AACCGCCAGCTTTCTAAAAAGGTCACCAA
CAGTTGTCTTTTGTGTTGACGAGCCAAGGTCTGTATAACTGTCCGCCGTTTGTGTGAC-TAT-TAAAAACA
AAAAATAAAAGCTTAGTATACCTTTCATTAAGAGGacaacagtaataattaaaa--GCGCTTAAAA
```

Fig. 1. An example of aligned sequence in multifasta format, which may be fed to PhyloGibbs: the promoter of NDT80, a gene from *Saccharomyces cerevisiae*, and orthologous regions from *Saccharomyces bayanus* and *Saccharomyces mikatae*.

are multiple-aligned and which species each sequence derives from. Therefore, the headers (sequence names) of the fasta file may need to satisfy certain requirements:

1. If a phylogenetic tree in Newick format is specified (*see* **Section 6.4** and **Fig. 2**) then each species will be denoted by a label in this tree. In this case, each header must contain the label corresponding to the species from which its sequence derives, and no other labels. (Conversely, each label must appear in the header of all sequences originating from the associated species, and in no other headers.)
2. If a “star phylogeny” is being considered—all species are descended directly from a common ancestor with varying divergence times—and, for every regulatory region being studied, sequence for every orthologous species exists and is provided in the same order, there is a simple alternative, the **-H** option. Also, in a star phylogeny where all species are equally diverged from the ancestor, an even simpler option exists, **-G**. In these special cases, no labels are required in the sequence headers. For brevity, we do not discuss these here; they are described in the program manual.
3. When the input set contains multiple sets of aligned orthologous sequences (for example, alignments of sets of orthologous promoters from multiple genes in a regulon) then all these alignments need to be supplied in one single file. In addition, the first header in each multiple alignment should start with a double marker (“>>”)

in a fasta header, instead of the usual single ">," to indicate that a new group of aligned sequences starts at this place. If such double marks were not included PhyloGibbs would interpret the file as a single giant alignment, with nonsensical results. Note that only the first header in each aligned group should carry this ">>" indicator.

6.4. Command-Line Options

In addition to the commands discussed earlier, the following commands are used when running PhyloGibbs on phylogenetically related sequences.

1. **-D**: the parameter **-D** tells PhyloGibbs if the input is to be interpreted as aligned and how to treat the alignment. **-D 0** tells PhyloGibbs to treat the sequences as not aligned; dashes in input sequences are ignored and the case of the letters is ignored as well. Conversely, with **-D 1** or **-D 2** PhyloGibbs will treat each group of sequences starting with a ">>" as multiply aligned. The difference between **-D 1** and **-D 2** is in the way regions with gaps are treated. Because of gaps in the alignment, some sequence segments of length w in one species will be aligned to segments of a different length in other species. Because we assume all binding sites have the same length, such segments are inconsistent with a site occurring at that position. When the option **-D 1** is used PhyloGibbs will split the alignment in such places into subalignments containing subsets of the species that are all mutually consistent. When **-D 2** is used PhyloGibbs will simply not allow binding sites to occur at positions inconsistent with the gap pattern of the alignment. PhyloGibbs automatically decides whether to assume that sequences are aligned or not, depending on whether or not it detects dashes ("—") in the input sequences so that in most cases it will do the right thing without specifying the **-D** option.
2. **-L treestring**: the **-L** option is used to specify the phylogenetic tree and takes as an argument a representation of the phylogenetic tree in the standard Newick format: each node is represented as a closed pair of brackets "()," and each leaf by a string labeling the species concerned. (As previously noted, it is required that these strings occur as substrings in the headers of the input fasta file, so as to uniquely identify the species of each sequence.) Each closed pair of brackets contains a comma-separated lists of either leaves (strings) or further nodes (closed pairs of brackets), followed by a colon ":" and the proximity to that node's or leaf's parent. For example, the tree in [Fig. 2](#) would be represented by the string "`((chimp:0.85,human:0.9):0.6,(rat:0.9,mouse:0.9):0.7)`" (this string may have to be placed in quotes, to protect the brackets from the Unix shell).
3. **-G**, **-H**: these commands may be used as shortcuts for specifying the phylogeny in simple cases, but because incautious use can lead to errors, we do not discuss them here; they are documented in the manual.

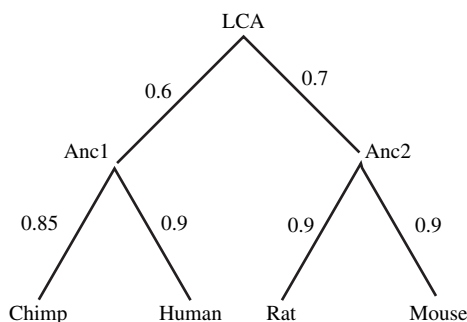


Fig. 2. A sample phylogenetic tree for mammals (numbers are only approximate). Because proximities are multiplicative, the proximity of chimp to LCA (the last common ancestor, or the root of the tree), for example, is $0.85 \times 0.6 = 0.51$. This tree would be represented on the PhyloGibbs commandline by `-L "(chimp:0.85,human:0.9):0.6,(rat:0.9,mouse:0.9):0.7)"` (assuming these substrings appear uniquely in the sequence headers to identify the species).

In addition, the following options have slightly different meanings when running on aligned sequence:

1. `-I` and `-y`: the numbers given in these options refer to the number of sites. It is important to note that, when running with aligned sequences, a binding site that occurs in region where n species are aligned does not count as n binding sites, but counts only as a single "extended" binding site. So if one runs on the multiple alignment of n orthologous upstream regions and one expects s sites in each species, then one should use `-I s` or `-y s`.

7. Other Parameters

PhyloGibbs' running can be controlled by a large number of other options, most of which may never be needed. However, by using some of them, PhyloGibbs can be stretched to perform a number of tasks that go beyond motif-finding. Here, we describe some of these uses.

1. `-c`: allowing an arbitrary number of windows and colors. The option `-c` is used to turn on so called "color moves," either by `-c n`, which specifies n color moves per cycle, or by `-c -1` where PhyloGibbs automatically chooses the number of color moves per cycle. When color moves are turned on, PhyloGibbs is no longer restricted to a certain number of sites or maximal number of colors, but can freely vary them. When color moves are used the values specified in `-I` or `-y` and `-z` are interpreted as the *a priori* expected number of sites and colors. So when using

color moves the user should specify a “best guess” at the total number of sites and motifs represented in the input.

2. **-r**: running on a single strand only. By default PhyloGibbs will search for sites both on the given sequence as well as on its reverse-complement. This is appropriate for DNA sequences where TFs can bind on both sides of the double-stranded helix. However, when one searches for sites on a single strand only, one should use the option **-r**. When **-r** is specified PhyloGibbs will only search for sites on the given strand.
3. **-B**: blocking positions in the input. In some cases, one may want to exclude certain segments of the input sequences. For example, in higher eukaryotes one may want to exclude repetitive sequences from consideration. In other cases, one may already know the existence of certain binding sites in the sequences and one may want to exclude these positions so as to ensure that PhyloGibbs does not rediscover these already known sites. When using **-B filename** PhyloGibbs will block all sequence segments in the file “filename” from consideration. Alternatively, one can replace each letter in the segments that one wants to exclude with the letter X. PhyloGibbs will also exclude all positions where an X occurs in the sequence from consideration.
4. **-A**: testing the significance of a configuration. When **-A filename** is used PhyloGibbs will read a binding site configuration from the file “filename” and, instead of performing an anneal, will take this configuration as the reference configuration. Statistics on this reference configuration will then be collected during a sampling run as usual and printed to the tracked output file.
5. **-M**: prespecifying a set of motifs. Probably the most useful additional feature is the ability of PhyloGibbs to include specific prior information about the motifs that are likely to occur in the input sequences. When using **-M filename** PhyloGibbs will read a set of weight matrices (WMs) from the file “filename.” The WM format used is the same format as used for each motif in the PhyloGibbs output files and also matches the WM format used by TRANSFAC (19). PhyloGibbs interprets each input WM as a set of binding sites from a common WM. When scoring binding site configurations PhyloGibbs will now also evaluate the probability that one or more of the groups of binding sites in the configuration derived from the WMs given in the input file, and score the configuration accordingly. This option thus allows one to either specify initial guesses for the motifs, or to search for binding sites for one or more well defined WMs. For details, see the manual.

8. Output and Track File Format

The output file (Fig. 3) shows the binding site configuration that was obtained at the end of the annealing, i.e., the reference configuration. (PhyloGibbs 1.0 had a slightly different output format from this description, but the information content was the same.) For reference, it first lists all the command-line options that were used, then the names of the input sequences and their lengths, the

```

Command-line arguments: -D 1 -L (cer:0.8,par:0.8,mik:0.58,kud:0.5,bay:0.45) -T 0.5 -m 15
-N 3 -F backgroundfile -y 45 -z 3 -f GATl_regions.fna -o GATl_regions.out -t GATl_regio
ns.track
Seq 0: >Scer_YDL237W      Length 178
Seq 1: Spar_2881      Length 178
Seq 2: Sbay_Contig5 Length 203
Seq 3: Smik_Contig2 Length 107
Seq 4: Skud_Contig1 Length 170
Seq 5: >Scer_YEL062W      Length 999
Seq 6: Spar_5973      Length 999
Seq 7: Sbay_Contig6 Length 999
Seq 8: Smik_Contig2 Length 844
Seq 9: Skud_Contig1 Length 999
... further sequences snipped ...
GSL Random number seed: 545
No. of moves: colour 0, single window 5085, shift 678, total 5763
Log-posterior probability of the reference state: 418.552977
===== Reference state obtained through annealing. =====
Motif 1.
Number of windows = 22 Top window score= 1.11454e-10

ttaatttCACGCTAGTTAAGTCaaagcag -- [fwd] seq 15 Scer_YJR011 pos 914 score 1.115e-10
ttaatttCACGCTAATTAAGTCaaagtag |-[fwd] seq 16 Spar_12348 pos 913
tcaattttTACATCAATCAAGCTtaagcag |-[fwd] seq 17 Sbay_Contig pos 924
tcagttttCTCATTAATTAAGCTtaagcata |-[fwd] seq 18 Smik_Contig pos 920
tcaattttCACATCAACTAAATCaagctag '-[fwd] seq 19 Skud_Contig pos 914
gtataggCTTGTTATTTCAGAAATgtgatcc -- [rev] seq 15 Scer_YJR011 pos 277 score 5.725e-07
gtataggCTTATTATTTCAGAAATgtgatcc |-[rev] seq 16 Spar_12348 pos 300
gtataggTTTATTATTTTAAATgttggtcc |-[rev] seq 17 Sbay_Contig pos 351
gtatgggCTTATTGTTTAAATatgatct |-[rev] seq 18 Smik_Contig pos 311
gtacgggCTTGTTATTTCAGAAATgttggtcc '-[rev] seq 19 Skud_Contig pos 364
tatatacCTTATTCATCAACACttttctcc -- [rev] seq 25 Spar_14750 pos 262 score 5.652e-06
tatatacTTTATTCATCAACACttttctct |-[rev] seq 26 Sbay_Contig pos 261
tatatacCTTATTCATCAACACttttctcc '-[rev] seq 27 Skud_Contig pos 261

... further sites snipped ...

----- Weight matrix for this motif (absolute base counts)-----
//
NA Motif_1
PO      A      C      G      T      cons      inf
01      0.00     26.15     4.67     10.28      C      0.73
02     11.78     2.98     4.14     21.48      T      0.38
03      4.17     12.03     0.00     24.04      T      0.70
04     25.98     0.00     15.07     0.98      A      0.91
05      0.00     13.45     0.98     26.37      T      0.94
06      3.86     18.04     1.00     17.65      Y      0.50
07     27.15     5.58     2.65     4.18      A      0.62
08     20.56     0.00     3.93     16.89      W      0.65
09      0.00     23.89     0.00     18.72      C      1.01
10      4.74     28.06     0.00     9.93      C      0.76
11     34.80     3.13     1.97     0.97      A      1.18
12     28.65     0.00     5.59     7.48      A      0.79
13     16.41     7.80     14.70     1.98      R      0.27
14     15.26     13.91     0.00     13.51      H      0.42
15      0.00     11.26     0.00     29.11      T      1.15
//
=====
... rest of output snipped ...

```

Fig. 3. Sample output file. Output sequences that occur in conserved blocks are grouped together.

random number seed that was used, and the total number of moves of each type that was performed during the run. It next shows the logarithm of the posterior probability of the reference configuration. After this the actual reference configuration is shown. For each motif the number of windows, i.e., sites, in the motif is shown plus the score of the highest scoring site in the motif. The score reported for each window in the output file is the difference between the log-posterior probability of the configuration that is obtained when the reference state is perturbed by removing the window in question and the log-posterior probability of the reference state. The smaller the score the “better:” a very small score indicates that the posterior probability of the configuration drops a lot when the window is removed. For each site the sequence (or sequences when the site occurs in a region where multiple species are aligned) is shown in capital letters together with flanking sequences in small letters. Then the strand is shown on which the site occurs, i.e., either [fwd] or [rev], followed by the position in the input sequence that corresponds to the start of the site. Finally, the score of the site is shown. Note that for sites that span multiple species the input sequence and position is shown for every individual site in the aligned segment but there is only one score for the entire aligned segment. The motifs are ordered by the score of the highest scoring site in each motif (starting with the best motif). After the list of sites in the motif the WM corresponding to the motif is shown. For each position in the motif the number of times the letters A, C, G, and T occur at that position are shown, together with a consensus base, and the information score at that position (which runs from zero bits for a completely random distribution of bases to two bits for a position at which all sites have the same base). The start and end of the WM are indicated by a line containing only //.

The tracked output file ([Fig. 4](#)) is very similar in structure to the output file just described. It also starts by listing the command-line options and input sequences used. Instead of listing the posterior probability of the reference configuration, the file then shows the average of the logarithm of the posterior probabilities visited during the tracking phase. The file then shows posterior probabilities for different sites to belong to the motifs that occurred in the reference state as obtained through the sampling run. For each motif, all sites are shown ordered by their posterior probability, i.e., the fraction of the time they occurred in this motif during annealing. The format of each site is the same as in the output file with the only difference that the site score is replaced by the posterior probability of the site. By default only sites whose posterior probabilities are at least 0.05 are shown but this cut-off can be changed (*see* **Option -E**, described below). Finally, after each list of tracked sites the WM

```

Command-line arguments: -D 1 -L (cer:0.8,par:0.8,mik:0.58,kud:0.5,bay:0.45) -T 0.5 -m 15
-N 3 -F backgroundfile -y 45 -z 3 -f GAT1_regions.fna -o GAT1_regions.out -t GAT1_regio
ns.track
Seq 0: >Scer_YDL237W      Length 178
Seq 1: Spar_2881      Length 178
Seq 2: Sbay_Contig5      Length 20
Seq 3: Smik_Contig2      Length 107
Seq 4: Skud_Contig1      Length 170
Seq 5: >Scer_YEL062W      Length 999
Seq 6: Spar_5973      Length 999
Seq 7: Sbay_Contig6      Length 999
Seq 8: Smik_Contig2      Length 844
Seq 9: Skud_Contig1      Length 999

... further sequences snipped ...

GSL random number seed 545
No. of moves: colour 0, single window 10035, shift 1338, total 11373
Average log-posterior probability of sampled configurations: 380.876670

== Posterior probabilities obtained through tracking the reference state. ==

Tracking stats motif 2
-----
ctgttttAAAATCCTTATCTTGtctcctt -- [fwd] seq 0 Scer_YDL237 pos 49 prob 1.00
ctgttttGGGTTCCTTATCTTGgctctttt - [fwd] seq 1 Spar_2881 pos 48
acattttAGATTCTTATCTTtctccctt | - [fwd] seq 2 Sbay_Contig pos 69
ttgcttcACGTGCTTATCTCGcttcttt '- [fwd] seq 4 Skud_Contig pos 37
tgggtgaATAATTTACCTAGCTgttggat -- [rev] seq 15 Scer_YJR011 pos 621 prob 0.71
gtcgcaagTAATTTACCTAGTtttcgggtt - [rev] seq 16 Spar_12348 pos 644
gtcgtagGAGACTTATCTAGTCatcgatt - [rev] seq 17 Sbay_Contig pos 671
ttcttgaATGATTTACTTGACTatccttt - [rev] seq 18 Smik_Contig pos 643
catgcgtAAGGTTTATCTAGTTattgatt '- [rev] seq 19 Skud_Contig pos 672
tacttatCTTAACCTTATCGTCTtctctcg -- [fwd] seq 5 Scer_YEL062 pos 333 prob 0.68
tacttatCTTAACCTTATCGTtttctctcg | - [fwd] seq 6 Spar_5973 pos 344
tacttatCTCGACCTTATCGCTtctctcg - [fwd] seq 7 Sbay_Contig pos 289
aacctatCATAACCTTATCGTAttcctcg - [fwd] seq 8 Smik_Contig pos 200
tagctatCGCAACCTTATCGTtttctctcg '- [fwd] seq 9 Skud_Contig pos 372
ttttatgTCGACCTTATCTAAgtaaata -- [fwd] seq 24 Scer_YLR023 pos 276 prob 0.54

... further sites snipped ...

----- Weight matrix for this motif (absolute base counts)-----
//NA Motif_2
PO      A      C      G      T      cons      inf
01      7.96    4.91    3.15    6.77      N      0.08
02      5.58    4.67    6.08    7.87      N      0.03
03      5.24    2.96    5.95    9.64      D      0.12
04      6.46    5.49    2.83    9.04      H      0.11
05      6.52    6.03    2.32    7.10      H      0.10
06      0.65    4.34    2.19    14.80     T      0.67
07      0.18    13.74    0.45    7.43      C      0.88
08      6.66    0.40    0.92    13.01     T      0.74
09      0.00    1.10    0.00    20.15     T      1.71
10      15.07    4.80    0.42    0.91      A      0.86
11      1.23    0.11    0.33    19.21     T      1.51
12      2.51    15.44    2.07    2.32      C      0.62
13      0.82    0.37    7.56    12.97     T      0.75
14      5.59    5.06    3.14    8.71      N      0.09
15      7.18    3.93    2.41    9.47      H      0.17
//
=====

... further output snipped ...

```

Fig. 4. Sample track file.

for this motif is shown. This WM differs from the one in the output file in that in its construction each site is weighed by its posterior probability.

Options affecting the output files:

1. **-E**: by default only sites with posterior probability 0.05 or larger are shown in the tracked output file. By using **-E x** all sites with posterior probability **x** or larger are shown.
2. **-R**: by default the site positions in the output files correspond to the start of the site as counted from the start of the sequence (with zero corresponding to the first position). By using **-R** one can instead report site positions counting from the end of each sequence. That is, with position **-1** corresponding to the last base in the sequence. This can be useful when running on upstream region, i.e., a site at position **-n** then corresponds to a site starting **n** bases upstream of transcription start.

9. Web Interface

To allow users to run the code without having to install it locally, and to provide a more user-friendly interface we have developed a web interface through which PhyloGibbs can be run. PhyloGibbs online can be found at <http://www.phylogibbs.unibas.ch>.

Three different user interfaces are provided at the website. First, for expert users there is an “expert” interface. Here, the user can upload the input files, i.e., the input sequence file and possibly files with background sequences or an input WM file, and essentially just type the command-line options as one would do on a Unix command-line.

The “advanced” interface provides a user with a page of fields that set the main command-line options, such as the total number of sites and motifs, and so on. For most fields default values are given so that the user only needs to specify some of the most essential options.

Finally, the “step by step” interface aims to provide lay users with a simple step-by-step process for providing PhyloGibbs with input sequences and parameters. Here, the user is asked a number of questions about that input sequences and the prior information about them. If the user wants to run on phylogenetically related sequences but does not yet have multiple alignments the interface will also use Dialign (15) to align the sequences.

Once the PhyloGibbs job has finished the results can be viewed on the website. In contrast to the command-line version the web interface also provides graphical representations of the results. These allow the user to see graphically where the sites for the different factors fall within the input sequences and it also provides WM logos for discovered motifs. Links to the raw text output file and tracked output file are also provided.

9.1. Downloads

The PhyloGibbs source code as well as executables for a number of different architectures can be downloaded from either <http://www.imsc.res.in/~rsidd/phylogibbs/> or <http://www.biozentrum.unibas.ch/~nimwegen/cgi-bin/phylogibbs.cgi>.

The source code is freely available under the GNU General Public License. By registering the user will stay informed of bug fixes and new releases of the code.

10. SwissRegulon Site Annotation Database

We have started producing genome-wide annotations of regulatory sites using PhyloGibbs on multiple related genomes in combination with data from ChIP-on-chip experiments, microarray expression data, and collections of known binding sites from the literature. These binding site annotations are made available at the website <http://www.swissregulon.unibas.ch>.

Currently, annotations for *S. cerevisiae* produced using a number of different methods are available and annotations for *Escherichia coli* and *Bacillus subtilis* are in preparation. The database graphically depicts the predicted binding sites on the genome together with what factor is binding each site, the strand on which the site occurs, the posterior probability of the predicted site, and a host of other information. It allows users to see at a glance that factors are predicted to regulate each gene and which sets of genes are predicted to be regulated by each factor.

11. Notes

1. These statements assume color-changing moves are not used. When color-changing moves are used the total number of sites PhyloGibbs predicts becomes more sensitive to the phylogeny parameters. That is, if the user specifies that the species are more distant than they actually are, then PhyloGibbs will overestimate the amount of functional conservation and likely overpredict the number of sites.

References

- 1 Siddharthan, R., Siggia, E. D., and van Nimwegen, E. Phylogibbs: (2005) A gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput. Biol.* **1**, e67.
- 2 Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262**, 208–214.
- 3 Bailey, T. L. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **2**, 28–36.

- 4 Rajewsky, N., Vergassola, M., Gaul, U., and Siggia, E. D. (2002) Computational detection of genomic cis-regulatory modules applied to body patterning in the early *Drosophila* embryo. *BMC Bioinformatics* **3**, 30.
- 5 Sinha, S., van Nimwegen, E., and Siggia, E. D. (2003) A probabilistic method to detect regulatory modules. *Bioinformatics* **19**, 292–301.
- 6 Sinha, S., Schroeder, M. D., Unnerstall, U., Gaul, U., and Siggia, E. D. (2004) Cross-species comparison significantly improves genome-wide prediction of cis-regulatory modules in *Drosophila*. *BMC Bioinformatics* **5**, 129.
- 7 Berman, B. P., Pfeiffer, B. D., Lavery, T. R., et al. (2004) Computational identification of developmental enhancers: conservation and function of transcription factor binding-site clusters in *Drosophila melanogaster* and *Drosophila pseudoobscura*. *Genome Biol.* **5**, R61.
- 8 Berman, B. P., Barret, Y. N., Pfeiffer, D., et al. (2002) Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the *Drosophila* genome. *Proc. Natl. Acad. Sci. USA* **99**, 757–762.
- 9 Johansson, O., Alkema, W., Wasserman, W. W., and Lagergren, J. (2003) Identification of functional clusters of transcription factor binding motifs in genome sequences: the MSCAN algorithm. *Bioinformatics* **19**, 169–176.
- 10 Quackenbush, J. (2001) Computational analysis of microarray data. *Nat. Rev. Genet.* **2**, 418–427.
- 11 Blanchette, M. and Tompa, M. (2003) FootPrinter: a program designed for phylogenetic footprinting. *Nucleic Acids Res.* **31**, 3840–3842.
- 12 Blanchette, M. and Tompa, M. (2002) Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res.* **12**, 739–748.
- 13 Dermitzakis, E. T., Bergman, C. M., and Clark, A. G. (2003) Tracing the evolutionary history of *drosophila* regulatory regions with models that identify transcription factor binding sites. *Mol. Biol. Evol.* **20**, 703–714.
- 14 Emberly, E., Rajewsky, N., and Siggia, E. D. (2003) Conservation of regulatory elements between two species of *drosophila*. *BMC Bioinformatics* **4**, 57.
- 15 Morgenstern, B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics* **15**, 211–218.
- 16 Siddharthan, R. (2006) Sigma: multiple alignment of weakly-conserved non-coding dna sequences. *BMC Bioinformatics* **7**, 143.
- 17 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 18 Brudno, M., Do, C. B., Cooper, G. M., et al. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 19 Matys, V., Fricke, E., Geffers, R., et al. (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.* **31**, 374–378.

Using the Gibbs Motif Sampler for Phylogenetic Footprinting

William Thompson, Sean Conlan, Lee Ann McCue,
and Charles E. Lawrence

Summary

The Gibbs Motif Sampler (Gibbs) is a software package used to predict conserved elements in biopolymer sequences. Although the software can be used to locate conserved motifs in protein sequences, its most common use is the prediction of transcription factor binding sites (TFBSs) in promoters upstream of gene sequences. We will describe approaches that use Gibbs to locate TFBSs in a collection of orthologous nucleotide sequences, i.e., phylogenetic footprinting. To illustrate this technique, we present examples that use Gibbs to detect binding sites for the transcription factor LexA in orthologous sequence data from representative species belonging to two different proteobacterial divisions.

Key Words: Gibbs sampling; phylogenetic footprinting; transcription regulation.

1. Introduction

The identification of transcription factor binding sites (TFBSs) is an important part of defining the regulatory network of an organism. TFBSs exert significant control over gene transcription through the binding of their cognate transcription factors (TFs). The promoters of genes regulated by a common TF, either within a species or across species, can be analyzed to predict potential regulatory sites. Experimental methods exist to detect coexpressed genes within a species, and can be used to identify a subset of sequences for analysis. Phylogenetic footprinting, on the other hand, does not require experimental data, and

instead uses orthologous sequence data from multiple species (genomes). This method relies on the assumption that among closely related species, orthologous genes are likely to be regulated by a common TF; thus, the TFBSs will be conserved, whereas the nonregulatory portion of the promoter sequences will be less conserved. Phylogenetic footprinting has been successfully applied to both prokaryotic species (*1,2–5*) and eukaryotic species (*6–10*) to locate putative TFBSs.

Although there have been a number of computational tools developed for detecting conserved sequence elements (*7,11–19*), the Gibbs Motif Sampler is one of the most mature. Gibbs sampling was first applied in the field of bioinformatics in 1993 (*20*), and there have been numerous enhancements to, and applications of, the original Gibbs sampling technique since its introduction (*1,3,4,8,21*). Here, the method is described only conceptually; mathematical details of the procedure as applied to the search for motifs in biopolymer sequences are described elsewhere (*22,23*). A key feature of the Gibbs sampling approach is the use of motif models to capture sequence patterns shared by multiple sequences. A motif is a common model of a collection of binding sites. The motif is usually represented as a position–weight matrix in which each row represents a position in the conserved pattern and each column represents a nucleotide. The elements of the matrix are typically probabilities or counts of nucleotides. The advantage of using motif models of multiple sequence alignments lies in the increase in the signal-to-noise ratio that results from averaging over the individual sequences. However, this advantage decreases when the sequences are correlated (possess a high degree of identity), as is the case for orthologous data from evolutionarily close species, in which insufficient time has elapsed for mutations to accumulate in nonsite regions. The Gibbs Motif Sampler also uses a principled Bayesian method for determining the number and positions of the TFBSs, and provides a number of options designed to model biological features of TFBSs, such as palindromic and directly repeating models. Although Gibbs employs a relatively mature algorithm, it is important to bear in mind that experimental validation is ultimately necessary to substantiate the biological function of predicted motifs.

2. Materials

2.1. Obtaining the Gibbs Software

There are two ways to access Gibbs for analyzing data:

1. Web interface: Gibbs may be accessed on the internet at <http://bayesweb.wadsworth.org/gibbs/gibbs.html> and at <http://www.bioinfo.rpi.edu/applications/bayesian/gibbs/>

[gibbs.html](#). These websites also provide access to a user manual, various data files, and online descriptions of phylogenetic footprinting and the analysis of prokaryotic coexpression data. Furthermore, each option and data input field is linked to context-sensitive help describing its use.

2. Command line program: Gibbs is available as a command-line tool for the Solaris, Solaris.x86, and Linux operating systems. In addition, a version is available for Microsoft Windows, which requires Cygwin (<http://cygwin.com/>). Gibbs is distributed free of charge to the academic research community for noncommercial, nonprofit internal research use. A license request may be obtained at <http://bayesweb.wadsworth.org/GIBBS-SAMPLER-ACADEMIC.htm>. A license for commercial use may be obtained at <http://bayesweb.wadsworth.org/GIBBS-SAMPLER-COMMERCIAL.htm>. A version for parallel computer systems supporting the Message Passing Interface is also available from the authors for computing clusters using the Linux, Solaris, or Solaris.x86 operating systems. This chapter will focus on the use of the command-line tool, but most of the comments will also apply to the web-based version.

2.2. Requirements

1. Hardware: a Sun workstation running Solaris or an x86-based PC running Solaris.x86, Linux, or Microsoft Windows are required.
2. Software: the current command-line version of Gibbs is 3.00 or later. The package as distributed contains the appropriate Gibbs binaries, the binaries to perform a Wilcoxon signed-rank test of statistical significance, and the unifiedcpp binary, which produces the background composition files for use as input to Gibbs. The Cygwin package is needed for the Microsoft Windows version. X-Windows support is required for the Solaris version. Memory and processor speed requirements vary depending on the size of the data sets being analyzed. The examples in this article can be run quite comfortably on a laptop computer with 512 Mb of memory and an Intel Pentium processor.

2.3. Installation

Gibbs is distributed as a gzipped tar file. To install Gibbs, copy the `Gibbs.tar.gz` file to a directory. In the next example, we use the directory `Gibbs`, but any appropriate directory name may be used. At a command-line, type (user entry is in **bold**):

```
~/ Gibbs > gunzip Gibbs.tar.gz  
~/ Gibbs > tar -xvf Gibbs.tar
```

The “tar” command will create the appropriate subdirectories for the various versions. The top level directory will contain a README file and a sample data

file, crp.dat. The subdirectories will contain the binary files for the appropriate operating systems.

2.4. Data Files

Gibbs accepts sequence data in FASTA format. The Gibbs distribution contains a sample data file, crp.dat. This file, along with all data files used in this article, is available for download at <http://bayesweb.wadsworth.org/gibbs/module>.

3. Methods

Gibbs has a large number of options and modes of operation (24). In this chapter we will concentrate on the subset of these options that are used for predicting TFBSs in cross-species data. The following examples illustrate the principles involved in computational detection of TFBS in prokaryotes, however, the principles are similar for the analysis of eukaryotic sequences. We will point out differences in the analysis of eukaryotic sequence data in the Note section (see **Note 2**).

3.1. Sequence Data

In phylogenetic footprinting, the sequences being analyzed represent orthologous promoter sequences, and as such are assumed to contain binding sites for a common TF. We typically chose a target species of interest and additional species based on their phylogenetic relationship to the target species (4,5). Orthologous promoters are identified by first identifying orthologous gene sequences in these species. If the species' genomes have been annotated, a pairwise reciprocal BLAST methodology such as INPARANOID (25) will efficiently predict orthologous genes from their protein translations. Alternatively, using the protein sequences from only the target species, a TBLASTN (26) procedure may be used to identify orthologous genes in the raw genome sequences of the related species (3). Briefly, the protein sequences from the target species' genome (which must be annotated) are used as TBLASTN queries to a nucleotide database composed of the target species' genome and the genomes of the related species. A number of heuristics are then applied to ensure that BLAST hits are likely true orthologs, rather than paralogs or domain-level matches:

1. The expectation value must be less than 10^{-20} .
2. The expectation value must be less than the second best hit in the target species.
3. The BLAST hit must start within 20 amino acids of the target query sequence.

Once orthologous genes are identified by either of the previously listed approaches, the orthologous upstream intergenic sequences are extracted. If orthologs have been identified by INPARANIOD from fully annotated genomes, the annotations are used to define intergenic sequence boundaries. Using the TBLASTN procedure, the boundaries of an intergenic region can be delineated using the BLAST results for the bounding orthologous gene, if the BLAST results indicate that the gene order between species is conserved. If conserved gene order is not detected, a maximum of m bases ($= 500$ bp by default), upstream of the orthologous gene is used. With this approach, the target species data will include only intergenic sequence, whereas the sequences for the additional species may or may not be trimmed to exclude upstream coding region. For eukaryotes, similar techniques may be used; however, the intergenic regions extracted are typically much larger. In these cases, we limit the sequence length, m , to a maximum of 3000–5000 bp. Parsing INPARANOID or BLAST output and generating orthologous intergenic sequence files on a whole-genome scale is generally accomplished using *ad hoc* Perl scripts.

The two examples shown next describe the use of Gibbs to predict TFBSs upstream of orthologous *lexA* genes. The file, *gamma_lexA.fa*, contains the *lexA* promoter sequences for *Escherichia coli* K12 and six additional γ -proteobacterial species: *Salmonella enterica* serovar Typhi CT18, *Yersinia pestis* CO-92, *Vibrio cholerae* El Tor, *Haemophilus influenzae* Rd, *Pseudomonas aeruginosa* PAO1, and *Shewanella oneidensis* (4). The file, *alpha_lexA.fa*, contains the *Rhodopseudomonas palustris lexA* promoter sequence plus seven additional orthologous sequences from α -proteobacterial species: *Bradyrhizobium japonicum*, *Brucella suis* 1330, *Caulobacter crescentus*, *Rhodobacter sphaeroides*, *Rhodospirillum rubrum*, *Novosphingobium aromaticivorans*, and *Hyphomonas neptunium* (5). Both these example files were generated using the TBLASTN procedure previously described, using the LexA protein sequences from *E. coli* and *R. palustris*, respectively, as the queries.

3.2. Background Composition

Gibbs sampling is a Markov-chain Monte Carlo technique and site positions are sampled into the motif model based upon a probability model. The probability that a particular position in the sequence is sampled as a site is calculated as the ratio of the probability of the site under the motif model to the probability under a background model. The background model describes the sequence in the absence of TFBS. Using a background model that is homogeneous in

composition is problematic, because noncoding sequence is often heterogeneous in composition, particularly in eukaryotes. Local variations in nucleotide composition have been shown to adversely affect sequence alignments (27,28). To address this, the Bayesian segmentation algorithm (29) is used to produce a position-specific background model. The algorithm calculates the probabilities of observing each of the four bases at each position in a sequence, based on the sequence's compositional heterogeneity and on the uncertainty in the heterogeneity. These probabilities are then used as position-specific background models in the Gibbs sampling procedure. To create a background composition file, use the `unifiedcpp` program, supplied with the Gibbs package:

```
~/Gibbs> unifiedcpp gamma_lexA.fa
```

`Unifiedcpp` will produce a number of files in the directory containing the FASTA file. These files will begin with the same name as the FASTA formatted sequence file, but will have an additional extension. The file ending with `info-det`, in this example `gamma_lexA.fa_info-det`, contains the background probabilities. This file will be used as an input parameter to Gibbs. The background composition file does not have to be recreated unless the number of sequences or length of the sequences in the FASTA file changes (recreating the composition file is not required after masking subsequences in the FASTA file, *see Note 3*).

3.3. Command-Line Parameters

We have attempted to build a number of options into the Gibbs software to model the biological properties of TF–TFBS interactions, because the more accurately a computational analysis reflects the underlying biology of the system under study, the more likely it is to be successful. Therefore, it is important to consider the biological context of the data being analyzed, and use this as a guide when choosing Gibbs parameters. For example, many prokaryotic TFs bind as symmetric, homodimeric protein complexes that have corresponding palindromic DNA binding motifs (e.g., *E. coli* Crp) (19), whereas others bind as directly repeating multimers and therefore have direct-repeat binding patterns (e.g., *E. coli* PhoB) (30). These binding site structures (palindromes and direct repeats) can be specified using options in Gibbs. In addition, the exact width of the binding pattern may not be known. Gibbs can infer this width using a modified version of the fragmentation algorithm (22). The command-line parameters control how the program makes inferences about these features.

Gibbs sampling is a stochastic process and thus also provides several parameters to control program runtime. Gibbs uses the posterior probability of the alignment, the MAP (maximum *a posteriori* probability) (22), as a measure

of the quality of the alignment. The MAP is calculated as the log of the alignment probability minus the log of the empty or “null” alignment. Thus, it is a measure of the extent to which a particular alignment is better than background. Gibbs sampling begins by generating a random alignment, then iterating through the sequences, examining each in turn and sampling motif sites. During each sampling iteration, the program updates the background and motif counts by adding the positions occurring in each motif element to motif counts and deleting them from the background counts. Sampling proceeds until a plateau is reached in the MAP values (i.e., the MAP does not improve for some specified number of iterations). For details of the sampling process, *see* refs. 1,8, and 22. Because it is possible for the Gibbs sampling procedure to become stuck in local optima, the program performs a number of random restarts called seeds. Command-line parameters control the number of seeds, the maximum number of iterations, and the plateau period. The user can obtain a complete list of options as follows (*see* Fig. 1):

```
~/Gibbs> Gibbs -h
```

The user can also obtain more extensive information about these options by consulting the website (<http://bayesweb.wadsworth.org/gibbs/bernoulli.html>).

3.4. Informed Priors

Using informed priors with Gibbs is another way in which information on the biological system under study can be leveraged; two commonly used types of prior information are described here. In some cases, there may be experimental evidence about which TF regulates a given gene, and it may be possible to obtain position weight matrix (PWM) for the specific TF. For example, the PRODORIC (31), TRANSFAC (32), and JASPAR (33) databases contain PWMs for a number of common TFs. These matrices can be used to provide clues about the expected binding patterns of TFs, but do not control the inference of sites and motifs by Gibbs. Also, an investigator may be interested in a TF that is a member of a family of TFs, but for which there is no available PWM. In these circumstances, an informed prior from the family can enhance the sensitivity of this algorithm (34). Details of using PWMs as informed priors can be found at <http://bayesweb.wadsworth.org/gibbs/prior.html>. In the absence of prior motif information, Gibbs will calculate uninformed priors based on the average background composition. In our examples, we use uninformed prior models.

In addition, it is known that many TFs bind to more than one site in a promoter and often bind cooperatively, particularly in eukaryotes. Thus, we

Gibbs 2.10.001 Dec 31 2005

```

USAGE (site sampler) :: Gibbs file lengths {flags}
USAGE (motif sampler):: Gibbs file lengths expect {flags}
USAGE (recursive sampler):: Gibbs file lengths expect -E max_sites {flags}

lengths = <int>[,<int>] : width of motif to be found.
expect = <int>[,<int>] : expect number of motif elements
max_sites = <int> : max. sites/seq

possible flags:

-A                               init sample from prior
-B <bkgnd_filename>             Background Composition Model
-C <cutoff_value>               cutoff for near optimal sampler
-D <seqs[,aligned_seqs]>        Homologous sequences, seqs default = 2, aligned_seqs def. =
all seqs
-E max_sites                    Set max sites/seq, use recursive sampler
-F                               Do not use fragmentation
-G                               Group Sampler
-H <weight_filename>           Sequence Weight File
-I <mnum, beg, end>*            direct repeat model between beg and end
-J                               Fragment sites in center only
-K <map>                        Alt. method of sampling sites/seq, min map to start (optional)
(-E option only)
-L                               Motif sample before recursive (-E only)
-M <mnum, width>*              maximum widths for fragmentation
-N <scan_filename>             output data for Scan
-O <prout_filename>            output informative priors
-P <prior_filename>            file of informative priors
-Q <sample_filename>           save sample counts in file
-R <mnum, beg, end>*            palindromic model between beg and end
-S <num_Seeds>                 number of seeds to try
-U <spacing_filename>          Spacing Model
-V <no. of seqs>                Verify Mode
-W <pseudo_wt>                 pseudosite weight (between 0 and 1)
-X <min><,<max>,<step>,<t>>>    Parallel Tempering (MPI version only)
-Y                               Calculate default pseudocount weight
-Z                               Don't write progress info
-a <mnum, beg, end, pal>*       Concentrate between beg and end
-b                               Sample from background
-c <mnum, beg, end>*            collapse alphabet between beg and end
-d <mnum, min, max>*            Allow width to vary
-f <cutoff_factor>              cutoff factor for recursive sampler
-frag                           alternate fragmentation sampling
-g                               Sample along length of site
-h                               this message
-hierarchical_model <iter>     hierarchical model for sites/seq
-i <iteration_num>              number of iterations to try
-j <period>                     Frag/Shift period
-k <iteration_num>              number of iterations to sample after plateau
-l <control_filename>          Wilcoxon Signed Rank test
-m                               Do not maximize after near optimal sampling
-n                               Use nucleic acid alphabet
-nopt                           Don't print Near Optimal output
-nopt_disp <value>             Min probability for Nearopt display
-o <out_filename>              file where results will be written
-p <plateau_per>               number of periods a maximum value hasn't changed
-q                               Sample width counts
-r                               turn off reverse complements with DNA
-s <seedval>                   random number generator seed
-sample_model                  sample probability model from dirichlet
-t                               Display sites used in near optimal sampling
-u                               Display output from suboptimal sampler
-v <overlap_value>             % to allow overlap at ends of sequence
-w <pseudo_cnt_wt>             pseudocount weight
-wilcox                        Wilcoxon sequences included in fasta file.
-x                               Do not remove low complexity regions
-y                               Don't print frequency solution

```

Fig. 1. Gibbs command-line options produced by the command “Gibbs -h.”

allow Gibbs to predict more than one site per sequence. In our example sequences, we have reasonably high confidence that among closely related species, an orthologous gene will be regulated in a similar way. Therefore, we allow Gibbs to search for a motif with up to two sites per input sequence and provide prior information that there is relatively low prior probability of finding no sites ($p = 0.05$), and equal probabilities of finding one or two sites ($p = 0.475$) in each sequence. To do this, we create a text file (called a prior file) containing the following statements:

```
>BLOCKS
0.05 0.475 0.475
>
```

In our examples, this text file is named *lexA.pr* and was provided as an argument at the Gibbs command-line. Further details on prior information may be found at the URL previously mentioned. As discussed next, the Gibbs sampler also permits searching for motifs of multiple TF simultaneously.

3.5. Running Gibbs

This first example searches for the LexA binding site in the *lexA* promoters from seven γ -proteobacterial species. Enter the following command to analyze the *lexA* sequences in *gamma_lexA.fa*.

```
~/Gibbs> Gibbs gamma_lexA.fa 16 7 -n -E 2 -r -R 1,1,8 -M 1,24  
-B gamma_lexA.fa_info-det -P lexA.pr -o gamma_lexA.out
```

The parameters are:

1. *gamma_lexA.fa*: the file name of the FASTA format sequence file.
2. 16: the initial motif width. This is the number of conserved positions in the TFBS. Fragmentation will allow the site to expand.
3. 7: the initial estimate of the total number of sites. For cross-species studies, we typically set this estimate equal to one site per input sequence.
4. -n: indicates that the FASTA format file contains nucleotide data and a DNA alphabet should be used.
5. -E 2: the maximum number of sites allowed in each sequence. In this case, each sequence may contain zero, one, or two sites.
6. -r: disables searching of the reverse complement of the input sequence data.
7. -R1, 1, 8: specifies that motif number one is a palindrome. The first eight conserved positions will be combined with the reverse complements of the last eight positions to form the palindromic model. When using a palindromic model, searching for sites on the reverse complement of the input sequence data should be disabled

using `-r` option. Not doing so when using a palindromic model generally results in an artificially asymmetrical motif model. When searching for non-palindromic models, it is often useful to search for TFBSs in both the forward and reverse complement directions. Even with non-palindromic models, searching both strands of the DNA may not be desired, however, if the input sequence data have been oriented with respect to the gene, as in the direct-repeat example below.

8. `-M 1,24`: set the maximum fragmentation width for motif model number one. This setting allows Gibbs to search for TFBS with widths between 16 (previously specified) and 24 bp.
9. `-B gamma_lexA.fa_info-det`: specifies the background composition file created by `unifiedcpp`.
10. `-P lexA.pr`: specifies the prior file containing the probabilities of the number of sites per sequence.
11. `-o gamma_lexA.out`: specifies the file that will be created by Gibbs to contain the results. If this option is omitted, the Gibbs output is written to the terminal window (STDOUT).

Except for the FASTA file name, the motif width, and “-n,” which indicates that the FASTA file contains nucleotide data, the other parameters are strictly optional. However, note that if an estimate of the total number of sites and the maximum number of sites allowed per sequence are not specified, Gibbs will default to a site sampling mode in which exactly one site will be identified in each sequence, a behavior that is generally not desirable for phylogenetic footprinting.

There are a number of technical command-line options that control behavior of the program. These are set to default values that we have found useful in our research, but may require adjustment depending on the type and amount of sequence data to be studied. These parameters include:

1. `-s`: `-s` followed by an integer indicates the number of random restarts (seeds) of the Gibbs sampling process. The default is 20 seeds.
2. `-p`: `-p` followed by an integer sets the plateau period. For example, `-p 50` sets a plateau period of 50 iterations; if there is no improvement in the alignment after 50 sampling iterations, then Gibbs will initiate the next random restart. The default value for this parameter is 50 iterations.
3. `-i`: `-i` followed by an integer sets the maximum number of iterations for each seed; the default is 500. When a large number of sequences are being examined, the default number of seeds and plateau period may be too low to effectively search the alignment space. Thus, as the amount of sequence data increases, it is a good idea to increase the number of seeds and the plateau period. If the plateau period is increased, the maximum number of iterations may be reached before a plateau in the MAP is reached. To avoid this problem, the maximum number of iterations may be increased by including the option, `-i`, followed by an integer. A rule of thumb that has proven useful is that the maximum number of iterations should be at least

five times the plateau period. For example, if the option, -p 200 were included, including the option, -i 1000 would be appropriate.

As it executes, Gibbs will write some temporary progress information to the user's console. Running a file such as the one in the example (seven sequences, each less than 500 bp) should take from 1 to 2 s to slightly less than 1 min on most computers. The amount of time required to process a set of sequences will depend on the length of the individual sequences, the number of sequences, the number of seeds, and the plateau period.

3.6. Interpreting the Output

Figure 2 shows a portion of the output file that results from running the example previously described. The complete output file from this example can be downloaded from <http://bayesweb.wadsworth.org/gibbs/module/> and a complete description of the Gibbs output format can be found in (1,24), as well as on the Gibbs website. At each sampling iteration, the posterior probability (the MAP) of the sampled alignment is calculated and saved. When all the seeds (random starts) have been run, Gibbs saves the alignment with the maximum MAP. Then, for this alignment solution, the algorithm continues to sample sites for a fixed number of iterations, to explore variations in the models. Gibbs counts the frequency at which each site is sampled to assess its reproducibility; these frequencies are sampling estimates of the probability that each site belongs to the common motif model. Sites selected with a frequency greater than 50 % are also displayed in the output. These sites represent a frequency solution. The frequency solution is often the same as the optimal MAP solution, but because it represents the reproducible samples from among the possible alignments, it is less likely to contain false-positives than the MAP solution.

Figure 2A shows the motif model detected in the γ -proteobacterial *lexA* example. This model consists of a table with a column for each of the nucleotides and a row for each conserved position in the motif. The numbers in the table indicate the frequency of occurrence of each nucleotide at each position within the motif. The last column is an information parameter, expressed in bits, that indicates how much the column adds to the model. Using the information value, it is possible to determine which positions are most conserved. **Figure 2B** shows the TFBSs predicted in the frequency solution in each species. The first column identifies the sequence number, followed by the motif element number for that sequence. The next column indicates where the motif element starts within the sequence. The fourth column contains the predicted TFBS in upper case; flanking sequences are shown in lower case. The motif element is followed by the ending site position within the sequence. Column six shows the

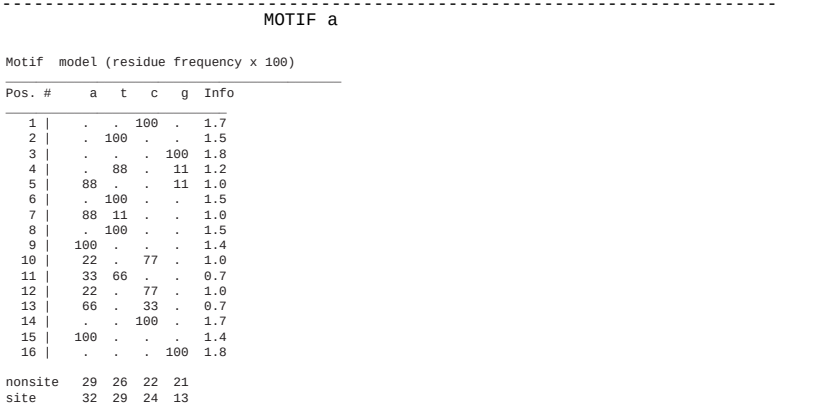


Figure 2A

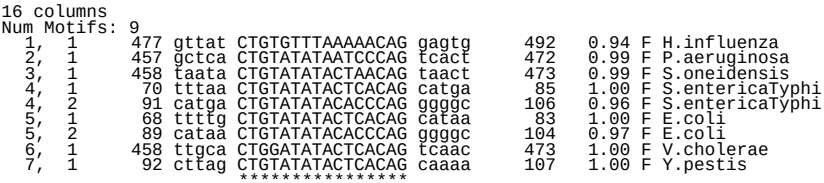
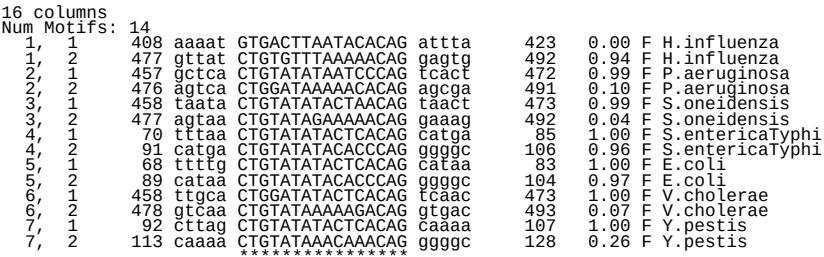


Figure 2B



Column 1 : Sequence Number, Site Number
Column 2 : Left End Location
Column 4 : Motif Element
Column 5 : Right End Location
Column 6 : Probability of Element
Column 7 : Forward Motif (F) or Reverse Complement (R)
Column 8 : Sequence Description from Fast A input

Figure 2C

Fig. 2. The motif model for γ -proteobacterial *lexA* orthologous promoters. (A) The probability model for the frequency solution shown in B. The optimal MAP model in C contains 14 sites from seven sequences. Several of these sites have low sampling probabilities and may represent false-positives.

sampling estimates of the probability of each of the sites in the solution. Finally, a portion of the FASTA header is displayed. The row of asterisks below the binding site indicates the fragmentation pattern. In this case the motif model was not fragmented. **Figure 2C** shows the MAP solution. LexA is the TF that regulates the DNA damage response (SOS response) and is widely distributed across prokaryotes. In *E. coli*, more than 30 genes are likely regulated by LexA as part of the SOS response regulon (35), and LexA regulates its own expression via auto-regulatory binding sites upstream of the *lexA* gene. In *E. coli* and related γ -proteobacteria, LexA is known to bind to a palindromic DNA sequence with the consensus, CTG-N₁₀-CAG (31). This is the pattern of the TFBSs shown in **Fig. 2**. The probability values in column six of **Fig. 2C** indicate that most of the sites have high reproducibility (0.9 or greater in this case) and thus we have high confidence that they match the predicted motif. However, several sites have low sampling probabilities and should be regarded with suspicion. These sites are not included in the frequency solution.

3.7. LexA Binding Motif in α -Proteobacteria

Because it is usually not known what types of TFBSs to expect in a given set of orthologous promoter sequences during phylogenetic footprinting, we typically do not use a motif PWM in our prior information, and we run Gibbs on the data several times using different parameters, for example, specifying a palindromic, non-palindromic, or direct repeat model. Gibbs motif predictions are then collected and considered potentially significant if they have a positive MAP. For example, running Gibbs with the α -proteobacterial promoter sequences in the file `alpha_lexA.fa` and using the same parameters as previously described, with the exception of varying the option for model type, produced the distinct motif alignments shown in **Fig. 3**. Specifying a palindromic model resulted in a motif consisting of only three sites (no sites were included for five of the species, see **Fig. 3A**). Running the data with either a non-palindromic (**Fig. 3B**) or direct repeat model (**Fig. 3C**) produced similar alignments: both alignments are composed of sites from seven of the species, however the sites are shifted slightly and fragmented differently in the two alignments. To specify a direct repeat, a `-I` parameter was used in place of the `-R` parameter previously used for a palindromic model, and a non-palindromic model was specified by not using either of these parameters. In addition, the `-r` option can be omitted when using direct repeat or non-palindromic models, thereby allowing Gibbs to search for sites on the reverse complement DNA strand, as well as the forward strand. However, we included the `-r` option in this example, because the sequences in `alpha_lexA.fa` are oriented 5' to 3', with respect to

```
~/Gibbs> Gibbs alpha_lexA.fa 16 8 -n -E 2 -R 1,1,8 -r -S 20 -p 50 -M 1,24 -B alpha_lexA.fa_info-det -P lexA.pr -o
alpha_lexA.pal.out
1, 1 54 acccc GAACAGATAGTGCCGTC atgat 72 1.00 F R.palustris
2, 1 79 atacc GAACATATAGTGCCGTC atgat 97 1.00 F B.japonicum
3, 1 191 gactg GAACATATAGTGTCGTC tgggt 209 1.00 F B.suis
*****
MAP = 10.47
```

Figure 3A

```
~/Gibbs> Gibbs alpha_lexA.fa 16 8 -n -E 2 -r -S 20 -p 50 -M 1,24 -B alpha_lexA.fa_info-det -P lexA.pr -o
alpha_lexA.non.out
1, 1 37 aagag GTTGCGGAACACCCCGAACA gatag 58 1.00 F R.palustris
2, 1 62 aagca GTTGCGGAACACATATCGAACA tatag 83 1.00 F B.japonicum
3, 1 174 aaacc ATTGCAGAACCAAGACTGGAACA tatag 195 0.93 F B.suis
4, 1 436 acctt GCAGCGGAACACCGGAGAACA ttcga 457 1.00 F C.crescentus
5, 1 381 acctg AACGGGGAACCAAGTAGAACT tccag 402 1.00 F R.rubrum
6, 1 65 acaca AAGGGGAACGCGGCAGAACA ggcgg 86 1.00 F R.sphaeroides
7, 1 445 acccg AACAGGGAACGCTTGTAGAACA aaagc 466 1.00 F H.neptunium
*****
MAP = 19.62
```

Figure 3B

```
~/Gibbs> Gibbs alpha_lexA.fa 16 8 -n -E 2 -r -S 20 -p 50 -M 1,24 -I 1,1,8 -B alpha_lexA.fa_info-det -P lexA.pr -o
alpha_lexA.dir.out
1, 1 40 aggtt GCGGAACACACCCCGAACA gatag 58 1.00 F R.palustris
2, 1 65 cagtt GCGGAACACATATCGAACA tatag 83 1.00 F B.japonicum
3, 1 177 ccatt GCAGAACCAAGACTGGAACA tatag 195 1.00 F B.suis
4, 1 439 ttcga GCGGAACACCAAGAGAACA ttcga 457 1.00 F C.crescentus
5, 1 384 tgaac GGGGAACCAAGTAGAACT tccag 402 0.96 F R.rubrum
6, 1 68 caaag AGGGAACGCGGCAGAACA ggcgg 86 0.98 F R.sphaeroides
7, 1 448 cgaac AGGGAACGCTTGTAGAACA aaagc 466 1.00 F H.neptunium
*****
MAP = 26.06
```

Figure 3C

Fig. 3. Predicted sites in α -proteobacterial *lexA* orthologous promoters. The three panels show the command-line, aligned sites, and the frequency solution found when using a palindromic motif model (A), a non-palindromic motif model (B), and a direct repeat motif model (C).

the promoter of interest; thus searching both DNA strands is not required. The motifs in Fig. 3B,C exhibit a direct repeat pattern, GAAC-N₇-GAAC, that has been shown to bind the *R. palustris* LexA protein in gel-shift experiments (36). Furthermore, the alignment in Fig. 3C had the highest MAP, suggesting that of the three models tested, a direct repeat model is the most appropriate model for the α -proteobacterial LexA binding sites. In each of these cases, the MAP and frequency solutions were identical.

Note that, regardless of the Gibbs parameters used, none of the motif patterns in Fig. 3 match the LexA motif in Fig. 2 from the γ -proteobacteria. The observation that the LexA protein is conserved across bacterial lineages, could lead one to naïvely assume that the LexA binding motif would be conserved across a broad spectrum of species. However, this is not the case, because the conserved region of LexA is involved in catalysis rather than DNA binding. Specifically, the LexA repressor remains bound to its cognate binding sites

until DNA damage is detected, at which point the LexA protein undergoes auto-proteolytic cleavage and dissociates from the DNA. This relieves LexA repression, allowing the expression of genes involved in DNA repair (SOS response). Unlike the conserved catalytic domain, the DNA binding domain of LexA is not well conserved and recognizes different *cis*-elements in different lineages of bacteria (37,38). **Figure 4** shows sequence logos (39) illustrating the differences between the motif models for the γ - and α -proteobacteria,

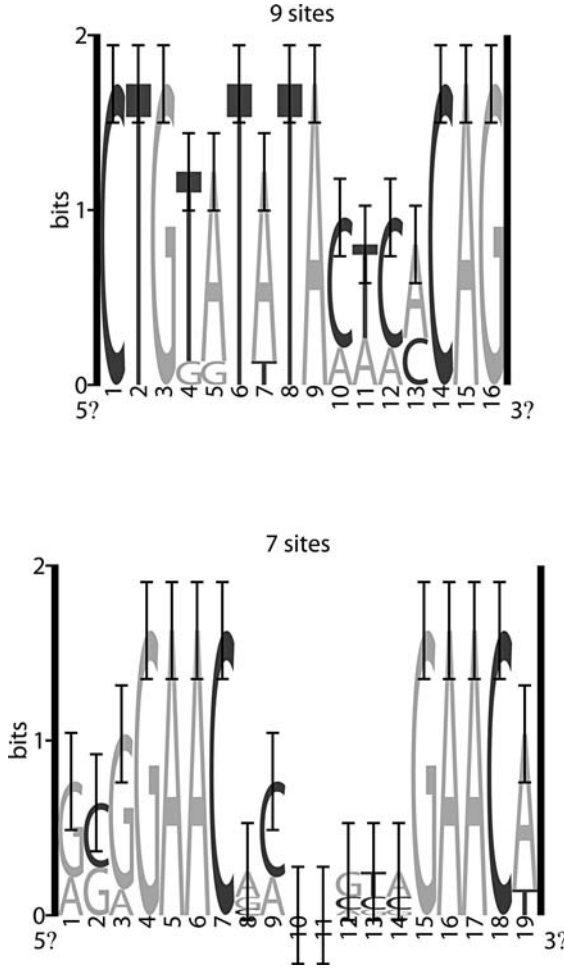


Fig. 4. Sequence logos (39) for the γ -proteobacterial LexA sites (top) compared to the α -proteobacterial LexA sites (lower).

respectively. This example highlights a difficulty that may be encountered when applying comparative genomic methods to distantly related species. It also demonstrates that using precomputed PWMs from a given species or set of species to search for *cis*-regulatory elements in another genome can be misleading. Therefore, although the use of matrices as prior information can markedly improve the sensitivity of the Gibbs sampling algorithms, it should be used with caution. Specifically, they should be used only when there is evidence to expect a similar binding pattern, such as when DNA binding motifs are known to be conserved. The *de novo* (using uninformed priors) approach described here avoids this potential problem, albeit with a potential loss in sensitivity, because it makes no assumptions about the pattern of the regulatory motif. Also, because the Gibbs Sampler simultaneously determines both the motif model and its distribution across the input sequences, it is not required that all input sequences contain a given *cis*-regulatory element. This is important, because not all species in a comparative study may employ a given regulatory pathway.

4. Notes

2. As with any software package, there are a number of possible problems that may arise in its use. The most obvious, and easily corrected, are errors caused by incorrect syntax or use of parameters. Gibbs provides reasonable defaults for most parameters. It also has extensive error checking and generates messages when parameter values are in error. Errors of this type are usually easy to repair. However, Gibbs is a large program with a number of interacting pieces. Furthermore, it is a research program under continual modification. As such, it may contain bugs. The program is able to detect certain conditions that might lead to a fatal error. When such a condition is encountered an error message beginning with the following phrase is output:

FATAL INTERNAL ERROR ::

If the user receives such a message, please contact the authors for assistance. Other problems are more subtle. They are typically caused by naïve choices of the data sequences or inappropriate parameter choices.

3. We have tried to show how one method of analysis, phylogenetic footprinting, can be used to predict TFBS, particularly in bacterial species. Analysis of regulatory regions from higher organisms (e.g., vertebrates) has additional complications relative to studies in prokaryotes. A number of repetitive elements have been identified in eukaryotic genomes, and it is useful to mask these repeat sequences using RepeatMasker (40) prior to attempting motif prediction. Furthermore, although prokaryotic promoter regions are typically short (less than 500 bp), in

eukaryotes the upstream regulatory region may be quite a bit larger, extending to thousands or tens of thousands of bases. Also, eukaryotic TFBS tend to be somewhat shorter (10–14 bases on average) than those in bacteria, and are less likely to be palindromic. The sites also typically occur in clusters, called regulatory modules, that consist of sites for multiple different TFs. Therefore, it is useful to search eukaryotic promoter data for multiple different motif models simultaneously (8). This can be done with Gibbs as follows:

```
~/Gibbs> Gibbs file.fa 10,10,10 5,5,5 -n -E 5 -P file.pr -B  
file.fa_info-det -p 200 -i 1000 -o file.out
```

In this example command-line, three motifs are specified, each with an initial motif width of 10, and an initial estimate of number of sites of 5. We will allow from zero to five sites in each sequence. Even though eukaryotic TFBSs tend to occur in modules, in a collection of relatively long orthologous eukaryotic promoter regions, it is quite possible that the short regulatory motifs (TFBSs), will be lost in the background noise. It has been shown, however, that in a collection of orthologous human and rodent promoters from skeletal muscle genes, 98 % of experimentally defined sequence-specific binding sites of skeletal-muscle-specific TFs are found in the 19 % of human sequences that are most conserved in the orthologous rodent sequences (9). In practice, this means that by concentrating on the highly conserved regions of the sequence, we can increase the signal to noise ratio in the data and more effectively predict TFBS. This can be accomplished by prealigning the promoter sequences from orthologous genes and sampling simultaneously from the aligned groups of sequences (8). Despite this, it is difficult to analyze promoter regions from single genes successfully using mammalian cross-species data, because of the relatively recent divergence of mammals. As additional mammalian sequences are added to an orthologous data set, the correlations in the sequence data tend to limit the marginal contribution from the sequences of multiple, related mammalian species (41). This version of the Gibbs sampler does not account for the phylogenetic relatedness of input species, but a version that takes into account the phylogenetic relationships among the input species is now under development, and will be included in updated releases.

3. Although this chapter has focused on phylogenetic footprinting, Gibbs is also commonly used to predict regulatory elements in coexpression data for a single species. Advances in technology have made feasible, in select organisms, the detection of coregulated genes, as well as protein–DNA interactions under a variety of physiological conditions. Traditional microarray techniques and promoter fusions (42) are commonly used to identify coregulated genes. Protein–DNA interactions are readily detected using gel-mobility shift assays or ChIP-chip assays (43). In addition, cutting edge technologies like ChIP-PET (44) have potential for whole-genome analysis of *cis*-regulatory elements in eukaryotes. These types of experiments are used to identify genes whose coexpression is

owing, at least in part, to regulation by a common TF, and Gibbs sampling can be used to identify the putative binding motif for the factor. We expect high-throughput experimental data, especially expression array data, to contain more noise, in the form of false-positives and secondary effects, than other more targeted techniques; therefore, additional optimization may be needed when using Gibbs to search for conserved sequences. For example, when analyzing microarray gene expression data, which provides evidence of coexpression, but not direct evidence of a commonality in the regulatory mechanisms, the prior probabilities on the number of sites per sequence can be changed to reflect the fact that only a subset of coexpressed genes may be regulated by a particular TF or module of TFs, by setting a higher expectation of zero sites. This setting reflects our expectation that more of the input promoter sequences are not regulated by a single regulatory mechanism, and thus are more likely to contain zero sites from the predicted module. Examples of the analysis of coexpression data are provided online at http://bayesweb.wadsworth.org/web_help_text.CE.apr232007.html.

The examples presented here, and in the supplementary online resources, demonstrate how the Gibbs Motif Sampler can be used to detect conserved regulatory motifs. The convergence of high-throughput sequencing initiatives and parallel experimental methods is providing the data necessary to delineate the complex regulatory networks of many organisms. The Gibbs Motif Sampler is under continuous development and future versions will allow us to better utilize this wealth of genomic and experimental data.

References

- 1 Thompson, W., Rouchka, E. C., and Lawrence, C. E. (2003) Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Res.* **31**, 3580–3585.
- 2 Yan, B., Methe, B. A., Lovley, D. R., and Krushkal, J. (2004) Computational prediction of conserved operons and phylogenetic footprinting of transcription regulatory elements in the metal-reducing bacterial family Geobacteraceae. *J. Theor. Biol.* **230**, 133–144.
- 3 McCue, L., Thompson, W., Carmack, C., et al. (2001) Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Res.* **29**, 774–782.
- 4 McCue, L. A., Thompson, W., Carmack, C. S., and Lawrence, C. E. (2002) Factors influencing the identification of transcription factor binding sites by cross-species comparison. *Genome Res.* **12**, 1523–1532.
- 5 Conlan, S., Lawrence, C., and McCue, L. A. (2005) *Rhodopseudomonas palustris* regulons detected by cross-species analysis of alphaproteobacterial genomes. *Appl. Environ. Microbiol.* **71**, 7442–7452.

- 6 Sandelin, A., Wasserman, W. W., and Lenhard, B. (2004) ConSite: web-based prediction of regulatory elements using cross-species comparison. *Nucleic Acids Res.* **32**, W249–W252.
- 7 Sinha, S., Schroeder, M., Unnerstall, U., Gaul, U., and Siggia, E. (2004) Cross-species comparison significantly improves genome-wide prediction of cis-regulatory modules in *Drosophila*. *BMC Bioinformatics* **5**, 129.
- 8 Thompson, W., Palumbo, M. J., Wasserman, W. W., Liu, J. S., and Lawrence, C. E. (2004) Decoding human regulatory circuits. *Genome Res.* **14**, 1967–1974.
- 9 Wasserman, W. W., Palumbo, M., Thompson, W., Fickett, J. W., and Lawrence, C. E. (2000) Human-mouse genome comparisons to locate regulatory sites. *Nat. Genet.* **26**, 225–228.
- 10 Lee, T. K. and Friedman, J. M. (2005) Analysis of NF1 transcriptional regulatory elements. *Am. J. Med. Genet. A*. **137A**, 130–135.
- 11 Bailey, T. L. and Elkan, C. (1995) Unsupervised learning of multiple motifs in biopolymers using EM. *Machine Learning* **21**, 51–80.
- 12 Thijs, G., Marchal, K., Lescot, M., et al. (2002) A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J. Comput. Biol.* **9**, 447–464.
- 13 Blanchette, M., Schwikowski, B., and Tompa, M. (2002) Algorithms for phylogenetic footprinting. *J. Comput. Biol.* **9**, 211–223.
- 14 Buhler, J. and Tompa, M. (2002) Finding motifs using random projections. *J. Comput. Biol.* **9**, 225–242.
- 15 Marsan, L. and Sagot, M. F. (2000) Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *J. Comput. Biol.* **7**, 345–362.
- 16 Sinha, S. and Tompa, M. (2002) Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic. Acids Res.* **30**, 5549–5560.
- 17 Stormo, G. D. (1990) Consensus patterns in DNA. *Methods Enzymol.* **183**, 211–221.
- 18 Sinha, S., Blanchette, M., and Tompa, M. (2004) PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics* **5**, 170.
- 19 Lawrence, C. E., and Reilly, A. A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* **7**, 41–51.
- 20 Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., and Wootton, J. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262**, 208–214.
- 21 Neuwald, A., Liu, J., and Lawrence, C. (1995) Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Science* **4**, 1618–1632.
- 22 Liu, J., Neuwald, A., and Lawrence, C. (1995) Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Amer. Stat. Assoc.* **90**, 1156–1170.

- 23 Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1999) Markovian structures in biological sequence alignments. *J. Amer. Stat. Assoc.* **94**, 1–15.
- 24 Thompson, W., McCue, L. A., and Lawrence, C. E. (2005) Using the Gibbs Motif Sampler to find conserved domains in DNA and protein sequences. In *Current Protocols in Bioinformatics*, (Baxevanis, A. D., Davison, D. B., Page, R. D. M., Petsko, G. A., Stein, L. D., and Stormo, G. D., eds.), John Wiley & Sons, Inc., New York, NY, pp. 2.8.1–2.8.38.
- 25 Remm, M., Storm, C. E. V., and Sonnhammer, E. L. L. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons *J. Mol. Biol.* **314**, 1041–1052.
- 26 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25**, 3389–3402.
- 27 Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410.
- 28 Marchal, K., Thijs, G., Keersmaecker, S. D., Monsieurs, P., Moor, B. D., and Vanderleyden, J. (2003) Genome-specific higher-order background models to improve motif detection. *Trends Microbiol.* **11**, 61–66.
- 29 Liu, J. and Lawrence, C. (1999) Bayesian inference on biopolymer models. *Bioinformatics* **15**, 38–52.
- 30 Wanner, B. L. (1996) Phosphorus assimilation and control of the phosphate regulon. In *Escherichia coli and Salmonella: Cellular and Molecular Biology*, (Neidhardt, F. C., ed.), ASM Press, Washington, DC, pp. 1357–1381.
- 31 Munch, R., Hiller, K., Barg, H., et al. (2003) PRODORIC: prokaryotic database of gene regulation. *Nucleic Acids Res.* **31**, 266–269.
- 32 Matys, V., Fricke, E., Geffers, R., et al. (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.* **31**, 374–378.
- 33 Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W. W., and Lenhard, B. (2004) JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.* **32**, D91–D94.
- 34 Sandelin, A., and Wasserman, W. W. (2004) Constrained binding site diversity within families of transcription factors enhances pattern discovery bioinformatics. *J. Mol. Biol.* **338**, 207–215.
- 35 Fernandez De Henestrosa, A. R., Ogi, T., Aoyagi, S., et al. (2000) Identification of additional genes belonging to the LexA regulon in *Escherichia coli*. *Mol. Microbiol.* **35**, 1560–1572.
- 36 Dumay, V., Inui, M., and Yukawa, H. (1999) Molecular analysis of the *recA* gene and SOS box of the purple non-sulfur bacterium *Rhodospseudomonas palustris* no. 7. *Microbiology* **145**, 1275–1285.
- 37 Fernandez de Henestrosa, A. R., Cune, J., Mazon, G., Dubbels, B. L., Bazylnski, D. A., and Barbe, J. (2003) Characterization of a new LexA binding motif in the marine magnetotactic bacterium strain MC-1. *J. Bacteriol.* **185**, 4471–4482.

- 38 Mazon, G., Erill, I., Campoy, S., Cortes, P., Forano, E., and Barbe, J. (2004) Reconstruction of the evolutionary history of the LexA-binding sequence. *Microbiology* **150**, 3783–3795.
- 39 Schneider, T. D., and Stephens, R. M. (1990) Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.* **18**, 6097–6100.
- 40 Smit, A. F. A., Hubley, R., and Green, P. RepeatMasker Open-3.0. 1996–2004 <http://www.repeatmasker.org>.
- 41 Newberg, L. A., and Lawrence, C. E. (2004) Mammalian genomes ease location of human DNA functional segments but not their description. *Stat. Appl. Genet. Mol. Biol.* **3**, 1–12.
- 42 Florczyk, M. A., McCue, L. A., Purkayastha, A., Currenti, E., Wolin, M. J., and McDonough, K. A. (2003) A family of *acr*-coregulated *Mycobacterium tuberculosis* genes shares a common DNA motif and requires Rv3133c (*dosR* or *devR*) for expression. *Infect. Immun.* **71**, 5332–5343.
- 43 Buck, M. J., and Lieb, J. D. (2004) ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics* **83**, 349–360.
- 44 Wei, C. -L., Wu, Q., Vega, V. B., et al. (2006) A global map of p53 transcription-factor binding sites in the human genome. *Cell* **124**, 207–219.

Web-Based Identification of Evolutionary Conserved DNA *cis*-Regulatory Elements

Panayiotis V. Benos, David L. Corcoran, and Eleanor Feingold

Summary

Transcription regulation on a gene-by-gene basis is achieved through transcription factors, the DNA-binding proteins that recognize short DNA sequences in the proximity of the genes. Unlike other DNA-binding proteins, each transcription factor recognizes a number of sequences, usually variants of a preferred, “consensus” sequence. The degree of dissimilarity of a given target sequence from the consensus is indicative of the binding affinity of the transcription factor–DNA interaction. Because of the short size and the degeneracy of the patterns, it is frequently difficult for a computational algorithm to distinguish between the true sites and the background genomic “noise.” One way to overcome this problem of low signal-to-noise ratio is to use evolutionary information to detect signals that are conserved in two or more species. FOOTER is an algorithm that uses this phylogenetic footprinting concept and evaluates putative mammalian transcription factor binding sites in a quantitative way. The user is asked to upload the human and mouse promoter sequences and select the transcription factors to be analyzed. The results’ page presents an alignment of the two sequences (color-coded by degree of conservation) and information about the predicted sites and single-nucleotide polymorphisms found around the predicted sites. This chapter presents the main aspects of the underlying method and gives detailed instructions and tips on the use of this web-based tool.

Key Words: Bioinformatics; genetics; genomics; transcription; DNA regulatory regions.

1. Introduction

One of the major cell mechanisms for gene expression control is at the level of transcription. Transcription factor (TF) DNA-binding proteins recognize relatively short DNA “signals” (typically 6–12 bp) in the vicinity of transcription

start sites (TSS) and initiate (activators) or repress (repressors) gene transcription. Although each TF has a preferred set of transcription factor binding sites (TFBS), this set is usually degenerate. **Figure 1A** shows an example of 30 aligned rodent binding sites of a TF. Once such a set of sites has been identified (usually with some biochemical method like DNA footprinting, SELEX, ChIP, and so on) then the information of their alignment can be organized and subsequently used to search for more TFBS in the promoters of other genes. The most widely used method for information encoding of an alignment of TFBS is the position-specific scoring matrix (PSSM). A PSSM model is a $4 \times L$ weight matrix (L is the length of the target site of the particular TF) in which each column reflects the observed frequencies of each of the four bases in the particular target position (for a review, *see* **ref. 1**). **Figure 1B** shows the count matrix of the alignment and **Fig. 1C** the corresponding log-transformed matrix. For practical reasons, this is the most commonly used form of PSSM models, sometimes also corrected for the background frequencies. The log-frequency values have been shown to correspond to the binding energies for some TFs (**2–4**). The binding preferences of a TF can be graphically represented by a LOGO (**5**). Each position is represented by a stack of symbols. The total height of the stack represents the information content (**6**) at this position (i.e., how conserved this position is); and the height of each symbol in the stack corresponds to the relative frequency the corresponding base appears in this position in the set of known sites. **Figure 1D** shows a LOGO example for the 20 aligned sites.

Even with the knowledge of the binding preferences of a TF (e.g., in the form of a PSSM model) the computational identification of new DNA targets becomes difficult because of the short length of the “signals,” the degeneracy of the patterns and the quality of the PSSM models (*see* **Note 1**). This is especially true for searches on mammalian promoters, where regulatory elements can be observed many kilobases away from the TSS of the gene, thus increasing the search space. To overcome this problem, a number of methods have been developed that take into consideration the evolutionary conservation of the TFBS. This is known as phylogenetic footprinting, a term coined by Tagle et al. (**7**). The basic idea behind phylogenetic footprinting is that the biologically important DNA *cis*-regulatory elements will be conserved throughout evolution. In other words, they should occur in roughly the same location in the promoters of two (or more) orthologous genes. Clearly, this method is going to miss TFBS that have been acquired by an organism in a more recent time (*see* **Note 2**), but it is expected that the number of false-positive predictions will be significantly reduced. This is important for the biologists when they design experiments to test putative binding sites. These tests are

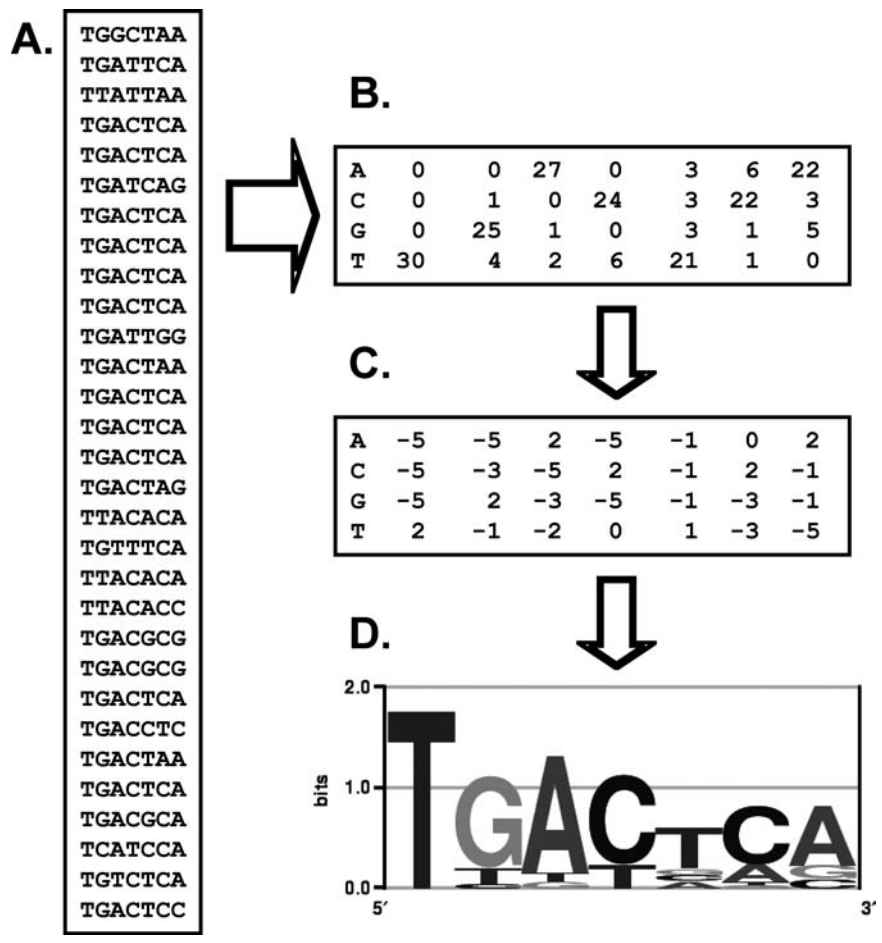


Fig. 1. Position-specific scoring matrix model representation. The information contained in a set of aligned target sites (**A**) is initially encoded into a count matrix (**B**) in which each column represents the number of sequences that contain each of the four bases in that position. For practical purposes, the count matrix is further transformed into some form of log-frequency matrix (**C**). Any of the three forms of information can be graphically represented as a LOGO of symbols (**D**).

usually time-consuming and the focus in a smaller set of high-probability candidate sites is desired.

In this chapter we will focus on the practical aspects of FOOTER, the phylogenetic footprinting algorithm designed for searching for binding sites of

known TFs in the promoters of mammalian genes. FOOTER is not designed for the identification of novel (unknown) DNA patterns in the promoters of multiple coexpressed genes (*see* **Note 3**). The detailed description of the algorithm and a brief introduction to the web tool has been done elsewhere (**8,9**). FOOTER differs from other similar methods in two ways (*see* **Note 4**): one is that it concurrently evaluates both the location of a putative TFBS in the promoter of the gene as well as its PSSM score conservation. In other words, a site is considered to be “true” if the location *and* the PSSM score are conserved between species. Second, the evaluation of these two criteria is quantitative and a weighted average *p*-value (WAP) is calculated.

2. Materials

1. System configuration. The FOOTER web tool and necessary databases are stored on our Dell PowerEdge 2650 server machine consisting of dual 2.8GHz Xeon processors with HT technology and 2 GB of RAM. The algorithm is written in Perl (v5.8.0) using the PG package with a web interface written in PHP (v5.0.3) and associated with MySQL 4.1 database. Alignment is currently performed with program DBA (**10**).
2. Web-tool accessibility. The FOOTER web tool is available over the web at <http://biodev.hgen.pitt.edu/Footer/>. Although it works with virtually any web browser, the application has been optimized for Firefox v1.0.7. Internet browsers are freely available for download from <http://www.mozilla.com/firefox/> (Firefox), <http://browser.netscape.com> (Netscape), and <http://www.microsoft.com/windows/ie/> (Microsoft Internet Explorer).

3. Methods

The FOOTER algorithm has been developed for the efficient identification of evolutionary conserved DNA *cis*-regulatory elements. Currently, it runs on human and rodent promoters only. Each of the human and rodent promoters is scanned independently for candidate TFBS from a set of 95 mammalian TFs with known binding preferences. The PSSM models we use are derived from selected target sequences deposited in TRANSFAC database (**11**). Whenever enough human and rodent sequences are available in TRANSFAC, species-specific PSSM models are constructed; otherwise, a mammalian-specific PSSM model is used. By species- or mammalian-specific model, we mean that the target sequences are in fact biochemically verified targets in human, mouse, or rat genes. We decided to follow this approach instead of general all-species matrix that other programs use because we have found species- or class-specific differences in the binding preferences of some TFs (**9**). The top 10 scoring sites for each 3 kb of analyzed sequence in each promoter are retained for further

analysis. An alignment between the two promoters provides some guidance as to where the conserved sections are located and helps determine the distance between putative sites, while correcting for local insertions/deletions. The idea behind that is that the “location conservation” is biologically important because TFs usually act in concert with other TFs and other proteins. Thus, the local conservation is more important than their absolute distance from the TSS. This is more easily understood if one thinks about the TFBS, which are located few kilobases away from the TSS.

It is important to note that the promoter alignment is only used to point to regions of conservation. However, because each program has its limitations, so does DBA (10) (the alignment program we use). Thus, we allow putative sites to be subsequently analyzed without the restriction of being in a DBA-“conserved” region. The subsequent analysis consists of comparing pairwise all putative sites in the two promoters and scoring their similarity according to the two criteria: (1) their relative distance in the two promoters (determined locally by the conserved regions’ boundaries) and (2) their relative PSSM scores according to the species- (ideally) or mammalian-specific matrices. For each of the two criteria, a *p*-value score is assigned that reflects the probability of observing the corresponding distance and PSSM scores merely by chance. The two *p*-values are weighted and combined in a single WAP score. The pairs with the best (lowest) WAP scores are reported as true binding sites. An outline of the work-flow of the web tool is presented in Fig. 2.

3.1. User Input

1. Promoter sequences. FOOTER requires the input of two DNA promoter sequences in FastA format (see Note 5). According to the FastA format the sequence is preceded by a single line starting with “>” followed by a sequence name or identifier. The rest of the lines contain the raw DNA sequence. One of the input sequences should be human and the other should belong to a rodent; each being placed into the proper field for that species. The sequences can either be copy-pasted (directly) or uploaded as a text file. FOOTER has been shown to be very successful in detecting *cis*-regulatory signals in large promoter sequences (e.g., 3 kb; see Note 6).
2. Selecting transcription factors. FOOTER has PSSM models for 219 TFs, including most of the well-known TFs such as Sp1, EGR-1, and NF-κB. The TFs have been classified into 15 families, based on their structural properties. The individual factors can be found within a folding tree consisting of the protein families. Factors can be selected individually by their checkboxes, or as a family (i.e., all TFs belonging to that family) by the family checkboxes. The users also can select to search for all available TFs by the “check all” function. If a user wants to select a particular factor but does not know its exact name or which family it belongs, there is a string

1. Input Sequences

Input

Human Promoter Region

```

P>Pdx-1 [Human] [Chr1] [contig=Hs1]
4000 bp
TCTCCAGGAGGAGAGCGCTGGAGCAGTGGCT
CGAGATGCCCCCTCTGCTCCCTTCTACTCTC
AATAGACCAATTACCAATTCATTCAGTGGTTC
GATCTCATGATCATCTCGAATAATATCTCA

```

< 1 2 3 4 5 6 7 8 9 10 >

[Browse...](#)

Mouse or Rat Promoter Region

```

P>Pdx-1 [Mouse] [Chr5] [N9474226.322]
4000 bp
CATCTAGTTTGTAAAAAATAAACACAGAGC
TGCACACTTCTAGGGGAGGCCACAGAGACTC
CGTAAATTGAAGCAGACAGAGGGATGAACAC
GGACAGGCATTTAGCCACACCGAGGACCT

```

< 1 2 3 4 5 6 7 8 9 10 >

[Browse...](#)

Please Enter All Sequences in FastA format

Trouble Finding The Human and Mouse Promoter Regions, We Recommend TRED

Still trouble finding promoter regions? Try UCSC Genome Browser or Ensembl



2. Select TFs

[illegible]

3. SNP reporting option

Search Promoter Sequences for SNPs from dbSNP
☐ None ☒ Human & Mouse ☐ Human & Rat



4. Submit job



5. View/analyze results

search function, which returns the TF names that contain the searching string in their name or in one of the synonyms (a TF can have multiple synonyms). Each factor name is hyperlinked so that the users can obtain information about the TFs synonyms as well as a graphical representation of the available PSSM model(s).

3. Single-nucleotide polymorphisms (SNP) presentation. FOOTER has recently made available a function that will allow the identification and presentation of known SNPs in the examined promoters. The SNPs are derived from the dbSNP database (12). Though the presence or absence of SNPs in the promoter region(s) does not weight into the FOOTER analysis, it can provide useful information to the user about possible sites that could be affected by a polymorphism.

3.2. Parameters

1. PF_d and PF_s weights. PF_d is the tail probability (p -value) that two sites are found by chance in a distance equal or less than the observed one. Similarly, PF_s is the tail probability (p -value) that the two PSSM scores are as high (or higher) than the observed ones by chance alone. The calculation of the PF_d is based on a uniform distribution model, whereas the calculation of the PF_s is based on the distribution of scores of each PSSM model. The two p -values are weighted in the negative log-scale and added to give the combined PF score or, equivalently, the WAP (i.e., exponent of the PF score). The weights attached to the PF_d and PF_s represent the influence of each of the two parameters in the total p -value (WAP). We have empirically determined that the weights of 0.85 and 0.15 for PF_d and PF_s , respectively provided the most accurate results (for a performance graph, we refer the readers to Corcoran et al. [9]) with a test set of well-studied promoter regions.
2. WAP. The WAP represents the probability that a pair of putative binding sites could have been found in the observed distance (or closer) with the observed PSSM scores (or better) merely by chance. The two distributions that are taken into account for this value are the PF_d and PF_s . The weight given to each of these two distributions in the calculation of the WAP can be adjusted. Based on our previous tests (9) we have empirically determined that a WAP cutoff of 5×10^{-4} provides the most accurate



Fig. 2. A flowchart of the execution of FOOTER web tool. The orthologous human and mouse promoters are retrieved from one of the suggested web-accessible repositories. Transcription factors are selected from the list, either individually or by structural category. A search utility helps users in the selection of transcription factors. FOOTER provides also information about the various transcription factors, including synonyms and LOGOs with their binding preferences. The users also have the option to request for known single-nucleotide polymorphisms to be displayed in either or both of the promoters.

results (for a performance graph, we refer the readers to Corcoran et al. [9]). This parameter can also be adjusted on the results page.

3. Number of seed sites. The seed parameter adjusts the number of potential binding sites of each TF that FOOTER will retain for subsequent analysis in each promoter. We have decided to retain “blindly” a particular number of sites per kilobase of DNA sequence rather than setting a strict threshold in the PSSM score for selecting sites. The reason for that is that we would like to allow more suboptimal sites to be subsequently compared to the other predictions. The idea is that if two sites are in the same location and similarly “suboptimal” (according to their PSSM scores), then these might be biologically relevant sites. By default, FOOTER retains one seed site in each promoter for every TF for every 300 bp. For a 3000-bp long promoter (the size we usually examine) this corresponds to 10 seed sites.

3.3. The Results Page

1. Interpreting the results. An example of the FOOTER results page is presented in [Fig. 3](#). The output is in both tabular and graphical format showing the reported TFBS predictions. The table lists the TF name, target sequence, flanking sequence, position, and WAP for each pair (human-rodent) of predicted sites that have met the user-specified WAP cutoff. The produced PNG image displays regions and percent of conservation between the human and rodent promoter sequences as calculated by the DBA, program (10) and the reported binding sites. Also available on the results page are links to the following text files: (1) the input sequences, (2) the DBA alignment, (3) the complete FOOTER output (including sites that did not meet the reporting criteria).
2. Adjusting the WAP. On the results page the user can adjust the WAP value threshold, which will reproduce the table and PNG image showing the predicted binding sites that meet the new cutoff. The recalculation of the reported sites is done without the need of rerunning the algorithm.
3. SNPs. If the user had selected to search the human and rodent sequences for SNPs, their location will be displayed in the PNG image. If any SNPs fall within 20 bp of a reported predicted site, the SNP will be identified as a highlighted hyperlink in the tabular results. The link will redirect the user to the dbSNP information page for that particular SNP. Two HTML pages will also be produced that show every SNP found as a highlighted hyperlink within the individual promoter regions.

3.4. Help Pages

FOOTER web tool includes help pages that describe its main features, the sequence input, the parameters and the output of the program. Another page describes the FastA format (input sequences) in detail. In the main page, an e-mail address is also listed for help, comments, and so on. Finally, a “Load example” option provides the user a first experience with the program.

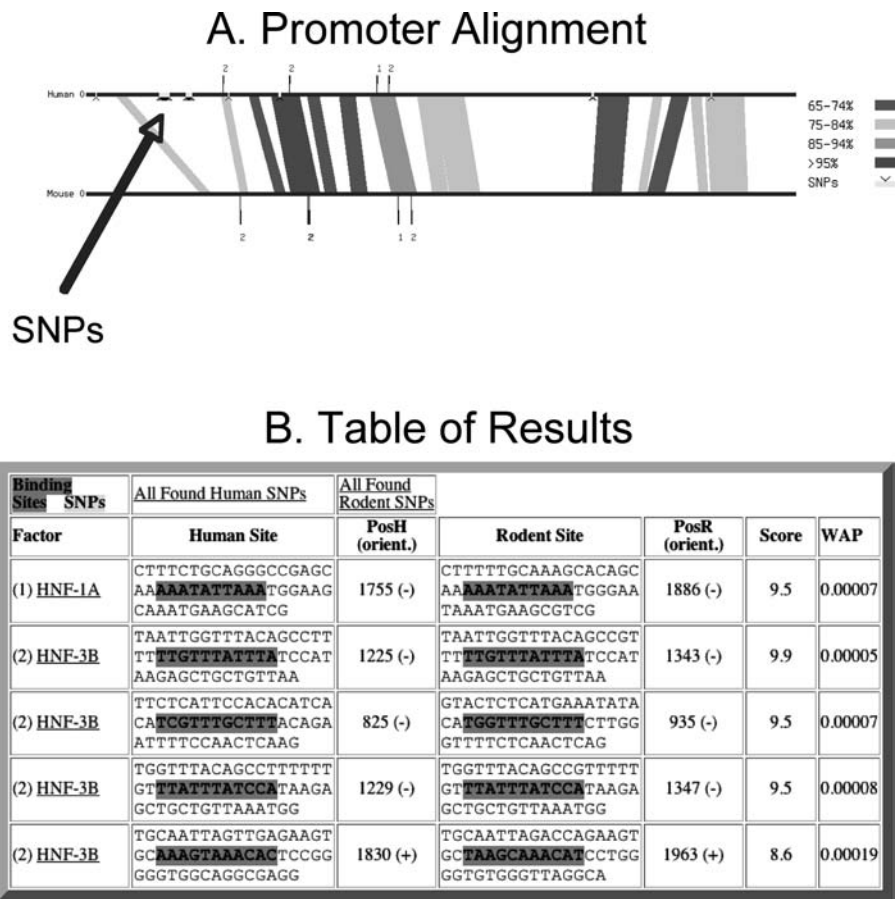


Fig. 3. Example of FOOTER results. FOOTER results page consists of primarily two sections. The upper section presents a color-coded alignment of the two promoters, with different colors denoting different conservation percentages. The lower part of the output presents the predicted sites in a tabular format. This table can be copy-pasted in and analyzed further with any spreadsheet program.

4. Notes

1. The methods that are using prior information, like PSSM models, to search for new DNA motifs are limited by the quality of the PSSM models. There are several factors that affect the quality of a PSSM model. One is the number of sequences that are available. FOOTER reports the number of sequences in the graphical display of LOGOs (just click on the TF name.) Another factor is the degree of

conservation of the pattern, or equivalently, its information content. As a general rule, the higher the information content, the better the model. The information content is measured in bits of information, which can be calculated from the LOGO graphical representation of the pattern. For example, EGR1 pattern has almost 13 bits of information whereas GATA-3 has 9 bits (Fig. 4). Finally, the quality of a PSSM model can be influenced by the number and diversity of species that contribute sequences to it. Because TFs are proteins that change through time, the same is true for their binding preferences (13). Thus, a PSSM model derived from sequences from a variety of evolutionary distant organisms (e.g., mammalian, avian, and amphibian) can obscure the mammalian motifs, resulting in an increased number of false-positive predictions. FOOTER uses species-specific (when possible) or mammalian-specific PSSM models.

2. Because phylogenetic footprinting methods depend on evolutionary conservation to distinguish between true, biologically important DNA “signals” and spurious events, they are going to miss functional *cis*-regulatory motifs that have been recently acquired by an organism. In our case, FOOTER is going to miss the motifs that human or rodents acquired after their last common ancestor.
3. FOOTER is following a single-gene multiple-species approach. Thus, it is not designed to find motifs in the promoters of multiple coexpressed genes, e.g., from microarray experiments (multiple-genes single-species approach). There are a number of other methods available for this purpose (for a recent review, *see* Tompa et al. [14]). The difference between the two approaches is that the first aims to identify targets of known TF proteins, whereas the latter identifies DNA “signals” that are presumed to be targets of some (usually unknown) TF. Recently, a hybrid approach was published (13). The SOMBRERO algorithm, a self-organizing

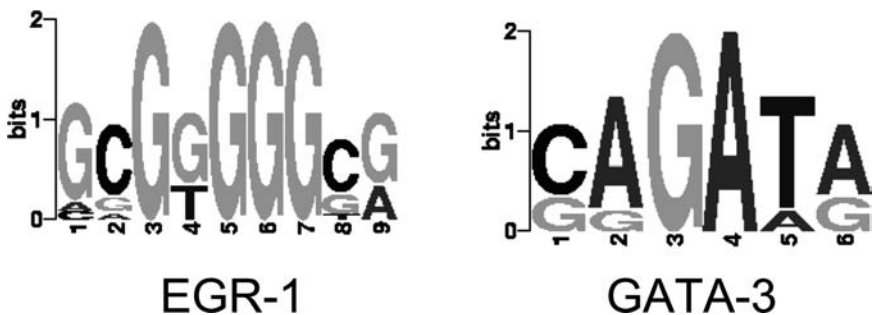


Fig. 4. Examples of LOGOs. The information content of the various transcription factors is indicative of the expected frequency of its target sites in the genome. The examples of two well-known factors are presented here for comparison. Based on that, GATA-3 sites are expected to be more frequent (by chance) in the genome than the EGR-1 sites.

map method, used prior information (PSSM models of familial binding profiles) to search the promoters of coexpressed genes. Thus the predicted motifs were associated with the TF where the initial familial binding profile came from.

4. When one tries to predict DNA regulatory sites in the promoters of genes, it is always useful to run different computational tools and compare the results before one decides to embark in time-consuming biochemical experiments. Besides FOOTER, there are two other methods that offer web-accessible tools for searching mammalian promoters for binding sites of known TFs: ConSite (<http://www.phylofoot.org/consite/>) and rVista (<http://rvista.dcode.org/>). Both algorithms evaluate the candidate sites qualitatively, taking into consideration mainly the position of the site in the promoter (the PSSM score is used to identify the candidate sites based on a threshold). ConSite uses mammalian-specific matrices to search mammalian promoters, whereas rVista uses all-species matrices. In a recent comparative analysis (9), FOOTER outperformed both ConSite and rVista, but the latter algorithm has been updated since.
5. The correct identification of the promoter sequences is important for FOOTER. In the past, we had offered an option for the automatic identification of the promoter sequence of human and mouse genes from a single protein sequence (human or mouse). However, we found that this approach was problematic. Currently, we direct the FOOTER users to obtain the promoter sequences themselves. The recommended source of orthologous promoter sequences is the TRED database (<http://rulai.cshl.edu/cgi-bin/TRED/tred.cgi?process=searchPromForm>). If the promoter of interest is not in the TRED database, then we recommend to the users to obtain the promoter sequences from one of the publicly available genome browsers, like the UCSC Genome Browser (<http://genome.ucsc.edu/cgi-bin/hgGateway>) or the Ensembl (<http://www.ensembl.org/>).
6. Promoter length. FOOTER has been tested on various promoter lengths and it has been shown to increase its sensitivity (sensitivity is the percent of true [confirmed] sites that FOOTER identified) with the length up to 3 kb (9). This is partly because the number of seed target sites that are retained for further analysis is proportional to the analyzed promoter length. Thus, as the promoter length increases, more seed sites are retained. FOOTER has not been tested in promoters with length more than 3 kb, but longer promoters might cause problems related to the reliability of the alignment. Usually, the sequence conservation decreases when the distance from the TSS increases. Thus, for comparison of longer promoters (e.g., 5, 10, 20 kb), it might be useful if the promoters are “split” into smaller, overlapping pieces of 2–4 kb. Promoter pieces in one species that do not share any similarity to the promoter pieces in the other species (typically, long insertions) can thus be omitted from the analysis, resulting into more accurate predictions.

Acknowledgments

This work was supported by National Science Foundation grant MCB03-16255. PVB was also supported by National Institutes of Health grant 1R01LM007994-01 and TATRC/DoD USAMRAA Prime Award W81XWH-05-2-0066.

References

- 1 Stormo, G. D. (2000) DNA binding sites: representation and discovery. *Bioinformatics* **16**, 16–23.
- 2 Benos, P. V., Lapedes, A. S., and Stormo, G. D. (2002) Probabilistic code for DNA recognition by proteins of the EGR family. *J. Mol. Biol.* **323**, 701–727.
- 3 Benos, P. V., Bulyk, M. L., and Stormo, G. D. (2002) Additivity in protein-DNA interactions: how good an approximation is it? *Nucleic Acids Res.* **30**, 4442–4451.
- 4 Benos, P. V., Lapedes, A. S., Fields, D. S., and Stormo, G. D. (2001) SAMIE: statistical algorithm for modeling interaction energies. *Pac. Symp. Biocomput.* 115–126.
- 5 Schneider, T. D., Stormo, G. D., Yarus, M. A., and Gold, L. (1984) Delila system tools. *Nucleic Acids Res.* **12**, 129–140.
- 6 Shannon, C. (1948) The Mathematical Theory of Communication. *Bell System Tech. J.* **27**, 379–423 and 623–656.
- 7 Tagle, D. A., Koop, B. F., Goodman, M., Slightom, J. L., Hess, D. L., and Jones, R. T. (1988) Embryonic epsilon and gamma globin genes of a prosimian primate (*Galago crassicaudatus*). Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.* **203**, 439–455.
- 8 Corcoran, D. L., Feingold, E., and Benos, P. V. (2005) FOOTER: a web tool for finding mammalian DNA regulatory regions using phylogenetic footprinting. *Nucleic Acids Res.* **33**, W442–W446.
- 9 Corcoran, D. L., Feingold, E., Dominick, J., et al. (2005) Footer: a quantitative comparative genomics method for efficient recognition of cis-regulatory elements. *Genome Res.* **15**, 840–847.
- 10 Jareborg, N., Birney, E., and Durbin, R. (1999) Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Res.* **9**, 815–824.
- 11 Wingender, E. (2004) TRANSFAC, TRANSPATH and CYTOMER as starting points for an ontology of regulatory networks. *In Silico Biol.* **4**, 55–61.
- 12 Wheeler, D. L., Barrett, T., Benson, D. A., et al. (2005) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.* **33**, D39–D45.
- 13 Mahony, S., Golden, A., Smith, T. J., and Benos, P. V. (2005) Improved detection of DNA motifs using a self-organized clustering of familial binding profiles. *Bioinformatics* **21**, i283–i291.
- 14 Tompa, M., Li, N., Bailey, T. L., et al. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.* **23**, 137–144.

Exploring Conservation of Transcription Factor Binding Sites with CONREAL

Eugene Berezikov, Victor Guryev, and Edwin Cuppen

Summary

Prediction of transcription factor binding sites (TFBS) is commonly used to formulate working hypotheses for experimental studies on gene regulation. Computational identification of functional TFBS is complicated because of short length and degeneracy of sequence motifs recognized by transcription factors. Information on conservation of predicted sites in orthologous sequences from different species (phylogenetic footprinting) can be used to distinguish potentially functional elements from background predictions. Results of phylogenetic footprinting may substantially depend on the algorithm used to construct an alignment of orthologous sequences, from which conservation of predicted TFBS is estimated. The CONREAL web server allows prediction and comparison of conserved TFBS based on AVID, BLASTZ, CONREAL, and LAGAN alignments. The web tool is particularly suited for the analysis of individual genes or genomic regions, although the underlying algorithm can also be used in high-throughput promoter analysis.

Key Words: Transcription factor binding site; regulatory element; promoter; phylogenetic footprinting; orthologous sequence; alignment.

1. Introduction

Transcription factors (TFs) play a central role in orchestrating gene expression through binding to specific DNA motifs in the vicinity of target genes. Identification of these transcription factor binding sites (TFBS) is thus an essential step in dissecting gene regulatory networks. Information on DNA binding specificity is available for many TFs, and in theory TFBS could be

found by simply looking up relevant DNA motifs in genome sequences. In practice, the short length (usually less than 10 bases) and the degeneracy of the sequences that can be recognized by TFs result in the prediction of very high numbers of potential target sites, from which most are irrelevant. Therefore, additional filtering is required to distinguish potentially functional TFBS predictions from noise. One commonly used approach is to use information on evolutionary conservation of target sites in the analysis. Based on the assumption that functional elements evolve slower than nonfunctional sequences because of selective pressure, most real TFBS are expected to be conserved, in contrast to spurious, nonfunctional matches. Comparison of multiple orthologous sequences to reveal conserved functional elements is known as phylogenetic footprinting (*1*), and has been successfully applied for the discovery of functional TFBS (*2–5*).

There are several essential steps in the identification process of conserved TFBS. First, orthologous sequences of interest from two or more species need to be obtained. Next, TFBS need to be located in these sequences, and finally, conservation of TFBS need to be evaluated. There is a number of versatile tools for scanning genome sequences with either consensus patterns or position weight matrices (PWM), depending on the level of degeneracy of motifs (*6–9*). In PWMs, different weights are assigned to different bases at particular positions, providing a more flexible definition of a consensus TFBS as compared to patterns (*10*). TransFac (*9*) and Jaspar (*11*) are well-known repositories of PWMs collected from public literature sources.

To evaluate conservation of a TFBS, an alignment of orthologous sequences is required. As a result, the success of phylogenetic footprinting approach depends to a large extent on the selection of species used in the analysis and the quality of produced alignments. Although aligning promoter sequences from closely related species is relatively straightforward, these alignments may lack the desired resolution power. On the other hand, alignments of diverged sequences can be more informative but are also more difficult to make, especially short regulatory sequences such as TFBS may be aligned incorrectly and remain undetected as conserved elements (*12, 13*). The general purpose aligners, such as ClustalW (*14*), LAGAN (*15*), AVID (*16*), or BLASTZ (*17*), can be used to produce alignments for phylogenetic footprinting. However, they were not designed to specifically address the problems of aligning diverged promoter sequences and TFBS identification. To this end we developed a conserved regulatory elements anchored alignment (CONREAL) algorithm (*18*), which takes into account the presence of a potential TFBS for constructing a promoter sequence alignment. CONREAL produces results that are comparable to

traditional phylogenetic footprinting approaches when applied to less diverged sequences (e.g., human and mouse), and was found to be more sensitive in analysis of diverged sequences (e.g., human and fugu). Although the majority of conserved TFBS are readily identified by different alignment methods, there is a substantial fraction of validated functional binding sites that were identified only in alignments produced by a specific algorithm and not by another (18). As a result, intersection of predictions based on different alignments can provide a better overall picture of TFBS conservation. Therefore, we developed a web interface to CONREAL (<http://conreal.niob.knaw.nl>), which also provides the option to run LAGAN, AVID, and BLASTZ aligners, and to visualize and compare results obtained by the different approaches (19). In addition, the CONREAL web server facilitates automated retrieval of orthologous promoter sequences from the Ensembl database (20) that are needed as input for the various algorithms, although any combination of custom sequences can be analyzed as well. The web server is very well suited for the detailed analysis of individual promoters by any (experimental) scientist, whereas a standalone version of CONREAL can be downloaded for local installation and high-throughput analysis by trained bioinformaticians.

2. Materials

1. CONREAL web server is accessible at <http://conreal.niob.knaw.nl>.
2. Standalone version of CONREAL software can be downloaded from <http://conreal.niob.knaw.nl/standalone>.

3. Methods

3.1. Preparation of Sequences for Analysis

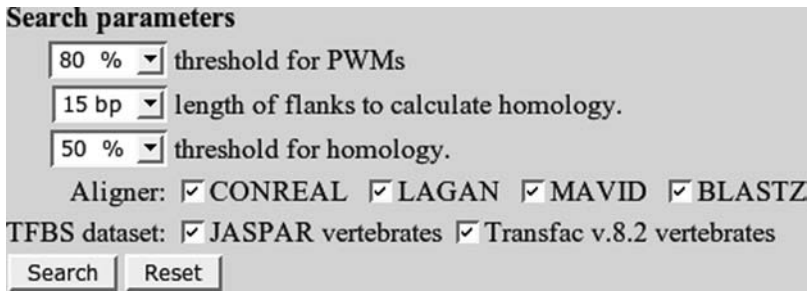
The CONREAL web server requires two DNA sequences in Fasta format as input for analysis. It is assumed that a user already has some knowledge about the region of interest (e.g., promoter of a certain gene) and can provide the correct orthologous sequences. Alternatively, the web server can assist a user in retrieving relevant promoter sequences.

1. If sequences were prepared outside the CONREAL framework (*see Note 1*), paste them in FASTA format in the text window (Fig. 1A) or provide a name of the file with the sequences (plain text file containing two sequences in FASTA format) (Fig. 1B). Proceed to **Subheading 3.2**.

[illegible]

Fig. 1. CONREAL sequence input form. **(A)** Two sequences in Fasta format can be pasted into the text field or **(B)** provided in a plain text file or **(C)** sequences can be automatically retrieved from the Ensembl database using a gene name or keyword and a species name.

2. To retrieve promoter sequences of a certain gene with assistance of the server, provide gene name or keywords in the text field (e.g., forkhead) and specify the organism of interest (e.g., *Mus musculus*). Press “Get gene from Ensembl” button (**Fig. 1C**).
3. On the next screen a list of Ensembl genes matching the query term will appear (**Fig. 2**). If the search produced no results, try different terms (*see Note 2*). If several genes are listed or the identity of the desired gene is not clear, follow the Ensembl link provided in the gene description field to see additional gene information. Once the ID of the correct gene is established, proceed to the next step by clicking on the gene ID.
4. The server generates a list of orthologous genes from different organisms based on Ensembl annotations. This list contains three fields: organism, gene and orthology type, and gene description (**Fig. 3A**). Select an organism and a gene to be included in the analysis. Only one gene can be selected at a time. Similarly, when the identity of a gene is not clear from the information provided, follow the link provided in the gene description field to find more information in the Ensembl database. In the case, when several genes are listed for a particular organism, annotation of orthology type can help to reach a decision: true orthologs are usually annotated as UBRH (Universal Best Reciprocal Hit).



Search parameters

80 % threshold for PWMs

15 bp length of flanks to calculate homology.

50 % threshold for homology.

Aligner: ☒ CONREAL ☒ LAGAN ☒ MAVID ☒ BLASTZ

TFBS dataset: ☒ JASPAR vertebrates ☒ Transfac v.8.2 vertebrates

Search Reset

Fig. 4. A variety of analysis parameters can be set.

3.2. Setting Analysis Parameters

There are six different parameters that can be changed at the bottom of the submission page (Fig. 4).

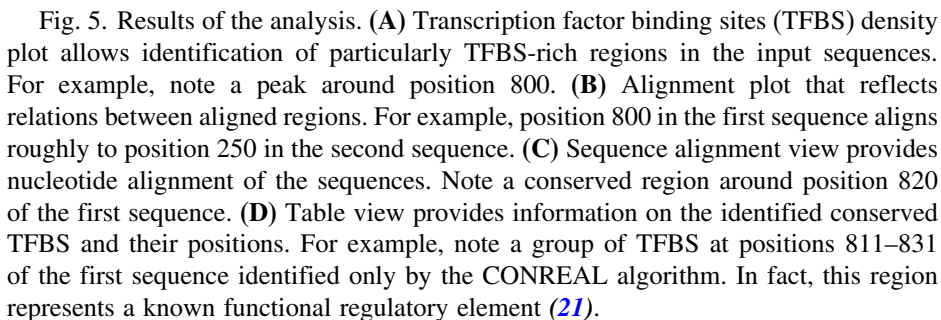
1. Set the threshold for PWMs identity. TFBS that are predicted using PWMs are assigned scores during the identification process. This threshold sets a minimum relative score of a TFBS hit to be considered in the analysis, in percentages. The lower PWM threshold, the more sites will be predicted but also more false-positives are found. The default value is set to 80%. Decrease the stringency for analysis of diverged sequences (e.g., from human and fish) and increase when closely related species are analyzed.
2. Set the length of flanks to calculate local homology, in bases. For every TFBS found, flanking sequences of the given length will be added when estimating conservation between sites. The default value is 15 bases. The longer the length of the flanks, the more weight will be assigned to TFBS with conserved context. Similarly, decrease the length for analysis of diverged sequences. Length of 0 bp means that only conservation of TFBS itself will be evaluated.
3. Set the threshold for homology. Only TFBS with conservation above the given threshold will be considered in the analysis. The default value is set to 50%.
4. Select the alignment methods to use. Available options are CONREAL, LAGAN, MAVID, and BLASTZ. At least one method should be selected (*see Note 4*).
5. Select PWM libraries to use. Jaspar is a publicly available database of manually curated high quality PWMs (11), whereas TransFac is a commercial database with only part of the content available free for academic use (9). Note that the CONREAL web server only uses the vertebrate subsets of PWMs from these databases (*see Note 5*).
6. Use the “submit” button to start the analysis or “reset” to revert all settings to default values.

7. Once the job is submitted to the server, a page displaying the job status will appear. The page is refreshed automatically every several seconds until the results are ready (see **Note 6**).

3.3. Interpreting Analysis Results

CONREAL web server produces three types of output: a graph visualizing alignment and TFBS density, a sequence alignment at the nucleotide level, and a table listing the conserved TFBS found. When several alignment methods were selected for analysis, the results for each method are presented individually in the same format.

1. The TFBS density plot (**Fig. 5A**) reflects how many different conserved TFBS are found in a particular region of the alignment. It facilitates the identification of potential regulatory regions that are expected to have a higher density of TFBS than surrounding sequences. Check if there are particularly dense regions on the histogram and note sequence coordinates for closer investigation of these regions in the alignment and table views (see **Note 7**). If fewer conserved TFBS were found than expected, it is likely that incorrect orthologous regions were used for the analysis (see **Notes 3 and 4**).
2. The alignment plot reflects which regions of the two sequences are aligned together (**Fig. 5B**). It helps to identify potential problems with sequence selection, e.g., a systematic skew owing to incorrect annotation of the first exon in one of the selected species (see **Note 3**).
3. The sequence alignment view provides information on aligned sequences at the nucleotide level, with conserved TFBS shown in uppercase letters (**Fig. 5C**) and coordinates shown above and below sequences (see **Note 8**). Use these coordinates to locate and inspect the alignments for the TFBS-dense regions identified in **step 1**.
4. Finally, the table with the conserved TFBS that are found is displayed, along with PWM ID used to identify the particular TFBS, conservation of the site, coordinates of the site in both sequences, strand of the sequences, PWM scores, TF annotation, and support for the site by other alignment methods (**Fig. 5D**). PWM IDs and annotations are linked to Jaspas and TransFac databases, where additional information about PWMs and TFs can be found (see **Note 9**). Use the coordinates field to investigate which TFs contribute to dense TFBS regions. Alternatively, distribution of TFBS for a particular TF can be investigated using the PWM ID and description fields.
5. To identify and prioritize regulatory regions for downstream applications (e.g., for experimental validation), combine information from different views to come to a working hypothesis. Generally, overlap in predictions between different methods can be used as a good proxy to assign a confidence level for a particular predicted regulatory element. However, predictions specific to one particular method can also be useful and should not be immediately discarded (**Fig. 5**).



6. It is recommended to run the analysis several times with different parameter values to estimate how results depend on different threshold levels. The most conserved elements tend to have little dependency on stringency of parameters (*see Note 10*).
7. To obtain additional support for TFBS predictions in the region of interest, investigate the conservation of the TFBS in another combination of species. This can be achieved by rerunning the analysis with orthologous sequence from different species. TFBS conserved between multiple species are more likely to represent real regulatory elements.

3.4. Using the Standalone Version of CONREAL

The CONREAL web server is designed for discovery and exploration of TFBS on a gene-by-gene basis. For high-throughput analysis, a standalone version of CONREAL can be downloaded from the site and installed on a local machine. Detailed instruction on installation and usage of the software are provided in the README file that is included in the package. Note that the standalone version only implements the CONREAL algorithm and does not assist in the retrieval of orthologous sequences.

4. Notes

1. Sequences should be provided in Fasta format, in which the first line begins with ">" followed by the sequence ID and an optional description. The nucleotide sequence itself starts from the second line and may contain spaces and numbers that are ignored by the program. An example of the format is shown on the main page of the server. It is always necessary to provide two sequences for the analysis.
2. The CONREAL web server forwards gene query terms to the Ensembl server (<http://www.ensembl.org>) and thus depends on annotation terms and descriptions that are present in this database. When exact search terms are not found or when the desired results are not returned, try to identify the Ensembl ID of the required genes by a more general search directly on the Ensembl web server or another genome database of choice, and use this Ensembl gene ID in the query field on the CONREAL server.
3. The CONREAL web server uses Ensembl annotations to identify the start position of a gene and uses the corresponding genomic coordinates for retrieval of the specified regions relative to this position. The same relative regions are retrieved for orthologous genes. However, first exons are notoriously difficult to annotate and, therefore, start positions are often inaccurately annotated in orthologous genes, and hence relative regions retrieved by the CONREAL server are not necessarily biologically corresponding regions. To account for these potential shifts in gene start annotations, first retrieve larger regions (~10 kb) and run a test analysis to evaluate the correctness of the annotations.

4. When analyzing long sequences, first do a test run with only one of the methods to assess whether the orthologous regions were defined correctly (*see Note 3*). When a skew (a systematic shift in alignment positions over the whole region analyzed) is observed, it may be necessary to trim or extend one of the two sequences.
5. TransFac contains substantially more PWMs than the Jaspar database. However, many PWMs are redundant and represent the same TF. For test runs, it is recommended to only select the Jaspar database to speed up computation.
6. Most of the jobs are expected to finish within 1–3 min. However, depending on the load on the server, analysis can take longer. Wait until results are displayed. When the analysis takes too long to finish, contact the server administrator (contact details are available on the Description page). Mention the job ID in the correspondence when reporting problems with the server.
7. Coordinates provided in the results section are absolute coordinates with the +1 position corresponding to the beginning of the submitted sequences. Note that these coordinates are different from coordinates used to retrieve sequences from Ensembl database at **Subheading 3.1., step 5**.
8. Note that for the alignments that are generated by the CONREAL algorithm the regions that do not contain conserved TFBS are shown in lowercase and may appear as unaligned or misaligned. This is because the CONREAL algorithm does no attempts to align regions between conserved TFs. The web server is intended for the identification of conserved TFBS and should not be used to compare promoter sequence alignments produced by the different algorithms.
9. To view information from TransFac database, a registration may be required, which is free for academic users. Information about most recent additions to the database is available only to commercial users. Entries, for which information is not publicly available, are marked “Pro only” and linked to TransFac Pro database.
10. Unlike other algorithms, CONREAL uses TFBS information to infer alignments, and result can substantially vary depending on parameter values as well as PWMs used for predictions.

References

- 1 Tagle, D. A., Koop, B. F., Goodman, M., Slightom, J. L., Hess, D. L., and Jones, R. T. (1988) Embryonic epsilon and gamma globin genes of a prosimian primate (*Galago crassicaudatus*). Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.* **203**, 439–455.
- 2 Gumucio, D. L., Heilstedt-Williamson, H., Gray, T. A., et al. (1992) Phylogenetic footprinting reveals a nuclear protein which binds to silencer sequences in the human gamma and epsilon globin genes. *Mol. Cell. Biol.* **12**, 4919–4929.
- 3 Aparicio, S., Morrison, A., Gould, A., et al. (1995) Detecting conserved regulatory elements with the model genome of the Japanese puffer fish, *Fugu rubripes*. *Proc. Natl. Acad. Sci. USA* **92**, 1684–1688.

- 4 Loots, G. G., Locksley, R. M., Blankespoor, C. M., et al. (2000) Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science* **288**, 136–140.
- 5 Wasserman, W. W., Palumbo, M., Thompson, W., Fickett, J. W., and Lawrence, C. E. (2000) Human-mouse genome comparisons to locate regulatory sites. *Nat. Genet.* **26**, 225–228.
- 6 Lenhard, B. and Wasserman, W. W. (2002) TFBS: computational framework for transcription factor binding site analysis. *Bioinformatics* **18**, 1135–1136.
- 7 Kel, A. E., Gossling, E., Reuter, I., Cheremushkin, E., Kel-Margoulis, O. V., and Wingender, E. (2003) MATCH: a tool for searching transcription factor binding sites in DNA sequences. *Nucleic Acids Res.* **31**, 3576–3579.
- 8 Cartharius, K., Frech, K., Grote, K., et al. (2005) MatInspector and beyond: promoter analysis based on transcription factor binding sites. *Bioinformatics* **21**, 2933–2942.
- 9 Matys, V., Kel-Margoulis, O. V., Fricke, E., et al. (2006) TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res.* **34**, D108–D110.
- 10 Stormo, G. D. (2000) DNA binding sites: representation and discovery. *Bioinformatics* **16**, 16–23.
- 11 Vlieghe, D., Sandelin, A., De Bleser, P. J., et al. (2006) A new generation of JASPAR, the open-access repository for transcription factor binding site profiles. *Nucleic Acids Res.* **34**, D95–D97.
- 12 Cliften, P. F., Hillier, L. W., Fulton, L., et al. (2001) Surveying *Saccharomyces* genomes to identify functional elements by comparative DNA sequence analysis. *Genome Res.* **11**, 1175–1186.
- 13 Tompa, M. (2001) Identifying functional elements by comparative DNA sequence analysis. *Genome Res.* **11**, 1143–1144.
- 14 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 15 Brudno, M., Do, C. B., Cooper, G. M., et al. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13**, 721–731.
- 16 Bray, N., Dubchak, I., and Pachter, L. (2003) AVID: a global alignment program. *Genome Res.* **13**, 97–102.
- 17 Schwartz, S., Kent, W. J., Smit, A., et al. (2003) Human-mouse alignments with BLASTZ. *Genome Res.* **13**, 103–107.
- 18 Berezikov, E., Guryev, V., Plasterk, R. H., and Cuppen, E. (2004) CONREAL: conserved regulatory elements anchored alignment algorithm for identification of transcription factor binding sites by phylogenetic footprinting. *Genome Res.* **14**, 170–178.

- 19 Berezikov, E., Guryev, V., and Cuppen, E. (2005) CONREAL web server: identification and visualization of conserved transcription factor binding sites. *Nucleic Acids Res.* **33**, W447–W450.
- 20 Birney, E., Andrews, D., Caccamo, M., et al. (2006) Ensembl 2006. *Nucleic Acids Res.* **34**, D556–D561.
- 21 Nishizaki, Y., Shimazu, K., Kondoh, H., and Sasaki, H. (2001) Identification of essential sequence motifs in the node/notochord enhancer of *Foxa2* (*Hnf3beta*) gene that are conserved across vertebrate species. *Mech. Dev.* **102**, 57–66.

Computational and Statistical Methodologies for ORFeome Primary Structure Analysis

Gabriela Moura, Miguel Pinheiro, Adelaide Valente Freitas, José Luís Oliveira, and Manuel A. S. Santos

Summary

Codon usage and context are biased in open reading frames (ORFs) of most genomes. Codon usage is largely influenced by biased genome G+C pressure, in particular in prokaryotes, but the general rules that govern the evolution of codon context remain largely elusive. To shed new light into this question, we have developed computational, statistical, and graphical tools for analysis of codon context on an ORFeome wide scale. Here, we describe these methodologies in detail and show how they can be used for analysis of ORFs of any genome sequenced.

Key Words: Genome; ORFeome; gene primary structure; codon context; codon usage.

1. Introduction

Genome sequencing is opening unprecedented ways for understanding the primary structure of open reading frames (ORFs) on a global scale (ORFeome) and the evolutionary forces that shape them. Codon usage has been intensively studied in many organisms and one already has a relatively good understanding of the structural and functional constraints that shape its evolution. Conversely, other important features, such as codon context (two neighbor codons), tandem codon repeats, and amino acid composition have not been so well studied and we are still far from understanding their importance for gene stability, mRNA decoding efficiency, and accuracy (1–5). Codon context is rather interesting because it is biased and has an important impact on tRNA decoding accuracy

but the rules that define good and bad context of neighbor codons are not yet understood. Additionally, it is not yet clear whether codon context is used to regulate speed of mRNA translation, if it influences ribosome drop out during elongation and how genes with bad codon context are translated under physiological stress. Considering that mRNA decoding accuracy is critical to ensure correct flow of genetic information from DNA to protein, understanding those rules is likely to provide new insight on the constraints imposed by the mRNA translation machinery on gene evolution. More importantly, codon context rules would allow one to redesign ORFs for optimal gene expression in heterologous hosts (6,7). This is of practical relevance because previous studies carried out in our laboratory have shown that codon context is species specific and consequently heterologous genes do not have the most appropriate context for translation by the host translational machinery.

Traditional methods for codon usage and context analysis do not provide user-friendly tools to study gene primary structure on a genomic scale. Codon usage tables, using absolute metric, are available in public databases for any sequenced gene or genome (<http://www.kazusa.or.jp/codon/>) and free-ware software for multivariate analysis (correspondence analysis) of codon and amino acid usage is also readily available (<http://bioweb.pasteur.fr/seqanal/interfaces/codonw.html>). However, sophisticated statistical and data visualization tools are clearly lacking. To study context bias in complete ORFeomes, we have constructed a bioinformatics system herein named ANACONDA, which imports FASTA files, and performs a series of analyses that permit elucidating how codons are associated in consecutive pairs, either in coding sequences or in noncoding regions. This methodology allows to differentiate general biases imposed by general rules of genome evolution, which are related to DNA replication biases (8–11), from biases imposed by the mRNA translational machinery (1,2,12–18). In here, we describe the architecture of ANACONDA and how it can be used to analyze gene primary structure on an ORFeome scale.

2. Materials

ANACONDA is a software package specially developed for the study of genes' primary structure. It reads ORFeomes downloaded from public databases in FASTA format and uses a set of statistical and visualization methods to reveal information about codon context, codon usage, and nucleotide repeats within ORFs. The general features of ANACONDA are described below:

1. Software: the ANACONDA software was developed in C++ language with Microsoft Foundation Class, it runs on MS Windows, and can be downloaded for noncommercial use from the website: <http://bioinformatics.ua.pt/aplicacoes/anaconda>.

2. Requirements: Windows (98/Me, NT 4.0 SP6, 2000 or XP), with a 600 MHz Intel Pentium III or equivalent processor and 128 MB of RAM memory (256 MB or more is recommended). There should also be 100 MB of available disk space and the minimum resolution should be 800×600 (1024×768 or higher is recommended).
3. Data: the genomes files processed by the ANACONDA must be in FASTA format.
4. Parameters: the reference values of relative synonymous codon usage (RSCU) are necessary for the calculation of the codon adaptation index (CAI) (19). This data must be introduced manually. In the current version, the ANACONDA database includes RSCU values for *Candida albicans*, *Saccharomyces cerevisiae*, and *Escherichia coli*.

3. Methods

In this section, we describe the main tools of ANACONDA which are divided into four main parts, namely: (1) uploading and validation of DNA sequence data into the local database, (2) building ORFeome maps for two-codon context bias, (3) visualization and analysis of the two-codon context biases in individual sequences, and (4) comparison of the codon biases across multiple ORFeomes. Also, and taking advantage of the fact that ANACONDA interprets DNA sequences as sequences of trinucleotides (codons), several tools regarding codon usage analysis have been implemented, as explained next.

3.1. Statistics Methods

ANACONDA uses contingency tables as the basic statistical methodology and identifies preferred and rejected codon pairs of an ORFeome through the analysis of adjusted residuals values of the contingency tables. The following list highlights the main statistical procedures performed by the software.

1. ANACONDA uploads ORFeome sequences from any genome and reads them in the 5' to 3' direction fixing each codon (ribosomal P-site) and memorizing its neighbour codons (A-site codon).
2. The data extracted by ANACONDA is then transferred to a 64×61 contingency table with two categorical variables: A and B (Table 1). Variable A represents the 64 possible codons located in the ribosome P-site and variable B represents the following codon (A-site codon) for each observed codon pair in the ORFeome (see Note 1).
3. ANACONDA then calculates the value of the Pearson's chi-squared statistic and the adjusted Pearson residual values. Pearson's statistic represents a global measure of the difference between observed and expected codon frequencies (20).
4. If the hypothesis of independence between the variables A and B, i.e., between contiguous codons, is rejected ANACONDA determines the contributions of each 64×61 codon pairs to Pearson's statistic value computing the adjusted residual values (21).

Table 1
Contingency Table^a

A \ B	AAA	AAC	...	...	...	UUU
AAA	n_{11}	n_{12}	...	n_{1j}	...	$n_{1,64}$
AAC	n_{21}	n_{22}	...	n_{2j}	...	$n_{2,64}$
...						
...	n_{i1}	n_{i2}	...	n_{ij}	...	$n_{i,64}$
...						
UUU	$n_{64,1}$	$n_{64,2}$	...	$n_{64,j}$	...	$n_{64,64}$

^a n_{ij} is the absolute frequency of the codon pair (Ai,Bj) in the ORFeome, where Ai represents one codon in ribosomal P-site and Bj the following codon in ribosomal A-site.

5. The obtained adjusted residual value, for each pair, is then converted into a gray scale and the information is displayed in a 64×61 codon context map, where light gray represents positive adjusted residual values greater than +5 (herein called preferred codon pairs) and the dark gray represents negative adjusted residual values lower than -5 (herein called rejected codon pairs). The adjusted residual values that fall within the interval of -5 to +5 correspond to codon contexts that do not contribute to context bias for confidence levels greater than 99% (21) and are shown in black.

3.2. Uploading of Raw Data

1. Reading sequences. The ANACONDA reads genome sequences stored in FASTA format (see Note 2). The length of each ORF and the number of ORFs in a single file are virtually unlimited. Several files can be opened simultaneously. The imported data, coming from single or multiple files, is classified in a hierarchical tree view, considering three different information levels: species, chromosomes and genes.
2. Validation of ORFs. When scanning the ORFeome, ANACONDA filters pseudo-genes or erroneous ORFs resulting from deficient annotation and/or sequencing errors. A number of quality controls are defined to allow for filtering the ORFeome. For example, very small ORFs (usually less than 100 nt in length), ORFs whose nucleotide sequences are not multiple of three, ORFs without stop codons or ORFs with premature stops, are excluded. Each rule can be individually activated according to user needs. For instance, if the goal is the analysis of all coding and noncoding sequences, all validation controls can be deactivated before opening the files.
3. Data processing (quantification). The imported sequences are then processed according to the statistical methodology that reveals the irregularities in the codon

context along the genome. In this phase, sequence processing can be avoided if the aim is to apply data from a previous statistical analysis to a current analysis. Also, sequences with particular characteristics, or groups of genes can be excluded (at the beginning or at the end) from quantification. The length of the codon context can also be modified, i.e., instead of analyzing codon-pairs, triplets of codons or long range context effects can be studied.

4. Evaluation of the sequences quality. Once the raw data is processed, ANACONDA generates a report showing rejected ORFs and a small description of the rejection. Valid ORFs, using particular set of filters, are shown on a specific menu “Valid Tab” on the left panel of the screen (**Fig. 1**). ORFs excluded from analysis appear in the “Rejected Tab” of the same panel. This allows simple visual inspection of all sequences present in the original FASTA files.

3.3. Working With Genomic Maps of Two-Codon Context

1. Creating an ORFeome context map. After processing valid sequences, an entry with the species' name, as given by the user, will appear on the left panel of the main window of ANACONDA (**Fig. 1**). This panel follows a hierarchical architecture with individual sequences, chromosomes, and genomes. Clicking on each group of ORFs, i.e., chromosome or genome, will open the respective map for two-codon context bias on the right panel of ANACONDA's main window (*see Note 3*).
2. Interpretation of genomic maps. The map represents the bias detected for two-codon contexts in the selected set of ORFs. The bias is given by a gray scale in which dark gray stands for rejected and light gray stands for preferred codon pairs, in relation to what would be expected in a non-association basis. Each possible combination of two codons, i.e., each possible context, is represented by one small square of the map and identified by the codon of the row and the codon of the column to which the small square belongs (*see Note 4*).
3. Data from individual contexts. To facilitate interpretation and analysis of genomic maps, the two-codon contexts can be selected with the cursor and individual information from them will be displayed in the status bar of the software's window. These include: (1) number of genes used to calculate the bias, (2) full name of both axes of the map, (3) residual value for that context, and (4) occurrence for that codon pair in the genome under analysis.
4. Additional data. Apart from the data directly included in the map, ANACONDA produces additional data about the sequences analyzed, namely:
 - a. Codon counting and rare codons. The frequency of each codon is plotted in a graph, for a chosen set of sequences, either for one chromosome or for the entire genome. This can be obtained with the tool Options→Rare Codon, because it allows determination of a threshold for codon usage that automatically indicates whether a codon is rare (*see Note 5*). This window also presents the total number of codons present in all valid ORFs of an ORFeome.

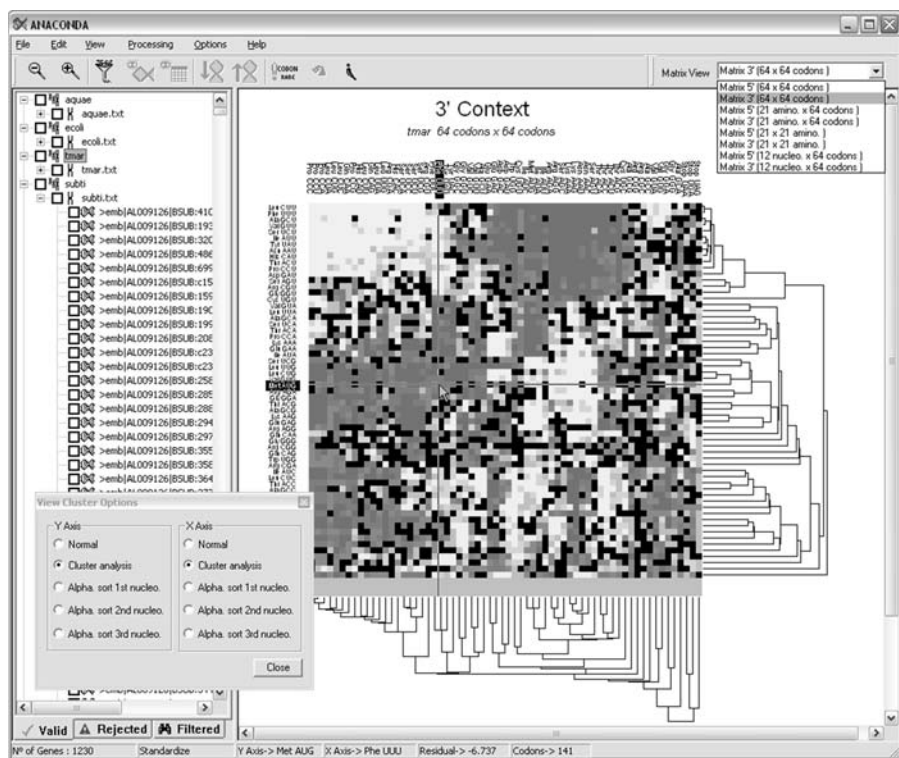


Fig. 1. Main window of the software package ANACONDA for two-codon context analysis at the ORFeome map level. The left panel presents a hierarchical tree of all genomes under analysis by ANACONDA. The Tab Valid includes all individual ORFs used to determine context bias and to build the respective map, whereas the Tab Reject allows visual inspection of ORFs that do not comply with the criteria selected during the opening of the ORFeome. An ORFeomic map for two-codon context bias obtained with the total set of predicted coding sequences of *Thermotoga maritima* (accession number AE000512 from GenBank), is shown.

- b. Nucleotide counting. Codon context has been further explored focusing on the relative frequency of each nucleotide on each position of the neighbour codon, either at the 5' or at the 3' sides. This information is available on dialog Options→Nucleotide Counting that produces a graphical visualization of nucleotide neighborhood for any given codon.
5. Further manipulation of the map. Certain aspects of the map for two-codon context can be altered by the user.

- a. Colors and intervals. The colors used to represent the deviation from the expected mean (the residuals scale) can be chosen from a color palette on the Options menu. Also, the residual values defining the different intervals can be modified by the user.
 - b. Cluster analysis. To define codon context patterns both axes of the 64×61 map can be clustered (22). Additionally, columns or rows can be ordered alphabetically by the nucleotide at each codon position (N1, N2, or N3). This approach was implemented in response to the preliminary observation that some positions from two consecutive codons are highly correlated (23) (see Note 6).
 - c. Exporting images. The entire map or parts of it can be copied and pasted as images into other applications (using the drag-and-drop or the edit-copy functionalities).
6. Exporting data. The numerical data that give origin to a map can be exported as an Excel worksheet. This will include raw data and residuals data of all map layouts, i.e., 64×61 codons, 21×21 amino acids, and so on, through the option File→Save Matrix.

3.4. Working With Individual ORF Sequences

1. Mapping ORFs. To detect the impact of codon context bias (as well as the presence of rare codons) on coding sequences, ANACONDA has additional tools for sequence mapping. These can be activated by selecting individual ORFs on the hierarchical left panel of the software's main window (Fig. 2). The layout for sequence analysis (called "view gene") will appear on the main panel and include written information about the ORF and the sequence itself, in which the codons have been colored with the same residual color scale of the ORFeome map. Again, passing the cursor over the sequences will highlight additional information about each selected context in the status bar of the main window. The threshold for coloring the sequences, together with the choice for mapping rare codons on them can be customized by the user at the dialog Options→View Gene.
2. Exogenous ORFs and codon optimization. To optimize ORF sequences for heterologous gene expression, or for *de novo* gene synthesis, ANACONDA has an algorithm that color codes the sequence of the heterologous ORF according to the codon context rules of the host expression system. For this, the user must open the heterologous ORF sequence using the "no quantification" option (see Note 7) and then redirect the file to the genome of the host of interest (see Note 8). The display window will then show the distribution of good and bad context for that gene.
3. Additional information. Apart from the sequence information shown in the gene view layout (see Note 9), the program offers additional information, obtained from individual sequences or groups of sequences, i.e., chromosomes or total ORFeomes. Selecting the Global gene information option in the View menu the available information about that particular sequence will be displayed (see Note 10). This includes codon and amino acid counting and also several indexes relevant for codon

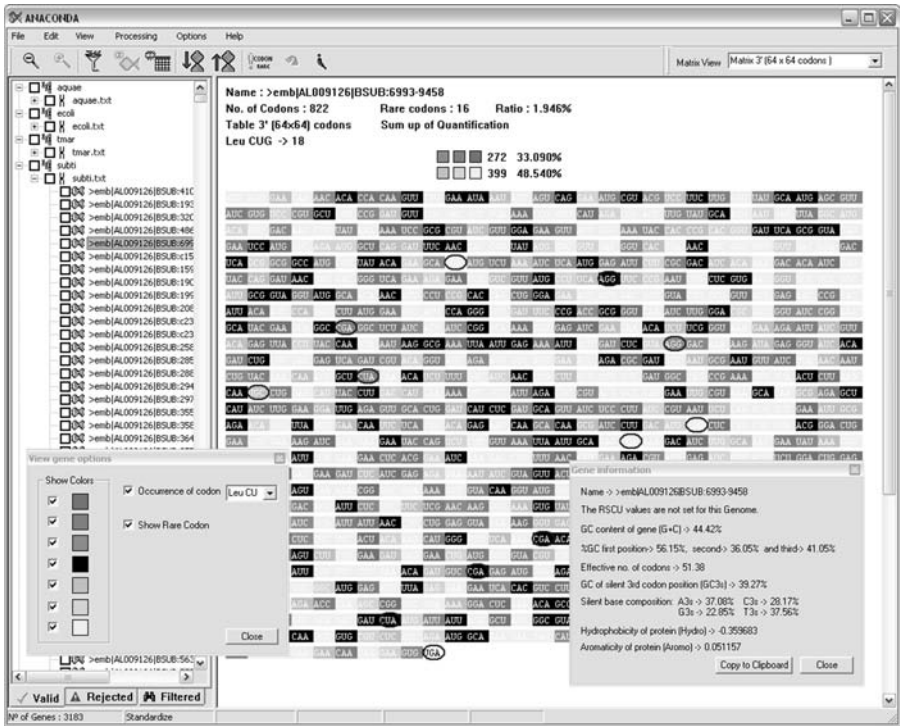


Fig. 2. Main window of the software package ANACONDA for two-codon context analysis in total genomes at the gene view level. Individual ORFs that were used to calculate codon context bias are shown in the hierarchical left panel. Clicking on one of them changes the main panel into the gene view layout. This is composed of a header with the name of the ORF as stated in the original file and the sequence itself. This sequence is colored according to the residual color scale obtained for that ORFeome, i.e., each codon pair is colored in the ORF sequence with the same color scale that it had in the ORFeome map for two-codon contexts. Rare codons are highlighted using circles.

usage characterization, such as G + C content at individual codon positions (first, second, or third); the effective number of codons (24); the RSCU value for each codon; and the corresponding CAI (19) (see Note 11).

4. Filters. Searching for specific ORFeome features can be performed using subsets of ORFs. The sequences that comply with the imposed rules are presented in a special tab in the left panel (Filtered). The available “filters” include: (1) searching for special color patterns or codon/amino acid sequences; (2) searching for runs of up to six rare codons; (3) looking for ORFs rich in bad contexts or rare codons; and (4) finding ORFs whose G + C% is included in a chosen interval. This filter

tool is very useful for studying the distribution of these variables along an entire ORFeome. It also helps finding specific sequences or ORFs with extreme values for a particular variable (*see* **Note 12**).

5. Image and data exporting. As with genomic maps, any part of the gene view layout can be selected and copied into another application. Also, numerical data associated with filtered ORFs can be exported as Excel worksheet by clicking on the ORF set at the Tab Filtered window with the right mouse button.

3.5. Working With More Than One ORFeome

1. Workspaces. ANACONDA allows the user to work with more than one ORFeome at a time. This creates large data sets that are difficult to deal with, in particular when multiple comparisons are being performed. To overcome this problem, ANACONDA has a Workspaces interface that permits saving all data sets, thus eliminating the need of repeating ORFeome analysis manually each time one inter-ORFeome analysis is required. When relevant ORFeomes have been opened for the first time the software creates a file of pathways that allows ANACONDA to reopen the same files at any time (*see* **Note 13**).
2. Visualization. All opened files are named as entered by the user, are represented in the hierarchical left panel and sorted by opening order. In this way, each file can be selected, “navigated,” and analyzed independently as previously described (*see* **Note 14**).
3. Tools for ORFeome comparison. Considering that vast number of ORFeomes can be analyzed simultaneously by ANACONDA, we have included extra tools to allow comparative studies.
 - a. Data normalization. Because adjusted residuals are sensitive to ORFeome size and there is a large size difference between small bacterial and eukaryotic ORFeomes the software includes an option for size normalization that allows direct comparison of all sequenced ORFeomes of the three domains of life (*see* **Note 15**).
 - b. Comparing maps. ORFeome maps for two-codon-context bias can be compared in pairs using the Processing→Compare Genomes option. This tool will produce a differential display map that results from subtracting both maps cell by cell. Differential display maps can also be manipulated by the user as described for normal ORFeome maps.
 - c. Clustering. Alternatively, all opened maps can be compared in one single display to allow detecting overall patterns of two-codon context. This can be achieved with the option Processing→Compare all genomes. When this option is selected, ANACONDA will transform the 64×61 maps of each opened ORFeome into one single column of 3904 lines, one for each possible codon pair. In a second step, all columns are aligned set side by side to allow immediate comparison of patterns. As with all 64×61 maps, it is possible to rearrange this large-scale comparative map through cluster analysis of both axes to highlight major common patterns (**Fig. 3**).

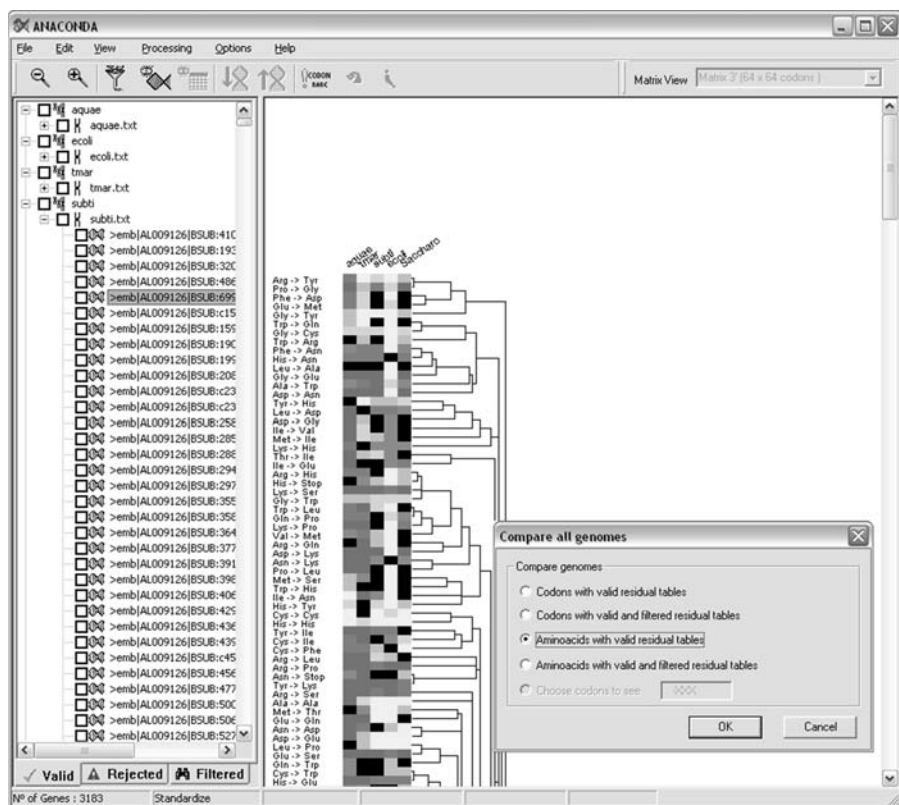


Fig. 3. Main window of the software package ANACONDA for two-codon context analysis at the ORFeome comparison level. When more than three ORFeomes are processed by ANACONDA it is possible to build a large-scale comparative map, as shown in the main right panel of the software's window. In this map, each column represents one ORFeome, with one line for all possible combinations of two consecutive codons. Visual comparison of different ORFeomes is possible if all ORFeomes are normalized to a given size and aligned using the same context order.

4. Exporting data. Similarly to the 64×61 maps, the adjusted residuals of large-scale comparison maps can be exported as CVS files for further mathematical analysis.

4. Notes

1. The contingency table is a 64×61 matrix. Because stop codons do not have codons on their 3'-side the three columns corresponding to these three codons are not defined.

2. For a more detailed description of FASTA format *see* www.ncbi.nlm.nih.gov/BLAST/fasta.html. As an example, the complete set of ORFs from a single species can be found in a format appropriate for ANACONDA in .ffn files of GenBank (<ftp://ftp.ncbi.nih.gov/genomes/>). If needed, this format must be applied to other sequences before opening them with ANACONDA.
3. In most cases, data presented by this software is calculated based on the ORF or ORF set selected in the left panel of the main window. If a special set of ORFs is to be analyzed it must be formatted as a FASTA file containing the chosen ORFs and then be opened by ANACONDA at later stage.
4. In the maps of two-codon context created by ANACONDA the rows represent fixed codons as indicated on the left, whereas the columns correspond to codons standing on the 5'- or 3'-sides of the fixed codon, as indicated at the top of the map. The type of context (5'- or 3'-side), as well as the type of map (showing codons, amino acids or nucleotide positions), can be chosen using a drop-down menu on the top-right corner of the main window.
5. Rare codons are highlighted by a blue circle in the sequence view layout, and will be considered in future versions of ANACONDA as codons to be preferentially optimized.
6. Usually, the last position of one codon (N3) is highly correlated with the first position of the following one (N4), as seen by the formation of single color larger squares in the maps (23).
7. By default, when opening a new set of DNA sequences the software will quantify them, i.e., will count codon pairs and calculate the adjusted residuals. However, sequences can be opened without quantification to be analyzed with residuals calculated with other sequences. This can be achieved by simply choosing the "No quantification" option of the Processing window.
8. A sequence that has been opened with no quantification can be analyzed with residual data extracted from other Orfeomes. For this, the user must select the sequence using the hierarchical left panel and click on its name with the right mouse button. Then the option "redirect" must be selected, as well as the genome whose residual data is to be used. The sequence will then appear at the gene view layout, colored as if it belonged to the host genome.
9. The header of the gene view layout includes: (1) the ORF name, (2) the total number of codons of that ORF, (3) the number of codons whose frequency is below the chosen threshold for rare codons, (4) the percentage of rare codons in the ORF, (5) the type of map and how data was quantified to reach the residuals used, and (6) the count and the percentage of two-codon contexts whose calculated residues belong to each color of the scale shown in the layout. Additionally, ANACONDA allows counting the total number of particular codons, as specified in the gene view options.
10. Alternatively, the same information can be obtained using the "i" button of the toolbar. Also, the option View→Gene (Nc, total GC, GC3, CAI) offers a reduced

version of the same information but in a floating window, that allows selecting different ORFs without closing it.

11. The CAI value for the selected sequence will appear only when the RSCU data for reference genes have been typed in. This has to be done manually, choosing Add in the window for defining RSCU values of the Options→Define RSCU Values menu. Each set of RSCU values can be saved for later use. To define the RSCU values of a genome, right button click in the genome name and choose RSCU values: set RSCU values.
12. Some filter tools include an option to visualize histograms showing how the variable is distributed across the entire ORF set. For example, to search for a set of ORFs with more than 10% of bad codon context the filter window should be open (either in the “Processing” menu or using the button at the toolbar). Then the option “Ratio” should be selected and the filter for “Residual Values” enabled. After choosing the degree of two-codon rejection to search for (according to the residual intervals chosen), and defining the search threshold at 10%, the filter should be run. The same filter can be used in several ORFeomes. However, each time a filter is run a new set of filtered ORFs will be displayed in the “Filtered” left panel, eliminating the previously displayed ones.
13. Workspaces can be named by the user and saved at any location in the file system.
14. Some windows allow selecting the ORFeome to be analyzed, through a scroll-down menu located in a field called “genome.” Usually, the default ORFeome is the first one that was opened, and attention must be taken to change this selection to analyze the intended ORFeome.
15. The adjusted residuals are corrected as if all ORFeomes had the same size, which can be fixed by the user in the Option→Standardize.

Acknowledgments

This study was supported by FCT/FEDER project grant REF: POCI/BIA-MIC/55466/04. GM is supported by FCT (SFRH/BPD/7195/2001). MASS is an EMBO YIP and his work is supported by the FCT/POCI program and the Human Frontier Science Program (Grant RGP45/2005). AVF is member of the R&D Unit “Matemática e Aplicações,” University of Aveiro (through POCTI/FCT, cofinanced by FEDER).

References

- 1 Ogle, J. M. and Ramakrishnan, V. (2005) Structural insights into translational fidelity. *Annu. Rev. Biochem.* **74**, 129–177.
- 2 Irwin, B., Heck, J. D., and Hatfields, W. G. (1995) Codon pair utilization biases influence translational elongation step times. *J. Biol. Chem.* **270**, 22, 801–22, 806.
- 3 Young, E. T., Sloan, J. S., and Riper, K. V. (2000) Trinucleotide repeats are clustered in regulatory genes in *Saccharomyces cerevisiae*. *Genetics* **154**, 1053–1068.

- 4 Borstnik, B. and Pumpernik, D. (2002) Tandem repeats in protein coding regions of primate genes. *Genome Res.* **12**, 909–915.
- 5 Karlin, S., Brocchieri, L., Bergman, A., Mrazek, J., and Gentles, A. J. (2002) Amino acid runs in eukaryotic proteomes and disease associations. *PNAS* **99**, 333–338.
- 6 Flis, K., Hinzpeter, A., Edelman, A., and Kurlandzka, A. (2005) The functioning of mammalian CIC-2 chloride channel in *Saccharomyces cerevisiae* cells requires an increased level of Kha1p. *Biochem. J.* **390**, 655–664.
- 7 Folley, L. S. and Yarus, M. (1989) Codon contexts from weakly expressed genes reduce expression in vivo. *J. Mol. Biol.* **209**, 359–378.
- 8 Cliften, P., Fulton, R., Wilson, R., and Johnston, M. (2006) After the duplication: gene loss and adaptation in *Saccharomyces* genomes. *Genetics* **172**, 863–872.
- 9 Van de Lagemaat, L. N., Gagnier, L., Medstrand, P., and Mager, D. L. (2005) Genomic deletions and precise removal of transposable elements mediated by short identical DNA segments in primates. *Genome Res.* **15**, 1243–1249.
- 10 Lin, Y. W., Thi, D. A. D., Kuo, P. L., et al. (2005) Polymorphisms associated with the DAZ genes on the human Y chromosome. *Genomics* **86**, 431–438.
- 11 Chen, S. L., Lee, W., Hottes, A. K., and McAdams, H. H. (2004) Codon usage between genomes is constrained by genome-wide mutational processes. *Proc. Natl. Acad. Sci. USA* **101**, 3480–3485.
- 12 Berg, O. G. and Silva, P. J. (1997) Codon bias in *Escherichia coli*: the influence of codon context on mutation and selection. *Nucleic Acids Res.* **25**, 1397–1404.
- 13 Akashi, H. (1994) Synonymous codon usage in *Drosophila melanogaster*: natural selection and translational accuracy. *Genetics* **136**, 927–935.
- 14 Percudani, R. and Ottonello, S. (1999) Selection at the wobble position of codons read by the same tRNA in *Saccharomyces cerevisiae*. *Mol. Biol. Evol.* **16**, 1752–1762.
- 15 Boycheva, S., Chkodrov, G., and Ivanov, I. (2003) Codon pairs in the genome of *Escherichia coli*. *Bioinformatic* **19**, 987–998.
- 16 Shah, A. A., Giddings, M. C., Parvaz, J. B., Gesteland, R. F., Atkins, J. F., and Ivanov, I. P. (2002) Computational identification of putative programmed translational frameshift sites. *Bioinformatics* **18**, 1046–1053.
- 17 Fedorov, A., Saxonov, S., and Gilbert, W. (2002) Regularities of context-dependent codon bias in eukaryotic genes. *Nucleic Acids Res.* **30**, 1192–1197.
- 18 Duan, J. and Antezana, M. A. (2003) Mammalian mutation pressure, synonymous codon choice, and mRNA degradation. *J. Mol. Evol.* **57**, 649–701.
- 19 Sharp, P. M. and Li, W. H. (1987) The codon adaptation index: a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res.* **15**, 1281–1295.
- 20 Haberman, S. J. (1973) The analysis of residuals in cross-classified tables. *Biometrics* **29**, 205–220.
- 21 Simonoff, J. (2003) *Analyzing Categorical Data*. Springer-Verlag, New York.

- 22 Everitt, B. S., Landau, S., and Leese, M. (2001) *Cluster Analysis*. Hodder Arnold, London, UK.
- 23 Moura, G., Pinheiro, M., Silva, R., et al. (2005) Comparative context analysis of codon pairs on an ORFeome scale. *Genome Biol.* **6**, R28.
- 24 Wright, F. (1990) The 'effective number of codons' used in a gene. *Gene* **87**, 23–29.

Comparative Analysis of RNA Genes

The caRNAC Software

Hélène Touzet

Summary

RNA genes are ubiquitous in the cell and are involved in a number of biochemical processes. Because there is a close relationship between function and structure, software tools that predict the secondary structure of noncoding RNAs from the base sequence are very helpful. In this article, we focus our attention on the inference of conserved secondary structure for a group of homologous RNA sequences. We present the caRNAC software, which enables the analysis of families of homologous sequences without prior alignment. The method relies both on comparative analysis and thermodynamic information.

Key Words: RNA; *in silico* folding; structure prediction; comparative analysis; thermodynamic model.

1. Introduction

It is now well-acknowledged that noncoding RNAs play an essential role in many cellular processes (e.g., protein synthesis, regulation), even if the function of the majority of RNAs remains to be elucidated ([1](#)). Many of noncoding RNAs have characteristic secondary structures that are highly conserved in evolution. Identifying conserved structure is the first step toward the comprehension of the function of the molecule. Computational approaches provide unexpansive and efficient tools for that purpose.

From a historical perspective, there are two main complementary approaches to address RNA folding prediction: thermodynamic models and phylogenetic models.

The secondary structure of an RNA molecule depends of the formation of basepairings: Watson-Crick (A-U and G-C), wobble (G-U), or even noncanonical pairings. In the thermodynamic approach, the fundamental assumption is that the molecule adopts a globally minimum free energy structure. Negative stabilizing energies are assigned to the stacking of basepairs in helices and destabilizing energies are assigned to unpaired elements, such as bulge or multibranched loops. In this model, the folding problem amounts to searching for the set of base pairs that minimizes the free energy level (2). This strategy is implemented in the Mfold (3,4) and RNAfold (5) programs. The main limitation of these methods is that the right structure may be overwhelmed by a large number of potential structures having equivalent, or even better, energy level. Furthermore, there is presently no thermodynamic parameters that deal with pseudoknots within this model.

The other line of research for structure inference is phylogenetic analysis. The main idea of this approach is to extract information from the similarities and differences between different, but homologous, RNA sequences. Phylogenetic analysis relies on the assumption that the spatial structure of a molecule is more highly conserved than is its sequence. In other words, the sequence is free to change during evolution. In terms of secondary structure, this means that mutation of a base involved in a pairing should generally be compensated by a change in its pairing partner. This guarantees that the ability of the both bases to form isosteric basepairs is retained. This phenomenon is called covariation, or compensatory mutation. If sufficient numbers of sequences are available, these covariations can be identified statistically directly from a multiple sequence alignment. The list of structures determined by comparative analysis is long: ribosomal RNAs, transfer RNAs, RNase P RNAs, HACA box RNAs, snoRNAs, and so on. (6). The drawback of pure phylogenetic approaches is that they need a large number of related sequences (more than 10) to be theoretically sound. Furthermore, the accuracy of the result strongly depends on the quality of the multiple alignment. Automatically aligning RNA sequences is a difficult issue (7).

The purpose of the caRNAc software is to achieve more flexibility than pure comparative methods by combining both thermodynamic and phylogenetic information. caRNAc does not require any prior alignment between sequences. This implies that it can successfully handle sequences with low level of conservation (from 60 %). The full algorithm is described in more detail in refs. 8 and 9). A comprehensive comparison of main folding programs, including caRNAc, can be found in ref. 10.

2. Materials

caRNAC is available on a website. All that is needed is a W3C compliant web browser (Firefox, Internet Explorer, Mozilla, and so on).

Frequent users also may download the platform and install it locally. CaRNAC requires a C compiler. All source codes are available on the website.

3. Method

3.1. Getting Started

The website is accessible at <http://bioinfo.lifl.fr/carnac>. Choose the “web server” section in the main menu. The “examples” section provides several data sets and commented results.

The input submission form contains three main fields.

1. Enter a name for the sequence (optional). This name serves as a label for the output page.
2. Enter the RNA sequences. The data set should include at least two distinct RNA sequences, and sequences should be in FASTA format. A sequence in FASTA format consists of a single-line description, followed by lines of sequence data. The first character of the description line is a greater than (“>”) symbol in the first column. **Figure 1** gives an example with three tRNA sequences, that will serve as a guideline for the remaining of this presentation. All nonalphabetic characters are removed. IUPAC symbols are not supported. Sequences may be pasted or uploaded from a file.
3. Enter an e-mail address. This address is used to send the identifier of the job to the user once the job is completed.

```
>Bacteriophage T4 Thr-tRNA
GCUGAUUUAGCUCAGUAGGUAGAGCACCUCACUUGUAAUGAGGAUGUCGGCGGUUCGAUUCCGUCAAUCAG
CA
>Yeast (S.cerevisiae) mitochondrial Phe-tRNA
GCUUUUUAUAGCUUAGUGGUAAAGCGAUAAAUUGAAGAUUUUUUACAUGUAGUUCGAUUCUAUUAAGGGC
A
>Halobacterium volcanii Phe-tRNA.
GCCGCCUUAGCUCAUACUGGGAGAGCACUCGACUGAAGAUCGAGCUGUCCCCGGUUCGAAUCCGGGAGGCG
GCA
```

Fig. 1. Three tRNA sequences in FASTA format.

The form then proposes several parameters that determine the final predictions. The default values lead to the most reliable results in average.

4. Eliminate redundant sequences. By default, caRNAc discards sequences that are too close (more than 98 % of identity). Uncheck the box to fold all the sequences.
5. Take GC content into consideration. When this option is selected, caRNAc uses variable energy thresholds for stems according to the average GC percent of the involved sequence.
6. Allow isolated stems. When this option is selected, caRNAc permits the creation of stem in one sequence alone, without any counterpart in any other sequence. This option may give better results if there is a large evolutionary distance between structures, or when the sequences are of radically different lengths. But it is time and space consuming. So it should be selected with caution.

The folding is launched by pressing the “RUN” button. Each job is assigned a unique identifier (ID). The computation of the putative foldings ranges from a few seconds for short sequences (less than 300 nt) to several minutes for longer sequences (up to 2000 nt). When the job is completed, the results are displayed on a new page, and an alert e-mail is sent to the user. Results are available for 24 h and may be retrieved with the ID using the “retrieve a result with an ID” section in the main menu.

3.2. Output Page

For each sequence, the predicted secondary structure is given in five formats, which are summarized in [Fig. 2](#). Note that all structures need not to be identical: the program is robust to minor variations in the structure between the sequences.

1. Connect notation (ct): it provides a textual description of the basepairings. The syntax is as follows: columns 1, 3, 4, and 6 redundantly give sequence indices, column 2 gives the sequences and column 4 gives “j” in position “i” if “(i,j)” is a basepair, otherwise this is zero. The heading of the file contains the size of the sequence and its name (found in the FASTA sequence).
2. Jpeg file: this file is generated from the CT file using the freely distributed drawing tool Naview ([11](#)). It contains a graphical two-dimensional representation of the secondary structure.
3. Postscript file: this is a conversion of the Jpeg file to the postscript format. This format is ready-to-print.
4. List of constraints: this text file gives an equivalent formulation of the structure. Each line contains the specification of one stem: “F i j k” means that there is an helix of length k formed between the positions [i,i+k-1] and [j-k+1,j]. This format is useful to specify a list of initial constraints for the Mfold and Kinefold programs (see **Note 1**).

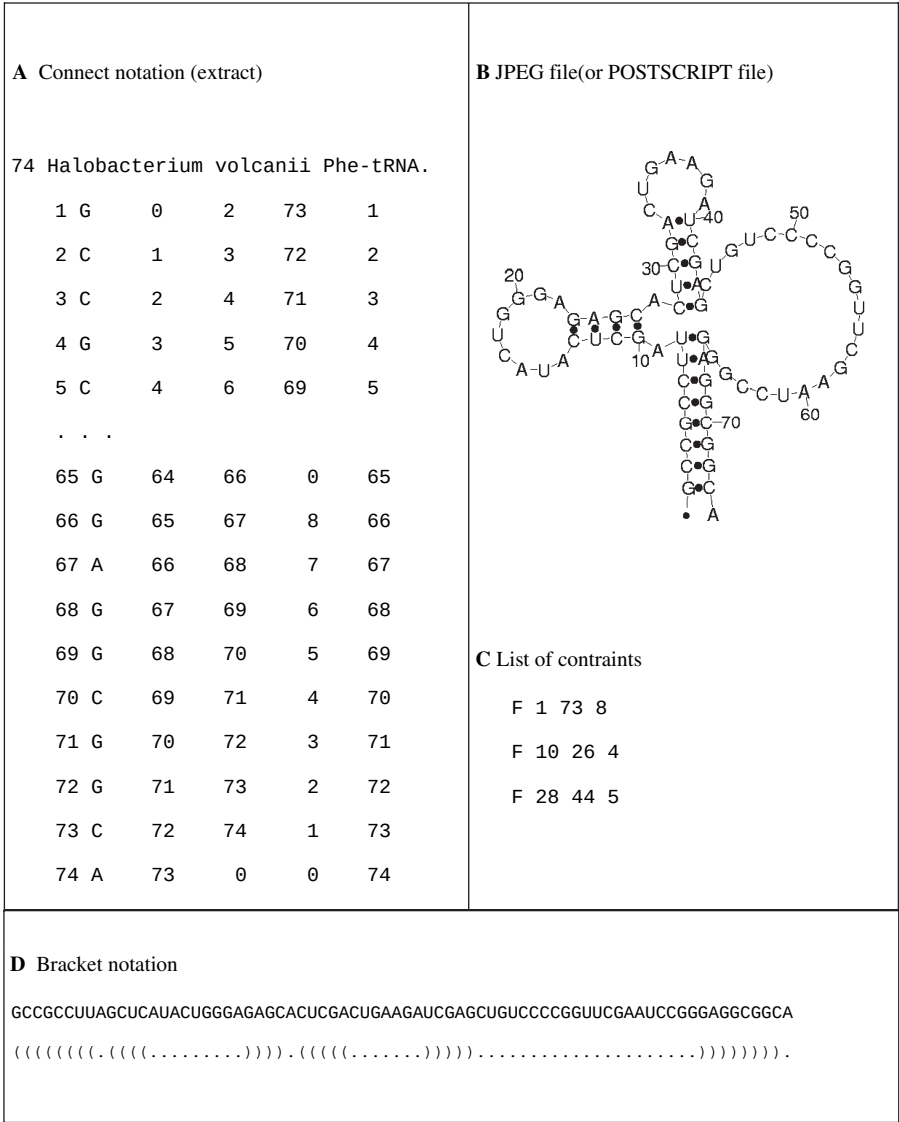


Fig. 2. Example of output formats for the structure predicted for the third tRNA sequence of Fig. 1. This structure is composed of three helices.

5. Bracket notation: it consists of two lines. The first line contains the sequence. The second line contains the set of associated pairings encoded by brackets and dots. A basepair between base “i” and “j” is represented by a “(” at position “i” and a “)”

at position “j.” Unpaired bases are represented by dots. The lack of pseudoknots in the secondary structure ensures that this notation defines a unique folding. This format is widely used in the Vienna Package (5).

If no structure is detected then the message “No structure found” is displayed. The first explanation is that the sequences actually do not share a common structure. Unfortunately, there are other cases where caRNAC fails to infer correctly the structure. We shall see it in the next section (*see* **Notes 2–4**).

6. RNAfamily (button “Visualize all foldings with RNAfamily”) allows the user to display all foldings at once. RNAfamily is a JAVA applet that is devoted to the visualization of multiple RNA sequences. It creates a plot using linear backbone representation. This is a concise representation that makes it convenient to compare several related structures at a glance. Each class of equivalent helices is assigned a color. RNAfamily includes the following functionalities: zooming, scrolling, selecting a stem, displaying the nucleotidic content. **Figure 3** gives a snapshot of RNAfamily.

It is also possible to download an archive storing all result files.

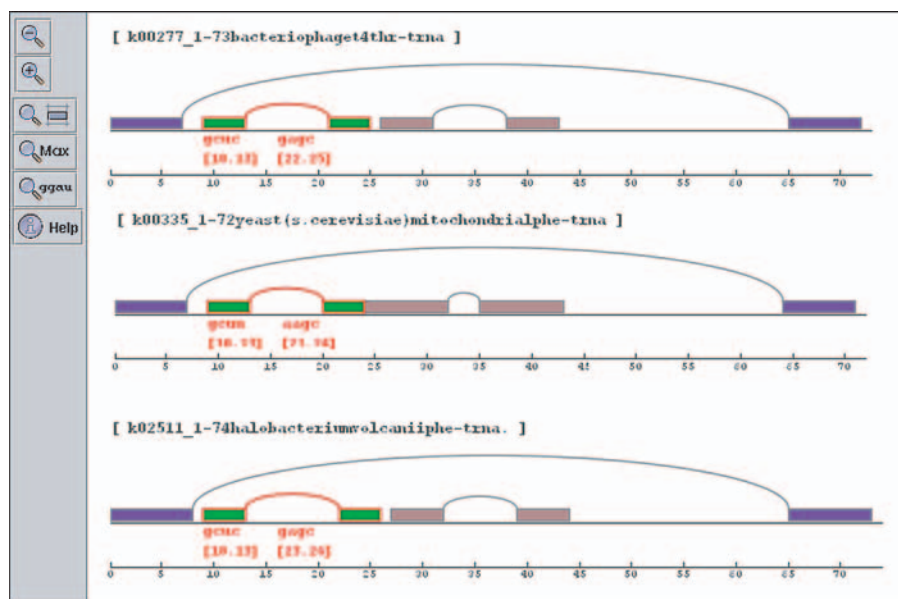


Fig. 3. Snapshot of RNAfamily. It shows the common structure for the three tRNA sequences of **Fig. 1**. Clicking on a stem displays the nucleotidic content of the stem (here the green stems). Clicking on “ggau” in the left menu displays the nucleotidic content of all sequences.

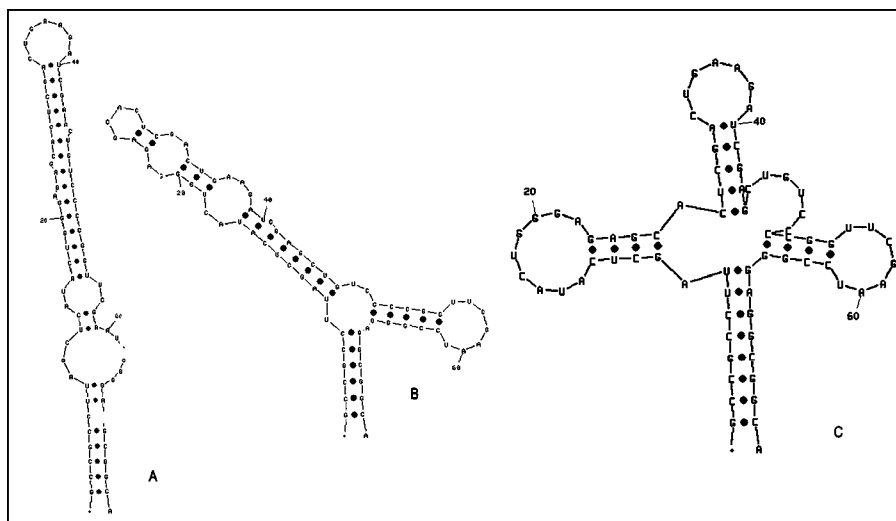


Fig. 4. Example of combination of caRNAc and Mfold. The first two structures (A,B) are the best two results given by Mfold alone for the third tRNA sequence of Fig. 1. The last structure (C) is obtained with Mfold using constraint information produced by caRNAc (file C in Fig. 2). In this case, Mfold correctly completes the structure and identifies the fourth stem that is missing in caRNAc output. This leads to the typical clover leaf structure (the acceptor stem is on the top).

4. Notes

We give here a list of pitfalls and limitations of the method. When possible, we suggest alternative programs that may prove to be more appropriate within the context. We also give further hints and rules of thumbs to maximize information from caRNAc output.

1. The predicted structure contains large unpaired regions.

The philosophy of caRNAc is to privilege selectivity to sensibility. So it may happen that the prediction misses some stems. But these stems may be recovered afterward with external programs, such as Mfold. This is the case of the structure inferred for the three tRNA sequences (we choose this example on purpose). On the one hand, the basepairs inferred by caRNAc are globally correct, but there is obviously one stem missing to form the cloverleaf structure (Fig. 2). This corresponds to the loop from position 45 to 65. On the other hand, the results obtained with Mfold alone are very poor on that data set (Fig. 4A, B). But combining caRNAc and then Mfold gives a better result (Fig. 4C). For the combination of caRNAc and Mfold, download all results using the constraint format, and paste it in the Mfold web server in the box “constraint information.” This opportunity is also especially attractive with the

kinetic-based Kinefold program that allows for pseudoknots (14). Kinefold supports the same format for the list of constraints.

2. The evolutionary distance is too small (more than 95 % identity).

The foundation of comparative analysis is that basepairings should be supported by compensatory mutations. It means that caRNAc is unlikely to find a complete structure if the sequences are very similar, because of the lack of mutations. In this context, it is wishable to use alternative tools that derive a consensus structure from an alignment. For example, RNAalifold (12) is a good alternative to caRNAc for very similar sequences. The initial multiple alignment can be built with ClustalW (13). Another possibility is to enrich the data set with new sequences at greater evolutionary distance, using similarity searching programs.

3. The evolutionary distance is too high (less than 50 % identity).

In this case, caRNAc is not guaranteed to recover a consensus structure because the search space is too wide. The solution here is to select few sequences with a higher conservation rate, if possible. As far as we know, no other program currently deals with such divergent sequences.

4. The structure may contain pseudoknots.

The algorithm of caRNAc is not designed for handling pseudoknots. If sequence are short (less than 70 bases), it might be a major source of error. In this particular context, it is more advisable to use a comparative pseudoknot-friendly program, such as comRNA (15). Note that comRNA is a time and space consuming program compared to caRNAc. It is limited to smaller data sets. For longer sequences, pseudoknots are usually not a problem. Kinefold may be used afterward to complete the structure and identify potential pseudoknots (we already mentioned this opportunity in **Note 1**).

5. Building an alignment with the structures obtained by caRNAc.

The multiple alignment tool allows the user to derive a structural alignment, taking into account both primary and secondary structures, from caRNAc output.

6. Discovering if the structure furnished by caRNAc is accurate.

Of course, the accuracy rate of caRNAc is not 100 %. Benchmark data show that predicted stems are usually correct, as soon as the number of stems is high enough to form a robust structure (like in **Fig. 2**). In this context, some rare missing stems may be recovered afterward (*see Note 1*). The situation is more complex with sparse structures containing mostly unpaired regions. It is a difficult task to decide if the stems actually exist or if they are false-positives occurring by chance. One solution is to compare the energy level of the sparse structure given by caRNAc with randomized equivalent data sets generated with shuffle-aln.pl (12). If the free energy is significantly lower with the initial data set, sequences are likely to share a common structure. We plan to integrate this functionality in the web server in the very near future.

The carRNAc website is under constant development. If there are any questions, please contact the authors at carnac@lifl.fr.

References

- 1 Eddy, S. R. (2001) Non-coding RNA genes and the modern RNA world. *Nat. Rev. Gen.* **2**, 919–929.
- 2 Eddy, S. R. (2004) How do RNA folding algorithms work. *Nat. Biotechnol.* **22**, 1457–1458.
- 3 Zuker, M. (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.* **31**, 3406–3415.
- 4 Zuker, M., Mathews, D. H., and Turner, D. H. (1999) Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide, in *RNA Biochemistry and Biotechnology*, (Barciszewski, J. and Clark, B.F.C., eds.), Kluwer Academic Publishers, Dordrecht/Norwell, MA.
- 5 Hofacker, I. L. (2003) Vienna RNA secondary structure server. *Nucleic Acids Res.* **31**, 3429–3431.
- 6 Brown, J. W. and Ellis, J. C. (2005) Comparative analysis of RNA secondary structure: the 6S RNA, in *Handbook of RNA Biochemistry*, (Bindereif, A., Hartmann, R., Schön, A., and Westhof, E., eds.), Wiley-VCH, Weinheim, Germany.
- 7 Gardner, P., Wilm, A., and Washietl, S. (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res.* **33**, 2433–2439.
- 8 Perriquet, O., Touzet, H., and Dauchet, M. (2003) Finding the common structure shared by two homologous RNAs. *Bioinformatics* **19**, 108–116.
- 9 Touzet, H. and Perriquet, O. (2004) CARNAC: folding families of non coding RNAs. *Nucleic Acids Res.* **142**, W142–W145.
- 10 Gardner, P. and Giegerich, R. (2005) A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics* **5**, 140.
- 11 Brucoleri, R. and Heinrich, G. (1988) An improved algorithm for nucleic acid secondary structure display. *Comput. Appl. Biosci.* **4**, 167–173.
- 12 Hofacker, I. L., Fekete, M., and Stadler, P. F. (2002) Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.* **319**, 1059–1066.
- 13 Higgins, D., Thompson, J., Gibson, T., Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive-multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 14 Xayaphoummine, A., Bucher, T., and Isambert, H. (2005) Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots, *Nucleic Acid Res.* **33**, 605–610.
- 15 Ji, Y., Xu, X., and Stormo, G. D. (2004) A graph theoretical approach for predicting common RNA secondary structure motifs including pseudoknots in unaligned sequences. *Bioinformatics* **20**, 1591–1602.

Efficient Annotation of Bacterial Genomes for Small, Noncoding RNAs Using the Integrative Computational Tool sRNAPredict2

Jonathan Livny

Summary

sRNAs are small noncoding RNAs that have been shown to perform diverse regulatory roles in a number of prokaryotes. Although several bioinformatic approaches have proven effective in identifying bacterial sRNAs, implementing these approaches presents significant computational challenges that have limited their use. To address these computational challenges, the author has developed and made publicly available sRNAPredict2, a program that facilitates the efficient prediction of putative sRNA-encoding genes in the intergenic regions of bacterial genomes. sRNAPredict2 identifies putative sRNAs by integrating genome-wide predictions of several different genetic features that are commonly associated with sRNA-encoding genes and identifying instances in which these features are colocalized in intergenic regions of the genome.

Key Words: sRNAs; sRNAPredict2; bioinformatics; annotation.

1. Introduction

sRNAs are small, noncoding RNA species that have been shown to regulate diverse cellular processes in a number of prokaryotes ([1,2](#)). Most sRNAs are 100–200 nt in length and, thus, are difficult to identify by traditional functional approaches such as random transposon mutagenesis. Furthermore, many sRNAs are significantly less abundant than other RNA species such as rRNAs, making their physical isolation from total cellular RNA difficult. Finally, because they do not encode proteins, sRNAs are difficult to identify

based on their primary sequence. Owing to the challenges of identifying sRNAs by traditional functional and bioinformatic approaches, until several years ago only 10 bacterial sRNAs had been identified. These bacterial sRNAs had been discovered serendipitously, as a result of their high cellular abundance, their inadvertent insertion into a multicopy plasmid along with other genes of interest, or their detection in the study of operons (3).

In 2001, the development of several bioinformatic approaches ushered in a new era in the study of bacterial sRNAs, one in which fortuitous discovery was replaced by accurate genome-wide predictive searches (4). In these computational studies, putative sRNAs encoded in the intergenic regions (IGRs) of the *Escherichia coli* genome were predicted based on the colocalization of genetic features commonly associated with previously characterized sRNAs. These features included predicted Rho-independent transcriptional terminators, putative promoters, sequences conserved among closely related species, and regions predicted to encode conserved secondary structure. A number of the putative sRNAs predicted in these studies were subsequently confirmed by Northern analysis. Of the approx 120 bacterial sRNAs known to date, the large majority has been identified using one or more of these integrative bioinformatic approaches.

Although bioinformatic approaches have proven successful in identifying *E. coli* sRNAs, their implementation has presented significant computational challenges that have severely limited their utilization. A genome wide search for putative sRNAs can include tens of thousands of individual predictive features such as terminators, promoters, and regions of conserved sequence. Integrating these thousands of individual features to identify the instances in which they are colocalized in IGRs has, in the past, required either the use of inefficient noncomputational methods, severely limiting the rate at which searches could be conducted, or the *de novo* development of a computational tools, necessitating a level of computer expertise not possessed by most biological researchers. Thus, although it is widely accepted that sRNAs are encoded by most if not all prokaryotes, until recently genome-wide annotations of sRNAs had been conducted in only 3 of the over 270 sequenced bacterial species.

To facilitate the efficient annotation of bacterial sRNAs, the author developed and made publicly available a program called sRNAPredict that flexibly integrates different combinations of individual sRNA predictors to rapidly identify putative sRNA-encoding genes in the IGRs of any annotated bacterial genome (5). By searching for putative transcriptional terminators encoded downstream of regions of sequence conservation, sRNAPredict identified 104 candidates for novel sRNAs in *Vibrio cholerae* IGRs. Nine

predicted *V. cholerae* sRNAs were subjected to experimental verification by Northern analysis; five were confirmed. A similar search was conducted in the opportunistic Gram-negative pathogen *Pseudomonas aeruginosa*, leading to the identification of 34 previously unannotated putative sRNA-encoding genes (6,7). Of these, 31 were experimentally tested by Northern analysis; 20 were confirmed. In addition, we developed an improved version of sRNAPredict2, sRNAPredict2, which we used to identify more than 2700 previously unannotated putative sRNA-encoding genes in the genomes of 10 other species of bacterial pathogens (7).

An sRNAPredict2 search is conducted in four main stages (Fig. 1). In stage I, primary data files such as databases of annotated ORFs, tRNAs, rRNAs, previously annotated sRNAs, and riboswitches, as well as genome sequence files are obtained from websites such as NCBI and TIGR. In stage II, the IGRExtract program is used to convert the genome of the species of interest to a database of intergenic sequences. In addition, primary ORF databases are converted to a format that can be used by TransTerm in its prediction of intergenic Rho-independent terminators. In stage III, the individual predictive elements of sRNAs such as regions of conservation and putative terminators are identified using several publicly available UNIX-based programs. If a database of putative promoters or transcription factor binding sites (TFBSs) is available, this database must be converted to the appropriate format by the user. In stage IV, the names of the databases obtained in stage I and of the output files and databases created in stage III are entered into the Initial Input File along with the values of various search parameters. This Initial Input File is then entered into sRNAPredict2 and the predictive search is conducted, producing an annotated database of intergenic putative sRNA-encoding genes.

2. Methods

The methodology below describes how to conduct sRNAPredict2 searches on the Mac OS X operating system. Although sRNAPredict2 can be compiled and executed on other operating systems, some of the more detailed instructions (such as “drag and drop”) are based on Mac OS X and may not be applicable to other operating systems. The instructions are written with the assumption that the reader has some basic experience using Unix-based programs.

2.1. Downloading sRNAPredict2

The sRNAPredict2 directory can be downloaded at <http://www.tufts.edu/sackler/waldorlab/sRNAPredict.html> and contains the sRNAPredict2 source code file (sRNAPredict2.cpp), executable (sRNAPredict2), as well as sample

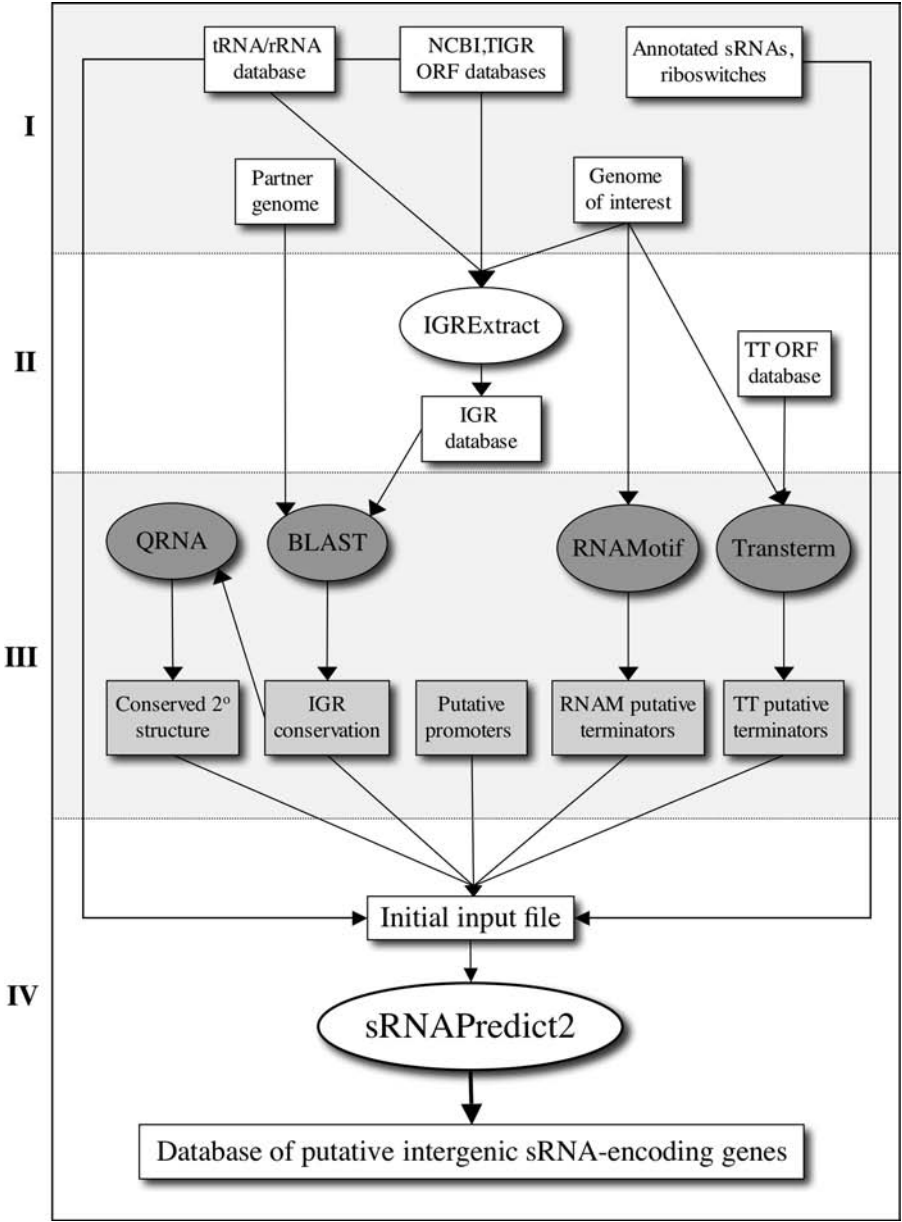


Fig. 1. Schematic of an sRNAPredict2 search for putative sRNAs using a single combination of predictive features. The number of each stage of the search is denoted on the left.

input and output files. These sample input files can be used to test sRNAPredict2 (see **Subheading 2.2.**). The source code for sRNAPredict2 is written in C++. The sRNAPredict2 executable was compiled on Mac OS X using the gcc compiler. To run sRNAPredict2 on a different operating system, the source code will need to be recompiled. Double-clicking the sRNAPredict2 icon should automatically open in the “Terminal” application on Mac OS X. If double-clicking on the executable does not launch sRNAPredict2, open a new terminal window, enter “chmod 777,” drag and drop the executable file, press return, then try to launch the executable again. If there is an error message while compiling the source code, enter “chmod 777” then drag and drop the source code file before attempting to recompile.

2.2. Testing sRNAPredict2

1. Ensure all sample input files are located in the same directory in which the sRNAPredict2 executable is located.
2. Launch sRNAPredict2 and at the “Enter the number of searches you would like to conduct:” prompt, enter 1.
3. At the “Enter the name of input file #1:” prompt, enter “Sample_initial_input.txt.”
4. The output file created will be named “Test_output.txt.” This file should be identical to file “Sample_output.txt” found in the sRNAPredict2 directory.

2.3. Stage I

2.3.1. Downloading Genome Sequence Files

1. From NCBI:
 - (a) Go to <ftp://ftp.ncbi.nih.gov/genomes/Bacterial/>.
 - (b) Follow the link for the species of interest and partner species.
 - (c) Download files with “.fna” extensions.
2. From TIGR:
 - (a) Go to <http://pathema.tigr.org/tigr-scripts/CMR/shared/Genomes.cgi>.
 - (b) Follow the green “T” link corresponding to the species of interest and partner species.
 - (c) Download files with “.lcon” extensions.
3. From the Sanger Institute:
 - (a) Go to (<http://www.sanger.ac.uk/Projects/Microbes/>).
 - (b) Follow the link for the species of interest and partner species.
 - (c) Follow the FTP link.
 - (d) Download files with “.dbs” extensions.

2.3.2. Downloading ORF Databases

sRNAPredict2 is designed to utilize NCBI and TIGR ORF databases. Databases available at NCBI contain the coordinate positions, strand orientations, and, usually, the locus and product names of all ORFs. They do not include annotated genes encoding frame-shift mutations. Databases available at TIGR contain the locus names and coordinates of all annotated ORFs and of annotated genes encoding frame-shift mutations. The strand orientations of ORFs in these databases are automatically inferred by sRNAPredict2 based on the order in which the start/end coordinates are listed. All ORFs found in the TIGR database that are not found in the NCBI database are assigned the locus and product name “TIGR.” TIGR ORF databases are not available for all sequenced genomes. If no TIGR database is available for the species of interest, sRNAPredict2 will utilize only the NCBI database.

1. Downloading ORF databases from NCBI:

- (a) Go to <ftp://ftp.ncbi.nih.gov/genomes/Bacterial/>.
- (b) Follow the link for the organism of interest.
- (c) Download file(s) with “.ptt” extensions (*see Note 1*).

2. Downloading ORF databases from TIGR:

- (a) Go to <http://pathema.tigr.org/tigr-scripts/CMR/shared/Genomes.cgi>.
- (b) Follow the green “T” link corresponding to the organism of interest.
- (c) Download file(s) with “.coords” extensions.

2.3.3. Downloading tRNA, rRNA Databases

1. Go to http://pathema.tigr.org/tigr-scripts/CMR/shared/MakeFrontPages.cgi?page=rna_list.
2. Follow the link of the organism of interest. This will go to a page that, in most cases, is split into three sections: tRNAs, rRNAs, and sRNAs.
3. Follow the “download” link found next to the heading of the tRNAs and rRNAs sections. This will open a new window containing the names and coordinates of putative tRNAs and rRNAs, respectively.
4. Download then copy/paste the database of tRNAs into an excel spreadsheet. Repeat with the rRNAs.
5. Delete all columns in the excel spreadsheet except those containing the coordinates of the tRNAs or rRNAs.
6. Copy/paste and save these coordinates in a text file.

2.3.4. Compiling Databases of Previously Annotated sRNAs and Riboswitches

1. Using Internet Explorer, go to http://www.sanger.ac.uk/cgi-bin/Rfam/genome_dist.pl (8) (see **Note 2**).
2. Follow the link for the species of interest.
3. Select all Rfam families that are not classified as tRNAs or rRNAs.
4. Click the “Selected families” button.
5. Copy/paste the list of putative sRNAs into a text file. Delete the headings.
6. To include genes not included in the Rfam database, add them to the end of this text file (see **Note 3**).

2.4. Stage II

2.4.1. Creating IGR Databases Using IGRExtract

In BLAST comparisons between the complete genomes of two closely related species, short stretches of intergenic sequence conservation are often not identified because of the relatively large amount of ORF conservation. Thus, to increase the sensitivity of BLAST searches for sRNA conservation, comparisons should be conducted using a database containing only the intergenic sequences of the genome of interest. To facilitate conversion of a genome sequence to a FASTA-formatted IGR database, the author has developed a C++ program named IGRExtract. The IGRExtract directory, which contains both the IGRExtract executable (compiled using gcc in Mac OS X) and the IGRExtract source code, is available for download at <http://www.tufts.edu/sackler/waldorlab/sRNAPredict.html>. Follow the instructions in **Subheading 2.1.** to download and launch IGRExtract (see **Note 4**).

1. IGRExtract takes as input (1) a file containing the genomic sequence (in FASTA format), (2) an ORF database(s) (NCBI and/or TIGR), and (3) a tRNA/rRNA database formatted as described above. When prompted, enter the name of these files (see **Note 5**).
2. Enter the size of the genome.
3. Enter how much sequence upstream/downstream of the IGRs are to be extracted. To extract only IGRs enter “0;” to exclude the ends of IGRs adjacent to their flanking ORFs, enter a positive number; to include ORF sequences flanking the IGRs, enter a negative number.
4. IGR sequences are extracted in FASTA format. Verify that each FASTA name in the output file begins with “*IG*” followed by the coordinates of the IGR flanked by parenthesis. This format is critical for conversion of IGR coordinates to chromosome coordinates by sRNAPredict2.

2.4.2. Creating a TransTerm-Compatible Input Files

Running the terminator-predicting program TransTerm requires 2 input files, one containing the genome sequence and the other containing ORF coordinates. The header of the genome sequence file must conform to format ">SpeciesName_id#." This header is followed by the genome sequence. The authors has developed a C++ program, TT_ORF, that converts NCBI or TIGR ORF databases to a TransTerm-compatible ORF database. This program is available for download at <http://www.tufts.edu/sackler/waldorlab/sRNAPredict.html>. Follow the instructions in **Subheading 2.1.** to download and launch TT_ORF. TT_ORF automatically assigns 1 as the id no. for all ORFs; thus if TT_ORF-produced databases are used, ensure 1 is entered as the id no. in the sequence file header.

2.5. Stage III

2.5.1. Identifying Regions of Intergenic Conservation Using BLAST

BLAST comparisons should be conducted using the WU-BLAST 2.0 program (9) (available at <http://blast.wustl.edu/licensing/>). BLAST comparisons should be conducted between the IGRs of the species of interest and either the entire genome or the IGRs of the partner species (*see Note 6*).

1. BLAST 2.0 supports databases in XDF (eXtended Database Format). Thus, before conducting a BLAST comparison, drag and drop the xdformat program icon (located in the BLAST 2.0 directory) into the Terminal window, type "-n," then drag and drop the IGR database file.
2. Once XDF formatting has been completed, drag and drop the blastn alias icon (located in the BLAST 2.0 directory) followed by the IGR database file and the genome sequence file or IGR database of the partner species into the terminal window. BLAST output file names can be assigned by entering "-o" followed by the desired output file name.

2.5.2. Creating Promoter/TFBS Databases

Promoters/TFBS databases must be formatted by the user. Each line should include only the start coordinate of the promoter or TFBS (*see Note 7*) and the strand orientation of the promoter or TFBS (0 for coding strand, 1 for noncoding strand).

2.5.3. Identifying Putative Rho-Independent Terminators Using RNAMotif

1. Download the RNAMotif directory by going to <http://www.scripps.edu/mb/case/casegr-sh-3.5.html> and following the rnamotif-3.0.4.tar.gz (10). The directory

contains a README file that includes instructions for installing, testing, and running RNAMotif.

2. Before running RNAMotif, the path to the directory containing the energy data must be specified. To do this, enter “setenv EFNDATA” then either manually enter the path to the efndata directory or drag and drop the directory into the terminal window.
3. To run RNAMotif, drag and drop the rnamotif executable icon (found in the src directory in the RNAMotif directory) into the terminal window. Enter “-descr,” then drag and drop the descriptor file followed by the genome sequence file (*see Note 8*).
4. The output of the RNAMotif search is written to the UNIX terminal window. Copy/paste this output from the terminal window to a text file. This output file can be used without modification as input in sRNAPredict2 searches.

2.5.4. Identifying Putative Rho-Independent Terminators Using TransTerm

sRNAPredict2 is designed to extract coordinates from TransTerm output files or, when available, from published TransTerm databases (*10,11*).

1. Go to <http://www.genomics.jhu.edu/TransTerm/transterm.html>. If a TransTerm database is available at for the species of interest, copy and paste the published database from Internet Explorer (*see Note 2*), and save it as a text file. The format of the database should conform to the format of the sample TransTerm database provided in the sRNAPredict2 directory.
2. If a TransTerm database is not available for the species of interest, TransTerm can be downloaded at <http://www.tufts.edu/sackler/waldorlab/sRNAPredict.html> and is accompanied by a README file that includes instructions for its installation (*see Note 9*).
3. To run TransTerm, first drag and drop the icon of the TransTerm executable (which is found in the src directory) into the terminal window.
4. Enter -s then drag and drop the genome sequence file (*see Subheading 2.3.1*).
5. Enter -c then drag and drop the formatted ORF database file (*see Subheading 2.3.1*).
6. Enter -o, the name to be assigned to the output file, then -g.
7. The TransTerm output file can be used without modification as input in sRNAPredict2 searches.

2.5.5. Identifying Regions of Predicted Conserved Secondary Structure Using QRNA

QRNA is a program that utilizes BLAST-generated sequence alignments to identify patterns of sequence homology that likely represent conservation of RNA secondary structure (*12*). A putative sRNA identified by sRNAPredict2

is reported to correspond to a region of conserved secondary structure (denoted by a “Y” in the “QRNA?” column of the output file) if that sRNA overlaps any region predicted by QRNA to encode conserved secondary structure (reported as “winner = RNA” in the QRNA output file).

1. Download the QRNA directory by going to <http://selab.wustl.edu/cgi-bin/selab.pl?mode=software#qrna>. The directory contains a PDF user guide (in the documentation directory) that includes instructions for installing and running QRNA.
2. Assign a location for the QRNA libraries by entering “setenv QRNADB” then dragging and dropping the lib directory into the terminal window (or manually entering the full location of the lib directory).
3. Before running a QRNA search, the BLAST output file must be converted to a QRNA input file. To do this, drag and drop the “blastn2qrnadept.pl” Perl file from the scripts directory followed by the BLAST output file (*see Note 10*) into the terminal window. These Perl scripts will create three new files in the scripts directory. The file with “.q” extension will serve as the input file for QRNA.
4. To run QRNA, drag and drop the icon of the QRNA Unix executable (located in the src directory) into the terminal window. Next, assign values for the window (-w) and slide (-x) parameters (*see Note 11*) and a name for the output file (-o). Finally, drag and drop the QRNA input file located in the scripts directory. Be aware that, depending on the size of the input file and the window and slide values, QRNA analysis may take many hours to be completed.
5. Once the QRNA analysis is completed, the output file can be used without modification as input in sRNAPredict2 searches. A sample QRNA output file (Sample_QRNA.txt) is provided in the sRNAPredict2 directory.

2.6. Stage IV

2.6.1. Creating the Initial Input File

The names of the primary data files, the desired name of the output file, and the values of various search variables are entered in sRNAPredict2 through an Initial Input File. If certain types of primary data files (such as a promoter database or a TIGR ORF database) are not to be included in the search, enter “none” in place of a file name. A sample Initial Input File is included in the sRNAPredict2 directory. This file was created with BBEdit (a program the author recommends over other text editing programs) and its format may be altered when opened with other text editing programs such as TextEdit. Thus if the Initial Input File cannot be opened with BBEdit, it should be opened using Word.

2.6.2. Running sRNAPredict2

1. Double-click on the icon of the executable.
2. The user will be prompted to enter the number of searches to conduct. Enter 1.
3. At the next prompt, enter the name of the initial input file (*see Note 12*).

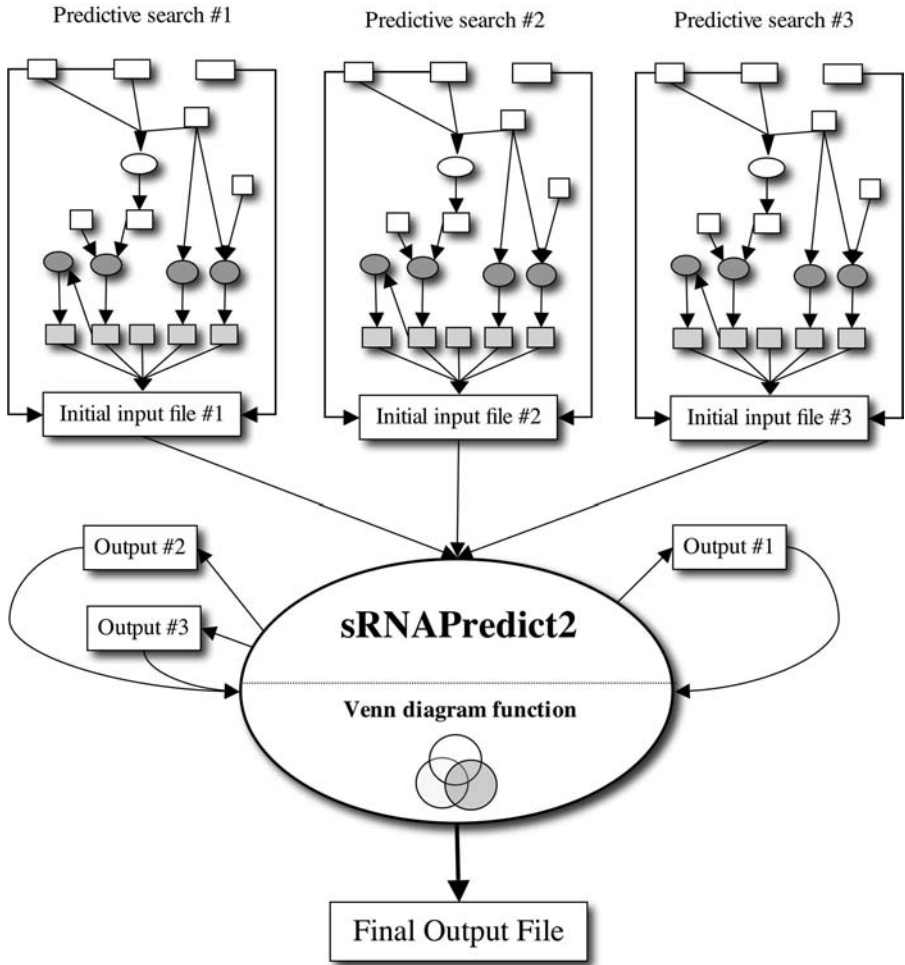


Fig. 2. Schematic of an sRNAPredict2 search for putative sRNAs using multiple combinations of predictive features. The Venn diagram function is automatically executed following the completion of all individual searches to identify sRNAs predicted in multiple independent searches.

2.6.3. Using the Venn Diagram Function of sRNAPredict2 to Identify Putative sRNAs Predicted in Multiple Independent Searches

sRNAPredict2 was designed with a Venn diagram function that allows putative sRNA-encoding genes that were predicted in multiple independent searches to be identified. This feature is particularly useful when searching for putative sRNA-encoding genes that are conserved in multiple species (*see Note 13*). The Venn diagram function is automatically executed when more than one Initial Input File is entered into sRNAPredict2 (**Fig. 2**).

1. Double-click on the icon of the executable.
2. Enter the number of searches to be conducted.
3. At the next prompt, enter the name of the first initial input file. Repeat this until all Initial Input File names have been entered.
4. After all Initial Input File names are entered, enter the name to be assigned to the Final Output File. This name will be assigned only to the Final Output File created by the Venn function. The output files of each of the independent searches will be named according to the names entered in their corresponding Initial Input File.

3. Notes

1. Remove all “*” from the “.ptt” file before running the search. This can be accomplished by simply using the find and replace all functions in Word or a text editing program.
2. Copying/pasting databases from other browsers such as Safari may change the format of the database. To ensure databases copied directly from the browser conform to a format compatible with sRNAPredict2, compare them to the corresponding sample databases included in the sRNAPredict2 directory.
3. Add annotated sRNAs or riboswitches using the following format:

```
“user  gene_name  start_coordinate  end_coordinate”
```

For example:

```
“user  tmRNA  102333  102469”
```

4. The IGRExtract directory includes several sample files that can be used to test the program. When testing the program, enter “2160837” when asked to enter the size of the chromosome and “0” for the amount of sequence upstream/downstream of the IGR to be extracted. Compare the output file to “Test_output.txt.”
5. If input files are located in the same directory as the executable, entering only the names of the files is sufficient. If not, the entire path name must be entered (either manually or by dragging and dropping the files into the UNIX terminal window).

6. Comparison between two IGR databases will take significantly longer than a comparison between an IGR database and a whole genome but the former may be more sensitive for identifying short stretches of sRNA homology than the latter. If the IGR database of the partner species (the query sequence in the BLAST comparison) was created using IGRextract, all “*” must be deleted or replaced from the FASTA names in this file before the BLAST comparisons are conducted (using the find and replace all functions in Word or a text editing program).
7. The start coordinate entered should correspond to the 3’ boundary of the predicted promoter or TFBS. This is strand specific, i.e., if a promoter is predicted from positions 10 to 20 on the + strand, enter 20; if predicted from 10 to 20 on the – strand, enter 10.
8. The format of the output file of RNAMotif depends on the descriptor file used in the search. sRNAPredict2 was designed to extract coordinates from RNAMotif searches which use a specific descriptor file provided by D. Ecker. This file, “RNAMotif_descr.txt,” is included in the sRNAPredict2 directory.
9. After compiling the source code using the “make” command, go into the TransTerm file (in the src directory) and change the line “\$path = ‘put a path here’;” to a path pointing to the directory in which the file now resides, for example “\$path = ‘/home/bob/favorite_programs/TransTerm/src;’.”
10. To ensure that the format of the QRNA output is compatible with sRNAPredict2, the BLAST comparison must be conducted as described in **Subheading 2.4.2.** with the species of interest entered first (as the subject sequence) and the partner species entered second (as the query sequence) in the command-line.
11. The author has found that a window size of 100 and a slide position of 50 provided the best results for identifying experimentally confirmed *P. aeruginosa* sRNAs but different values might yield better results in other species. Be aware that increasing the window size can significantly increase the time it takes to complete a QRNA search.
12. sRNAPredict2 creates a number of intermediate output files that are passed from one function to another during the sRNAPredict2 search. These will be overwritten every time the search is run unless removed from the directory in which sRNAPredict2 is located. Furthermore, if the name of the output file in the Initial Input File is not changed between searches, the previous output file is overwritten.
13. The Venn diagram function reports the total number of independent searches in which each predicted sRNA was identified. If each of the independent searches used conservation between the species of interest and a different BLAST partner as a predictor of sRNA-encoding genes, the name(s) of the BLAST partner(s) in which each sRNA was found to be conserved will be reported by the Venn diagram function in the Final Output File. The name assigned to each partner species in the Final Output File will be the name of the BLAST file OR the first word in the name of the BLAST file flanked by underscores. For example, if the

name of the BLAST file is “BaIGRBc0.txt” the BLAST partner name reported in the Final Output File will be “BaIGRBc.txt.” If the name of the BLAST file is “BaIGR_Bc_0_.txt,” the name in the Final Output File will be “Bc.”

References

- 1 Dennis, P. P. and Omer, A. (2005) Small non-coding RNAs in Archaea. *Curr. Opin. Microbiol.* **8**, 685–694.
- 2 Gottesman, S. (2005) Micros for microbes: non-coding regulatory RNAs in bacteria. *Trends Genet.* **21**, 399–404.
- 3 Gottesman, S. (2004) The small RNA regulators of *Escherichia coli*: roles and mechanisms. *Annu Rev Microbiol.* **58**, 303–328.
- 4 Hershberg, R., Altuvia, S., and Margalit, H. (2003) A survey of small RNA-encoding genes in *Escherichia coli*. *Nucleic Acids Res.* **31**, 1813–1820.
- 5 Livny, J., Fogel, M. A., Davis, B. M., and Waldor, M. K. (2005) sRNAPredict: an integrative computational approach to identify sRNAs in bacterial genomes. *Nucleic Acids Res.* **33**, 4096–4105.
- 6 Alifano, P., Rivellini, F., Limauro, D., Bruni, C. B., and Carlomagno, M. S. (1991) A consensus motif common to all Rho-dependent prokaryotic transcription terminators. *Cell* **64**, 553–563.
- 7 Livny, J., Brencic, A., Lory, S., and Waldor, M. K. (2006) Identification of 17 *Pseudomonas aeruginosa* sRNAs and prediction of sRNA-encoding genes in 10 diverse pathogens using the bioinformatic tool sRNAPredict2. *Nucleic Acids Res.* **34**, 3484–3493.
- 8 Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R. and Bateman, A. (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.* **33**, D121–D124.
- 9 Altschul, S. F., Madden, T. L., Schaffer, A. A., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.
- 10 Macke, T. J., Ecker, D. J., Gutell, R. R., Gautheret, D., Case, D. A., and Sampath, R. (2001) RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Res.* **29**, 4724–4735.
- 11 Ermolaeva, M. D., Khalak, H. G., White, O., Smith, H. O., and Salzberg, S. L. (2000) Prediction of transcription terminators in bacterial genomes. *J. Mol. Biol.* **301**, 27–33.
- 12 Rivas, E. and Eddy, S. R. (2001) Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics.* **2**, 8.

Methods for Multiple Alignment and Consensus Structure Prediction of RNAs Implemented in MARNA

Sven Siebert and Rolf Backofen

Summary

Multiple alignments of RNAs are an essential prerequisite to further analyses such as homology modeling, motif description, or illustration of conserved or variable binding sites. Beyond the comparison of RNAs on the sequence level, structural conformations determined by basepairs have to be taken into account. Several pairwise sequence-structure alignment methods have been developed. They use extended alignment scores that evaluate secondary structure information in addition to sequence information. However, two problems for the multiple alignment step remain. First, how to combine pairwise sequence-structure alignments into a multiple alignment and, second, how to generate secondary structure information for sequences whose structural information is missing. Here, we describe MARNA, its underlying methods and its usage. MARNA is an approach for multiple alignment of RNAs taking into considerations both the primary sequences and the secondary structures. It relies on the pairwise sequence-structure comparison strategy by generating a set of weighted alignment edges. This set is processed by a consistency-based multiple alignment method. Additionally, MARNA extracts a consensus-sequence and structure from this generated multiple alignment. MARNA can be accessed via the webpage <http://www.bioinf.uni-freiburg.de/Software/MARNA>.

Key Words: Multiple alignment; RNA; sequence structure; consensus structure.

1. Introduction

RNAs are nucleic acid polymers consisting of covalently bound nucleotides. RNA is primarily made up of four different bases: adenine, guanine, cytosine, and uracil. Single-stranded RNA molecules tend to form hydrogen bonds

resulting in spatial arrangements of these nucleotides. Many RNAs conserve a secondary structure of basepairing interactions more than they conserve their sequence. Since the discovery of RNAs that act as enzymes ([1](#)) and the detection of huge classes of noncoding RNAs involved in regulation processes, RNAs became more and more important. For the discovery of RNA classes, multiple sequence structure alignments are the best choice to detect RNAs with the same function. Furthermore, multiple alignments are an essential prerequisite to further analyses such as homology modeling, motif description, or illustration of conserved or variable binding sites.

Here, we want to focus on the concepts and the methods used in multiple alignment of RNAs (MARNA). MARNA is an approach to align multiple RNAs taking into consideration both the primary sequences and the secondary structures. It is based on pairwise sequence-structure comparisons of RNAs as proposed by [ref. 2](#). From these sequence-structure alignments, libraries of weighted alignment edges are generated. The weights reflect the sequential and structural conservation. For sequences whose secondary structures are missing, the libraries are generated by sampling low energy conformations. The libraries are then processed by a consistency-based multiple alignment method, which is implemented in the T-Coffee system ([3](#)). In addition, MARNA is able to extract a consensus-sequence and -structure from a multiple alignment.

Suppose that one has a set of RNA sequences provided with secondary structures. In summary, the coarse grain of the MARNA method is as follows:

1. Generate and weight alignment edges between pairwise RNAs reflecting sequence and structure similarities.
2. Collect all weighted edges in a so-called library. This library is processed by a consistency-based multiple alignment method.
3. Find a consensus-sequence and structure from this multiple alignment.

MARNA is able to align RNAs without known conformations as well. For these sequences, several methods to assign structures to the sequences exist. MARNA is capable of integrating these methods and, thus, to align RNAs with initially unknown structures.

In this work, we focus on the methods used in MARNA and give some hints about parameter settings that determine the alignments, and structure choices, especially when structures are missing for some sequences. For detailed comparison studies of MARNA with related multiple alignment tools, e.g., *see* [refs. 4 and 5](#).

2. Methods

2.1. Definitions

1. A sequence S is a word over the alphabet $\{A, C, G, U\}$. $S[i]$ denotes the i -th symbol in S .
2. An arc is a pair $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ s.t. $i < j$. i and j are the ends of the arc. An arc represents a basepair.
3. A base is called free, if it is not involved in any arc.
4. A secondary structure is a set of arcs P , s.t. for any two arcs $(i_1, j_1), (i_2, j_2) \in P$ with $i_1 < i_2$ either $i_1 < j_1 < i_2 < j_2$ or $i_1 < i_2 < j_1 < j_2$.
5. An RNA is a tuple (S, P) , where S is the sequence and P is the set of arcs in a secondary structure.
6. An alignment A of two RNAs (S_1, P_1) and (S_2, P_2) is a subset of $\{1, \dots, |S_1|\} \cup \{-\} \times \{1, \dots, |S_2|\} \cup \{-\}$, where for all pairs $(i, j), (i', j') \in A$ holds
 - a. $i \leq i' \Rightarrow j \leq j'$
 - b. $i = i' \neq - \Rightarrow j = j'$ and
 - c. $j = j' \neq - \Rightarrow i = i'$.

Requirement: for every $i \in \{1, \dots, |S_1|\}$ there is some j with $(i, j) \in A$ (and vice versa for $j \in \{1, \dots, |S_2|\}$).

7. The pairs $(i, j) \in A$ are called alignment edges.
8. An alignment edge is called realized if neither $i = -$ nor $j = -$.

2.2. Pairwise Alignment

The scoring of an alignment A of two RNAs (S_1, P_1) and (S_2, P_2) is based on the notion of edit operations on bases as well as on arcs. We recall the edit operations as given in **ref. 2** and present a slightly modified scoring scheme to finally compute an optimal alignment between two RNAs. Optimal means to find an alignment with minimum costs assuming that the costs of an alignment are composed of the costs of all executed edit operations.

2.2.1. Edit Operations

1. Edit operations on free bases are:
 - a. Base match: the base at position i in the first RNA is matched with the base at position j in the second RNA, i.e., $S_1[i] = S_2[j]$. The costs are 0.
 - b. Base mismatch: the base at position i in the first RNA is aligned with the base at position j in the second RNA s.t. $S_1[i] \neq S_2[j]$. The costs are positive.
 - c. Base deletion/insertion: the base at position i in the first RNA is aligned with a gap (deletion operation). The opposite case is the insertion operation. Both costs are positive.

2. Edit operations on arcs: consider an arc $(i, j) \in P_1$ such that i is aligned with i' and j is aligned with j' for $i', j' \in S_2 \cup \{-\}$.
 - a. Arc match: an arc match occurs if i', j' form an arc $(i', j') \in P_2$ and $S_1[i] = S_2[i']$ and $S_1[j] = S_2[j']$.
 - b. Arc mismatch: an arc mismatch occurs if i', j' form an arc $(i', j') \in P_2$ and $S_1[i] \neq S_2[i']$ or $S_1[j] \neq S_2[j']$.
 - c. Arc deletion: arc deletion means that $(i', j') \notin P_2$. Depending on how many gaps the two positions i', j' occupy, we may have
 - i. Arc breaking: an arc breaking occurs if none of i' and j' equals the symbol $-$.
 - ii. Arc altering: an arc altering occurs if exactly one of i' and j' equals the symbol $-$.
 - iii. Arc removing: an arc removing occurs if both i' and j' are equal to $-$.
3. Edit operation on arcs are depicted in **Fig. 1**. Arc costs are as follows:
 - a. An arc match has costs 0.
 - b. An arc mismatch operation has costs $W_{am}(i, j, i', j')$ for two arcs $(i, j) \in P_1$ and $(i', j') \in P_2$.
 - c. An arc deletion operation has costs $W_{ad}(i, j, i', j')$. These costs are determined by the bases and by the number of gaps involved. We decompose the costs $W_{ad}(i, j, i', j')$ into a sum of two single functions for the left and right ends of the arcs:

$$W_{ad}(i, j, i', j') = W_{ad}^l(i, j) + W_{ad}^r(i', j')$$

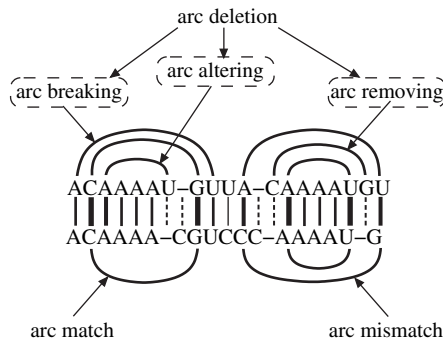


Fig. 1. An alignment of two RNAs with corresponding edit operations on arcs. Alignment edges are drawn as solid lines (realized edges) and dashed lines (nonrealized edges). The thickness of realized edges corresponds to similarity weights between bases. Nonrealized edges are skipped for the multiple alignment step.

In the following, we do not distinguish between left and right arc ends, and, thus, introduce the function $W_{ad}^e(i, j) = W_{ad}^l(i, j) + W_{ad}^r(i, j)$. We even simplify the scoring scheme further by defining $W_{ad}^e(i, j)$ to be composed of a base match, base mismatch, or base deletion together with a fixed cost for deleting an arc. Hence, we set

$$W_{ad}^e(i, j) = W_{base}(i, j) + \frac{1}{2} W_{ad}^{const},$$

where W_{ad}^{const} are the costs for deleting one arc.

2.2.2. Alignment Algorithm

In the following, we specify our algorithm similar to the one given in **ref. 2** that computes an optimal alignment between two RNAs with given secondary structures (S_1, P_1) and (S_2, P_2) . We introduce two simple functions:

$$\Psi_S(i) = \begin{cases} 1, & \text{if base at position } i \text{ not free} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\chi(i, j) = \begin{cases} 1, & \text{if } S_1[i] \neq S_2[j] \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Here, the costs for the edit operations on free bases base match, base mismatch, and base deletion are combined into a single cost function $W_{base}(i, j)$, where $W_{base}(i, j) = 0$ only if $S_1[i] = S_2[j]$. Now, we can specify the alignment algorithm:

Input: two RNAs (S_1, P_1) and (S_2, P_2) .

Output: sequence structure alignment.

Method: *ALIGN – RNAs*()

for $a_1 = (i_1, i_2) \in P_1$ and $a_2 = (j_1, j_2) \in P_2$
do for $i \leftarrow i_1 + 1$ to $i_2 - 1$
do for $j \leftarrow j_1 + 1$ to $j_2 - 1$
do

$$M(i, j) = \min \left\{ \begin{array}{l} M(i-1, j) + w_{base}(i, -) + \Psi_1(i) \frac{1}{2} w_{ad}^{const}, \\ M(i, j-1) + w_{base}(-, j) + \Psi_2(j) \frac{1}{2} w_{ad}^{const}, \\ M(i-1, j-1) + w_{base}(i, j) + (\Psi_1(i) + \Psi_2(j)) \frac{1}{2} w_{ad}^{const}, \\ M(i'-1, j'-1) + B(a_k, a_1) + (\chi(i', j') \cdot \chi(i, j)) w_{am}(i', i, j', j'), \\ \quad \text{if } a_k = (i', i) \in P_1 \text{ and } a_1 = (j', j) \in P_2 \end{array} \right.$$

$$B(a_1, a_2) = M(i_2 - 1, j_2 - 1)$$

1. We need two two-dimensional matrices, both not exceeding the size of nm . The matrix B contains the minimum costs of aligning the intervals $(i_1 + 1, i_2 - 1)$ and $(j_1 + 1, j_2 - 1)$ for arcs $a_k = (i_1, i_2) \in P_1$ and $a_l = (j_1, j_2) \in P_2$ provided that both arcs are aligned; i.e., we have an arc match or arc mismatch. The matrix M is constructed when the two arcs a_k and a_l are considered. It is computed within the arc intervals in almost the same manner as a sequence alignment except that arc breaking costs are considered and computed at each single base. The algorithm proceeds from inside to outside, thereby taking arcs with minimal sequence lengths first.
2. From the previously described algorithm it is easy to see that the time complexity of $O(n^2m^2)$ results from running over the arcs in both sequences and computing the best alignment in between. The space complexity is determined by the sizes of the two matrices B and M .
3. The resulting alignment can be obtained by a traceback step.

2.2.3. Alignment Weights

The alignment algorithm computes an alignment between two RNAs, which is equivalent to an edit transcript composed of edit operations weighted with edit costs. For the multiple alignment step, these costs have to be transformed into similarity weights.

1. Note that the costs are a function d with positive values fulfilling the metric conditions:
 - a. $d(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$, i.e., the costs of two RNAs S_1 and S_2 are 0 if and only if the two RNAs are equal.
 - b. $d(S_1, S_2) = d(S_2, S_1)$, i.e., the edit transcript of transforming S_1 into S_2 has the same costs as the edit transcript of transforming S_2 into S_1 .
 - c. $d(S_1, S_3) \leq d(S_1, S_2) + d(S_2, S_3)$, i.e., the costs of transforming S_1 into S_2 into S_3 are at least so high as the costs of transforming S_1 into S_3 directly.
2. Transformation from distances to similarities:
 - a. Realized and nonrealized edges: consider **Fig. 1**. Alignment edges are constructed by means of edit operations. Nonrealized edges, i.e., dashed lines in **Fig. 1**, denote alignment edges that have exactly one gap at one of their ends. They are skipped for the multiple alignment step because they contain no information about aligning two nucleotides. Hence, we are left with realized edges. They are shown as thick or thin lines in **Fig. 1**. The thickness corresponds to the similarity weights.
 - b. Similarity weights: similarity weights are assigned to edit operations computed by the alignment algorithm. Here, we consider the number of nucleotides r involved in an edit operation. We call this number the order of the edit operation. In our case, we have edit operations with

- i. $r = 4$ for an arc match or an arc mismatch.
- ii. $r = 2$ for a base match or a base mismatch.
- iii. $r = 1$ for a base deletion.

Because we have split the arc deletion operation into two separate edit operations for the arc ends, we have an edit operation with $r = 2$ if the arc end is aligned with a nucleotide, and an edit operation with $r = 1$ if the arc end is aligned with $-$. The similarity weights can be achieved by choosing a maximal similarity value M , such that every value can be subtracted from the value $r \cdot M$. The value M is multiplied by r because we, therefore, ensure that all similarity values are positive. All edit operations on arcs with their associated distances and similarity weights are listed in [Fig. 2](#)

2.3. Multiple Alignment

Now, we are ready for the multiple alignment step. Suppose we have set of n RNAs together with their secondary structures. The main idea is to use the same strategy as proposed by the multiple alignment tool T-Coffee ([3](#)):

1. Recall that a single alignment between two RNAs provides a set of weighted, realized alignment edges.
2. The pairwise comparison strategy in a set of n RNAs yields $n(n-1)/2$ alignments. All these alignments produce an amount of weighted alignment edges each reflecting the sequence structure similarity between two bases. These edges are collected in a so-called library.
3. Now, the T-Coffee strategy is performed on this data set:
 - a. Library extension: the library containing all pairwise alignments with their weighted alignment edges is turned into an extended library to improve all pairwise alignments by taking into considerations how all other sequences align with the current two. For instance, if we consider two RNAs specified by their alignment and their weighted alignment edges then a third sequence is considered how this sequence is aligned with the first and the second sequence. For any alignment of two RNAs R_1 and R_2 , any other RNA R_3 is considered for improving the initial alignment. For this purpose, T-Coffee considers the alignment of R_1 and R_2 via R_3 by considering alignment edges from the alignment of R_1 and R_3 with edges from the alignment of R_2 and R_3 . These additional weighted edges together with the edges of the direct comparison of the first two RNAs are considered to improve this alignment by a dynamic programming approach. This procedure is executed $n(n-1)/2$ times, i.e., for each pairwise set of RNAs. The result is the extended library containing all improved pairwise alignments.

Edit-Op	Name	Distance	Similarity
	arc match	0	$4 \cdot M$
	arc mismatch	$w_{am}(A, U, G, C)$	$4 \cdot M - w_{am}(A, U, G, C)$
	arc breaking arc altering (realized edge)	$w_{base}(A, G) + \frac{1}{2} w_{ad}^{const}$	$2 \cdot M - w_{base}(A, G) - \frac{1}{2} w_{ad}^{const}$
	arc breaking arc altering (realized edge, two arcs)	$w_{base}(A, G) + w_{ad}^{const}$	$2 \cdot M - w_{base}(A, G) - w_{ad}^{const}$
	arc breaking arc removing (non-realized edge)	$w_{base}(A, -) + \frac{1}{2} w_{ad}^{const}$	no realized edge

Fig. 2. Edit operations on arcs together with the associated distances and their similarity values given to the T-Coffee system. Note that for arc-match and arc-mismatch, we assign half of the total similarity value to each alignment edge when building the library. Here, $W_{base}(A, C)$ are the costs for aligning A with C independent of whether the bases are free or not. W_{ad}^{const} are the costs for deleting an arc.

- b. Progressive alignment: pairwise distances of the sequence set were computed owing to the alignment algorithm. They form the distance matrix which is used to produce a neighbor-joining tree (6) that guides the alignment process. Residue weights that are stored in the extended library are now used for this task. The two closest sequences are aligned first. This alignment is fixed and the next closest sequence is aligned to this existing alignment or two new sequences are aligned or two existing alignments are aligned. In the case of aligning an already existing alignment, the average score in each column is taken. We do not need gap penalties because they are already included in the alignment as sequence identities and residue weights, i.e., residues which are aligned with gaps get a weight of zero.

2.4. Combining Several Structures

The multiple alignment of these RNAs assumes the existence of a known structure for each RNA like, e.g., an experimentally confirmed structure.

1. Whenever the structures are not known in advance, secondary structure prediction programs like Mfold (7) and RNAfold (8) may help to assign the minimum free structure to an RNA. The drawback here is that these structures are not necessarily the real existent structures which might be responsible for their functions.
2. To overcome this difficulty, we assign multiple structures to each sequence covering different folds. We call this set the ensemble of structures. We mainly use two different programs for generating these structure:
 - a. RNAsubopt (8): this program generates suboptimal structures by stochastic backtracking. The number of desired structures can be set individually.
 - b. RNASHAPES (5): this program avoids the large output of similar suboptimal structures; instead, it outputs structures of more fundamental differences.
3. The generated structures for each sequence S_i form an ensemble, denoted $E_S^i = \{E_1^i, \dots, E_n^i\}$. Because each structure E_i^l has its own energy, it occurs with probability, say $\Pr(E_i^l)$. Here, we consider rather a small set of important structures, in contrast to the explosive number of all suboptimal structures.
4. Probability: because of their different energies of these structures assigned to sequence S_i , the probability of seeing a certain structure E_k^l in a set of structures E_{S_i} with restricted size n is:

$$\Pr(E_k^l | E_{S_i}) = \frac{\Pr(E_k^l)}{\sum_{1 \leq i \leq n} \Pr(E_i^l)} \quad (3)$$

where $\Pr(E_k^l)$ is the probability of forming structure E_k^l in sequence S_i . In MARNA, the simplification of the uniform distribution is made, i.e., each structure has the same probability.

5. Alignment weights: consider two sequences S_1 and S_2 with n_1 structures for the first sequence and n_2 structures for the second sequence. If both $n_1 = 1$ and $n_2 = 1$, then the alignment algorithm outputs weighted alignment edges as we have seen before. These alignment edges are all multiplied by one because the number of structures in the ensemble equals one. Suppose we consider an ensemble of structure greater than one, then we have to make $n_1 \times n_2$, comparisons, i.e., each combination of (S_1, E_k^1) and (S_2, E_l^2) , $1 \leq k \leq n_1$, $1 \leq l \leq n_2$, has to be considered. The number of realized alignment edges is quadratic, i.e., proportional to $n_1 \times n_2$. The alignment weights are now influenced by the structural diversity. Each alignment edge is reweighted by the factor $\Pr(E_k^1 | E_{S_1}) \Pr(E_l^2 | E_{S_2})$. If both $|E_{S_1}| = 1$ and $|E_{S_2}| = 1$, then the alignment edges are weighted by the factor 1.

2.5. Consensus Structure

Once we have computed the final alignment, we are ready to calculate a consensus structure from this alignment. Here, we explicitly use structure information for the calculation of the alignment. Hence, the calculation of the consensus structure should be based on these ensemble structures.

1. To exemplify the basic idea, suppose that exactly one structure per sequence is given. Each structure must then be interpreted as the “real” known structure. A conserved basepair between two columns in the alignment is found if the majority of sequences have a basepair at the corresponding sequence positions. The remaining problem is that the resulting set of conserved basepairs alone does not form a secondary structure and is thus not a valid consensus structure. This is a problem common to all approaches for calculating a consensus structure.
2. We find a remedy by calculating a consensus secondary structure that maximizes basepair conservation. So let c, c' be two columns with $1 \leq c < c' \leq m$, where m is the number of columns of the multiple alignment. Furthermore, let $bp\_cons(c, c')$ be the number of sequences that have a basepair between the corresponding sequence positions. The consensus structure is then defined to be a secondary structure $P \subseteq [1..m] \times [1..m]$ such that

$$\sum_{(c,c') \in P} bp\_cons(c, c')$$

is maximized.

3. This can be calculated using dynamic programming. Let $N_{i,j}$ with $1 \leq i, j \leq m$ be the maximal basepair conservation for all columns between i and j :

$$N_{i,j} = \max_P \sum_{\substack{(c,c') \in P \\ i \leq c < c' \leq j}} bp\_cons(c, c')$$

The corresponding recursion equation for $N_{i,j}$ is

$$N_{i,j} = \max \begin{cases} N_{i+1,j} \\ N_{i,j-1} \\ N_{i+1,j-1} + bp\_cons(i, j) \\ \max_{i < k < j} \{N_{i,j+k} + N_{i+k+1,j}\} \end{cases}$$

It is a dynamic programming approach, where the traceback reports the consensus structure of the alignment.

4. Finally, we have to consider again the case where we are given structure ensembles for some (or all) sequences. Consider a multiple alignment of K sequences. For each sequence S_k , let E_{S_k} be the ensemble of structures calculated for S_k . For each

column c , let i_c^k be either the position that corresponds to column c in sequence S_k (if aligned), or - otherwise. Furthermore, let $\delta_P(c, c')$ be the index function of P , i.e., $\delta_P(c, c')$ is one if $(c, c') \in P$, and 0 otherwise. Then

$$bp\_cons(c, c') = \sum_{k=1}^K \sum_{E_i^k \in E_k} \delta_P(i_c^k, i_{c'}^k) \Pr(P|E_{S_k}),$$

where $\Pr(P|E_{S_k})$ is defined as given in **Eq. 3**.

3. Notes

1. MARNA can be tested online via the webpage <http://www.bioinf.uni-freiburg.de/Software/MARNA/index.html>. MARNA is also available as a downloadable file.
2. MARNA offers mainly two choices to adjust the alignments:
 - a. Parameter settings: MARNA relies on the comparison of pairwise RNAs. These comparisons are accomplished by alignments with costs assigned to edit operations on bases and arcs. These costs can be set individually.
 - b. Structure computation: the alignment of RNAs take into account both the primary sequences and the secondary structures. The easiest case is when the secondary structures are known in advance, and the computation is reduced to find common sequential and structural properties. Otherwise, the structures have to be found. MARNA provides in addition to user-defined structures the assignment of different sets of structures. These include the assignment of minimum free energy structures, shaped structures or an ensemble of low energy structures.
3. Parameter settings: parameters can be set individually depending on weighting some edit operations more or less. A series of tests has brought three data sets to obtain alignments based on sequential or structural properties or on a mixture on both. These data sets are shown in **Fig. 3**.

edit operations	default	sequential	structural
base deletion	2.0	2.0	0.1
base mismatch	1.0	1.0	0.1
arc breaking	1.5	0.1	1.5
arc mismatch	1.8	0.1	1.8

Fig. 3. Data sets found out for weighting sequential or structural properties or on a mixture of both (default values). The values correspond to costs that can be set in the MARNA system.

4. Parameters settings influence the resulting alignments. Choose the default parameter settings first. It has been confirmed that this data set recognizes conserved sequential and structural properties very well.
5. Beyond the parameter settings, the assignment of different structures to the sequences are quite important as well. The easiest case is when user-defined structures are given as input.
6. Structure choice: here are some hints to choose the right structure assignments if no structures are given to the sequences.
 - a. If the RNAs are sequentially related and have nearly the same length then choose the minimum free energy structures.
 - b. The shaped structures are suited to cover a lot of diverse structural conformations for each single sequence. Choose shape structures, if no clear consensus structure is observable at first glance.
 - c. The ensemble set of low energy conformations is the best choice if it is assumed that these RNA sequences resemble structurally in some way. An ensemble consists of multiple structures. This ensemble contains similar structures if almost all suboptimal structures are similar.
7. The running time of MARNA crucially depends on the structure choices. Suppose n RNAs of nearly the same length without structure specifications are given. If the mfe structures are chosen that are assigned to the sequences then the multiple alignment and the consensus structure computation can be done in reasonable time. Suppose the user chooses an ensemble of three suboptimal structures to each RNA, then the computation time is ninefold because for each pair of RNAs nine pairwise sequence structure comparisons have to be made.

References

- 1 Doudna, J. A. and Cech, T. R. (2002) The chemical repertoire of natural ribozymes. *Nature* **418**, 222–228.
- 2 Jiang, T., Lin, G., Ma, B., and Zhang, K. (2002) A general edit distance between RNA structures. *J. Comput. Biol.* **9**, 371–388.
- 3 Notredame, C., Higgins, D. G., and Heringa, J. (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* **302**, 205–217.
- 4 Siebert, S. and Backofen, R. (2005) MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics* **21**, 3352–3359.
- 5 Giegerich, R., Voss, B., and Rehmsmeier, M. (2004) Abstract shapes of RNA. *Nucleic Acids Res.* **32**, 4843–4851.
- 6 Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**, 406–425.

- 7 Zuker, M. (1994) Prediction of RNA secondary structure by energy minimization. *Methods Mol. Biol.* **25**, 267–294.
- 8 Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, S., Tacker, M., and Schuster, P. (1994) Fast folding and comparison of RNA secondary structures. *Monatshefte f. Chemie.* **125**, 167–188.

Prediction of Structural Noncoding RNAs With RNAz

Stefan Washietl

Summary

The function of many noncoding RNAs (ncRNAs) depend on a defined secondary structure. RNAz detects evolutionarily conserved and thermodynamically stable RNA secondary structures in multiple sequence alignments and, thus, efficiently filters for candidate ncRNAs. In this chapter, we provide a step-by-step guide on how to use RNAz. Starting with basic concepts, we also cover advanced analysis techniques and, as an example for a large scale application, demonstrate a complete screen of the *Saccharomyces cerevisiae* genome.

Key Words: Noncoding RNA; gene finding; conserved RNA secondary structure; RNA structure prediction.

1. Introduction

1.1. Prediction of Noncoding RNAs

In contrast to protein-gene finders that are routinely used for genome annotation, noncoding RNA (ncRNA) gene finders are still in their infancy. Systematic *de novo* prediction of ncRNAs is hindered by the fact that there are no common statistically significant features in primary sequence (e.g., open reading frames or codon bias), which could be exploited for efficient algorithms.

It is not clear what is defined as “ncRNA.” There is no doubt that independent “RNA genes” with a defined molecular function, such as tRNAs, microRNAs, or snoRNAs, should be called ncRNAs. But the situation is not always clear. The transcriptional activity of mammalian genomes is much more complex than anticipated ([1](#)). We see mRNA-like ncRNAs, non-polyadenylated RNAs

from both intronic and intergenic regions, overlapping transcripts, extensive antisense transcription, and transcribed protein-pseudogenes. In addition, there is a recent example of a noncoding transcript that only is expressed to interfere with and downregulate the transcription of a neighboring gene, but the produced RNA molecule itself does not have any obvious function (2). There is even an example of a functional RNA encoding a protein (3). The spectrum of ncRNAs and their mode of action is very heterogeneous. It can be safely assumed that the full spectrum of functions is not yet discovered and that a general ncRNA gene finder is an unrealistic goal, even in the long term.

However, there is a subclass of ncRNAs that—with the help of comparative genomics—can be predicted with fair accuracy. Structural ncRNAs have a defined and evolutionarily conserved secondary structure that is of functional importance. Most of the well-known “classical” ncRNAs, as for example tRNA, rRNA, RNase P, or SRP RNA, are of this class. Pioneering work in the prediction of structural ncRNAs by comparative genomics was preformed by Rivas and Eddy. QRNA predicts conserved RNA secondary structures on pairwise alignments using a probabilistic approach based on a stochastic context free grammar to model RNA structure (4–6). RNAz (7) takes a different approach. It is based on minimum free energy (MFE) structure prediction algorithms (8,9). It relies on the fact that structural RNAs have two characteristic features: (1) unusual thermodynamic stability and (2) conservation of secondary structure. The following section outlines the basic principles of RNAz.

1.2. The RNAz Approach

1.2.1. Thermodynamic Stability

It is easy to calculate the MFE as a measure of thermodynamic stability for a sequence using, e.g., RNAfold (9). However, the MFE depends on the length and the base composition of the sequence and is, therefore, difficult to interpret in absolute terms. RNAz calculates a normalized measure of thermodynamic stability by comparing the MFE m of a given (native) sequence to the MFEs of a large number of random sequences of the same length and base composition. A z -score is calculated as $z = (m - \mu) / \sigma$, where μ and σ are the mean and standard deviations, respectively, of the MFEs of the random samples. Negative z -scores indicate that a sequence is more stable than expected by chance. RNAz does not actually sample random sequences but approximates z -scores, which is faster but of the same accuracy.

1.2.2. Structural Conservation

RNAz predicts a consensus secondary structure for an alignment by using the RNAalifold approach ([10](#), see also **Chapter 33 of this book**.) RNAalifold works almost exactly like single sequence folding algorithms (e.g., RNAfold), with the main difference being the energy model is augmented by covariance information. Compensatory mutations (e.g., a CG pair mutates to a UA pair) and consistent mutations (e.g., AU mutates to GU) give a “bonus” energy, whereas inconsistent mutations (e.g., CG mutates to CA) yield a penalty. This results in a consensus MFE E_A . RNAz compares this consensus MFE to the average MFE of the individual sequences $\bar{E}$ and calculates a structure conservation index: $SCI = E_A/\bar{E}$. The SCI will be high if the sequences fold together equally well as if folded individually. On the other hand, SCI will be low if no consensus fold can be found.

1.2.3. Putting it Together

The two independent diagnostic features of structural ncRNAs, z-score and SCI, are finally used to classify an alignment as “structural RNA” or “other.” For this purpose, RNAz uses a support vector machine (SVM) learning algorithm, which is trained on a large test set of well known ncRNAs.

Using RNAz, it is possible to efficiently screen alignments for functional RNA secondary structures. It is important to note that RNAz cannot distinguish functional RNA elements that are part of ncRNAs from elements that are *cis*-regulatory elements of mRNAs.

1.3. The Focus of This Chapter

There are two main goals of this chapter. First, we want to give detailed technical advice on how to use RNAz. Second, we want to provide the user with a well-founded understanding of the results from RNAz. We want to assist in a sensible interpretation of RNAz predictions—leading, as we hope, to reasonable conclusions for the application.

In the first part, we explain how to install RNAz and all necessary helper programs on the system. Next, we demonstrate the basic usage of RNAz including the correct formatting of the input alignments. More advanced techniques, which require preprocessing steps of the input alignments, are discussed afterward. In the last section, we demonstrate how to conduct a RNAz screen of a large number of automatically generated alignments on the example of genome-wide screen of *Saccharomyces cerevisiae*.

1.4. General Remarks and Typographical Conventions

There is no graphical user interface for RNAz. All steps are carried out in a command-line (terminal). Lines starting with a “#” are commands and they should be typed into the terminal window, followed by pressing return. The “#” sign stands for the command-line prompt and may not look of same on different systems. If a command is too long for one line in this chapter, it is separated by a backslash “\” and continues on the next line. Do not input the backslash, simply type in the command on one line.

All programs are implemented as filters, i.e., they read from the standard input and write to the standard output. Therefore, we make use of the pipe (“|”) and redirection operators (“<”, “>”).

Online documentation on the usage of each program can be obtained by using the `--help` option, e.g.:

```
# RNAz --help
```

Most command-line options have a long (e.g., `--help`) and a short (e.g., `-h`) form. For didactic reasons, we use long option names throughout this chapter.

2. Materials

1. Hardware. RNAz is generally fast. Small- to medium-sized data sets, for example the yeast screen in **Subheading 3.6.**, can be analyzed within reasonable time on a single modern desktop or laptop computer.
2. Operating system. If available, we recommend to use a Linux/UNIX system for the analysis. Also Mac OS X, in principle a full-featured UNIX system, is an adequate platform. Alternatively, RNAz can also be run on Microsoft Windows. Most of the methods described in this chapter can be carried out on Windows without any modification.
3. Perl. The RNAz program is bundled with a variety of helper programs, which are written in the Perl programming language. To run these programs, Perl needs to be installed on the system, which is most likely the case in all Linux/UNIX systems and Mac OS X. Perl is not part of a standard Windows system. Windows users can download it from www.activestate.com. Choose the latest ActivePerl MSI installer package for Windows and simply follow the installation instructions. Make sure that selected the “Add Perl to the PATH environment variable” and “Create Perl file extension association” options have been selected during installation.
4. RNAz. The RNAz program can be downloaded from www.tbi.univie.ac.at/~wash/RNAz. For the examples in this chapter, RNAz v1.0 was used. For Linux/UNIX and OS X, download the file `RNAz-1.0.tar.gz`. Windows users can download the file `RNAz-1.0-win32.msi`.

5. Optional software. Some advanced analysis steps (**Subheadings 3.6.8.** and **3.6.10.**) require additional software to be installed on the system. To create HTML formatted output of the results as described in **Subheading 3.6.8.**, the Vienna RNA package (www.tbi.univie.ac.at/RNA) and the postscript interpreter Ghostscript (<http://www.cs.wisc.edu/~ghost/>) need to be installed. To perform automatic database searches of predicted ncRNA candidates the user will need NCBI Blast (<ftp://ftp.ncbi.nih.gov/blast>).
6. Example files. Most of the example files used in this chapter are part of the RNAz package. To reproduce the *S. cerevisiae* screen described in **Subheading 3.6.**, the data file can be downloaded file from www.tbi.univie.ac.at/papers/SUPPLEMENTS/MiMB/.

3. Methods

3.1. Installation of RNAz

3.1.1. Linux/UNIX and OS X

In the simplest case, run the following series of commands to build and install RNAz:

```
# tar -xzf RNAz-1.0.tar.gz
# cd RNAz-1.0
# ./configure
# make
# su
# make install
```

This requires root privileges and installs all files under the `/usr/local` tree. The RNAz executable is installed in `/usr/local/bin` and the user should now be able to run the program (try `RNAz --version` on a terminal window). If the user does not have root privileges or experiences other problems (e.g., gcc compiler not found) see **Note 1**.

The Perl programs are installed to `/usr/local/share/RNAz/perl`. To make these programs available from other locations, the user can either add this directory to the `PATH` of executables environment variable or copy the Perl programs to an existing directory already in the `PATH`. **Note 2** describes how to run Perl programs.

3.1.2. Microsoft Windows

To install RNAz on Windows simply double click on the `RNAz-1.0-win32.msi` and follow the instructions. Open a command prompt and type `RNAz --version` to test the installation.

3.2. Installation of Optional Software

We cannot cover in detail the installation procedure of the optional software. We will just give an outline of how to install the Vienna RNA package and NCBI blast on a standard Linux system. Together with an existing Ghostscript installation, this will allow the examples to be run in **Subheadings 3.6.8.** and **3.6.10.** for Windows and OS X (*see Note 3*).

To install the Vienna RNA package, get the latest `ViennaRNA-X.X.tar.gz` file from `www.tbi.univie.ac.at/RNA`. The package can be installed in exactly the same way as RNAz, using `./configure` and `make`. Please refer to the `INSTALL` document for detailed installation options. Make sure that the Perl programs in the `Utils` directory are in the `PATH` of executables.

To install NCBI Blast, download the `blast-2.*.tar.gz`-package matching the platform from `ftp://ftp.ncbi.nih.gov/blast/executables/LATEST/`. Copy it to the installation directory of choice and “untar” it. The executables are located in the `bin` subdirectory, which should be added to the `PATH` variable.

3.3. Installation of Example Files

Move the example file `yeast-examples.tar.gz` to the directory of choice and “untar” the file:

```
# tar -xzf yeast-examples.tar.gz
```

If Windows is being used, download the file `yeast-examples.zip` and unzip it in the directory of choice.

3.4. Basic Usage of RNAz

3.4.1. Input Alignment

RNAz takes a multiple sequence alignment as input. RNAz does not align sequences, so other programs need to be used to create the alignments. If the alignments are prepared manually (in contrast to automatic genome-wide alignments as in **Subheading 3.6.**), we recommend using Clustal W (**1**). It is an easy-to-use and widely available tool, which performs well on structural RNAs (**12**). For hints on preparing the alignments *see Note 4*.

RNAz can read two different alignment formats: Clustal W (**Fig. 1A**) and MAF (**Fig. 1B**). The Clustal W format is a concise format, which is supported by many programs and thus suitable for everyday use.

For genomic screens, however, it is necessary to exactly store the genomic locations of aligned sequences. For this purpose, the MAF format was developed, which requires six fields for each sequence entry:

A

```
CLUSTAL W (1.83) multiple sequence alignment

sacCer1      GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGTTAGGGGTTTCGAG
sacBay       GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGTTAGGGGTTTCGAG
sacKlu       GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGTTAGGGGTTTCGAG
sacCas       GCTTCAGTAGCTCAGTCGGAAGAGCGTCAGTCTCATAATCTGAAGGTCGAGAGTTCGAA
              ** *   * ** ** ***** ** ***** * *** ***** ***** * ***** *

sacCer1      CCCCTACAGGGCT
sacBay       CCCCTACAGGGCT
sacKlu       CCCCTACAGGGCT
sacCas       CTCCCCTGGAGCA
              * **   * **
```

B

```
##maf version=1
a score=119673.000000
s sacCer1.chr4      1352453 73 - 1531914 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT..
s sacBay.contig_465 14962 73 - 57401 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT..
s sacKlu.Contig1694 137 73 + 4878 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT..
s sacCas.Conti 128 258 73 + 663 GCTTCAGTAGCTCAGTCGGAAGAGCGTCAGTCTCAT..
```

C

```
##### RNAz 0.1.1 #####

Sequences: 4
Columns: 73
Reading direction: forward
Mean pairwise identity: 80.82
Mean single sequence MFE: -27.20
Consensus MFE: -26.50
Energy contribution: -23.62
Covariance contribution: -2.88
Combinations/Pair: 1.43
Mean z-score: -2.18
Structure conservation index: 0.97
SVM decision value: 2.39
SVM RNA-class probability: 0.993311
Prediction: RNA

#####

>sacCer1.chr4 1352453 73 - 1531914
GCCUUGUUGGCGCAAUCGGUAGCGGUAUGACUCUUAUCAUAAGGUUAGGGGUUCGAGCCCCUACAGGGCU
(((((((.(((.....))))).(((.(.(((.....)))))))))((((.....))))).))))). (-29.20)
>sacBay.contig_4651496273 - 57401
GCCUUGUUGGCGCAAUCGGUAGCGGUAUGACUCUUAUCAUAAGGUUAGGGGUUCGAGCCCCUACAGGGCU
(((((((.(((.....))))).(((.(.(((.....)))))))))((((.....))))).))))). (-29.20)
>sacKlu.Contig169413773 + 4878
GCCUUGUUGGCGCAAUCGGUAGCGGUAUGACUCUUAUCAUAAGGUUAGGGGUUCGAGCCCCUACAGGGCU
(((((((.(((.....))))).(((.(.(((.....)))))))))((((.....))))).))))). (-27.20)
>sacCas.Contig128 258 73 + 663
GCUUCAGUAGCUCAGUC
(((((((.(((.....))))).(((.....))))((((.....)).))))).))))). (-23.20)
>consensus
GCCUUGUUGGCGCAAUCGGUAGCGGUAUGACUCUUAUCAUAAGGUUAGGGGUUCGAGCCCCUACAGGGCU
(((((((.(((.....))))).(((.....))))).((((.....)))))))).
(-26.50 = -23.62 + -2.88)
```

Fig. 1. Supported alignment formats and RNAz output. (A) Clustal W format, (B) MAF format (sequences have been shortened because of space restrictions), (C) output of RNAz on the MAF file shown in B.

1. A unique identifier of the source sequence.
2. The start position of the aligned subsequence with respect to this source sequence.
3. The length of the aligned subsequence without gaps.
4. A “+” or “-” indicating if the sequence is in the same reading direction of the source sequence or the reverse complement.
5. The sequence length of the complete source sequence.
6. The aligned subsequence with gaps.

The full specification of the format can be found at <http://genome.ucsc.edu/goldenPath/help/maf.html>. Note that RNAz and all other helper programs do not make use of field 5 and also ignore the value of the “score=” field in the header line. So it is possible to simply fill these fields with 0 or any other arbitrary values, if the real values are not easily available.

The RNAz package contains several example files, which are by default installed to `/usr/local/share/RNAz/examples`. To run the following examples change into this directory.

3.4.2. Running RNAz

As soon the alignments have been prepared, they can be immediately scored with RNAz. In the simplest case, type:

```
# RNAz tRNA.maf
```

The file `tRNA.maf` is that one shown in **Fig. 1B** and the command gives the output shown in **Fig. 1C**.

3.4.3. Understanding the Output

As described in the introduction, RNAz calculates various folding characteristics to classify the alignment. These are displayed in the header section of the RNAz output.

The mean single MFE is compared to the consensus MFE, which results in the SCI, a measure for structural conservation (**Subheading 1.2.2.**). In this ideal example of a tRNA, we observe a very high SCI of 0.97. The SCI depends on the mean pairwise identity and the number of sequences in the alignment. So, it is not possible to interpret the significance of a SCI-value in absolute terms. As a rule of thumb, a SCI near or even more than the mean pairwise identity is “good” and might indicate structural conservation. For example, given an alignment with five sequences and a mean pairwise identity of 60%, a SCI of 0.75 can be regarded as strong hint for a conserved fold. On the other hand, on a pairwise alignment with 90% identity, SCI=0.75 does not indicate a conserved fold at all.

The second characteristic is thermodynamic stability, which is expressed as the mean z -score of the sequences in the alignment (*see Subheading 1.2.1.*). z -scores of MFEs are not normally distributed, so the user cannot directly give a statistical significance for the z -score. However, mean z -scores less than -3 or -4 generally indicate very stable structures that should arise only in rare cases by chance. Also here, one has to consider the overall sequence divergence in the alignment. On a pairwise alignment with 90% identity a z -score of -4 is much more likely to occur by chance than on an alignment of six sequences with only 60% identity.

Apart from SCI and z -score, there are a few other values displayed in the RNAz output (for their meaning *see Note 5*).

RNAz assists in the final classification by providing an overall “RNA-class probability,” or “ p -value.” It is important to know that this is not a p -value in a strict statistical sense, simply because there is no underlying statistical model. Instead, RNAz uses a rather *ad hoc* machine learning technique to calculate this value. If $p > 0.5$, the alignment is classified as “RNA.” The false-positive rate at this cutoff was found to be $\approx 4\%$, i.e., we expect four positive hits in 100 random alignments. For many applications it is useful to set a more stringent cutoff of $p = 0.9$ with an associated false-positive rate of $\approx 1\%$. Reasons why estimations of false-positives must always be taken with caution are given in **Note 6**.

It turned out to be a useful practice to use $p = 0.5$ and $p = 0.9$ as two main levels of significance. A more sophisticated interpretation of the p -value without considering the other values is generally not useful. In most cases, the user cannot say that, for example, a hit with $p = 0.97$ is more reliable than a hit with $p = 0.95$. *See Note 7* to assess the reliability of a hit based on other criteria.

In the lower part of the RNAz output the predicted structures are explicitly seen for the sequences. There will be structure predictions for each single sequence and a consensus structure prediction for the whole alignment. The predicted structures are given below the sequences in a “dot-bracket” notation. Each basepair in the secondary structure is indicated by a pair of brackets: “(” and “)”. Unpaired bases are shown as dot: “.”. Next to the structure the MFE will be seen in kcal/M. A graphical output can be obtained by using RNAalifold, which is described in detail in Chapter 33 of this book.

3.5. Advanced Usage of RNAz

3.5.1. Analyzing Forward and Reverse Strand

For a given alignment, a putative RNA can either be read in the forward direction or in the reverse complementary direction. Therefore, both reading directions should be scanned. By default, only the forward direction is scored,

but the `--forward`, `--reverse`, and `--both-strands` flags can be used to explicitly specify the reading direction.

If there is a strong RNA signal in one strand, in many cases, a signal can also be observed in the reverse complement. Usually the signals (SCI, z -score, consensus MFE) are stronger in the “correct” direction. In most cases, this also goes along with a better p -value. That is not always the case and, therefore, RNAz uses a separate SVM decision model to predict the correct strand. Please note, that in v1.0 this is still an experimental feature. The following command will analyze both strands of the tRNA and, in addition, activate the strand prediction:

```
# RNAz --both-strands --predict-strand tRNA.maf
```

In this example, the signal from both strands are almost indistinguishable and also the p -values are almost the same (0.993 and 0.999). RNAz still suggests the correct (forward) strand and displays a “strand class probability:”

```
# Strand winner: forward (0.88)
```

3.5.2. Scoring Alignments With More Than Six Sequences

RNAz is currently limited to alignments with not more than six sequences. If there are more than six sequences in the alignment, the number must be reduced either manually or by using the `rnazSelectSeqs.pl` program to filter the alignment before it is put into RNAz:

```
# rnazSelectSeqs.pl miRNA.maf | RNAz
```

The file `miRNA.maf` contains 12 aligned microRNAs. With default parameters, `rnazSelectSeqs.pl` selects a subset of six sequences trying to reach an optimal mean pairwise identity around 80%.

The default behavior can be customized in various ways (use `--help` for details). The following command, for example, samples three different alignments with four sequences each.

```
# rnazSelectSeqs.pl --num-seqs=4 \
--num-samples=3 miRNA.maf | RNAz
```

By default, the first sequence in the alignment is always in the set of selected sequences. This is the desired behavior for genomic screens, where one usually likes to retain a reference sequence.

3.5.3. Scoring Long Alignments

RNAz cannot score alignments longer than 400 columns. In practice, it is generally advisable for the user to score long alignments, say >200 columns, in shorter, overlapping windows. For general purpose screens, we recommend a window size of 120. This window size appears large enough to detect local secondary structures within long ncRNAs and, on the other hand, small enough to find short secondary structures without losing the signal in a much too long window.

The file `unknown.aln` contains a noncoding region conserved in vertebrates. The file can be scanned for RNA secondary structures by typing:

```
# rnazWindow.pl --window=120 --slide=40 unknown.aln\  
| RNAz --both
```

The results show that RNAz does not predict an RNA in this region. On UNIX like system, “`| grep Prediction`” can be added to get a quick overview on the results. The `rnazWindow.pl` program has numerous additional functions and will be used again in **Subheading 3.6**.

3.6. Large-Scale Genomic Screens

3.6.1. Overview

An analysis pipeline suitable for scanning a large number of genomic alignments is outlined in [Fig. 2](#). In the following, we demonstrate the usage of this pipeline on the example of a genomic screen of *S. cerevisiae*. We want to describe the method as general as possible and we will focus here mainly on technical details. A paper describing the results of a comprehensive RNAz screen in yeast is in preparation ([13](#)).

3.6.2. Choosing Raw Input Alignments

Choosing a reasonable set of input alignments is one of the most important steps during the analysis. There are a variety of different programs available to generate genome-wide alignments. Here, we use Multiz alignments of up to seven *Saccharomyces* species, which can be downloaded from the UCSC genome browser (genome.ucsc.edu). In principle, we could use all alignments covering the complete genome. The biggest problem in large genomic screens is probably specificity. We have a relatively constant background signal of false-positives. The more sequences we put into the screen, the more false-positives we get out. It is, therefore, a good idea to choose the input set as small

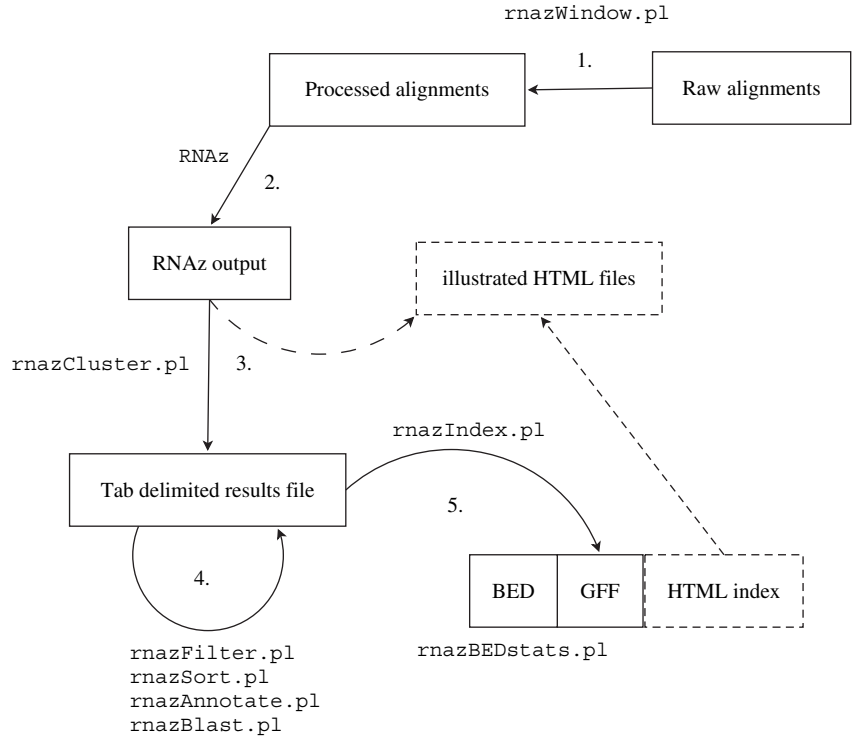


Fig. 2. Analyzing pipeline illustrating the use of RNAz and the helper programs. (1) `rnazWindow.pl` slices the input alignments in overlapping windows and performs a variety of filtering and preprocessing steps. (2) The processed alignments can be scored with the RNAz program. (3) Overlapping hits are merged with `rnazCluster.pl`. In addition, all relevant data is extracted from the raw output and stored in a tabulator delimited data file. Using the `html` option, `rnazCluster.pl` generates a tree of HTML pages with illustrations of the predicted structures. Additional software is needed for this step to work. (4) The results can be filtered, sorted and annotated in various ways. All programs read a tab-delimited file and write a tab-delimited file. (5) Using `rnazIndex.pl`, the tab-delimited data files can be exported to standard formats as GFF and BED. It is also possible to create a HTML formatted index file for the optional HTML output created in step 3.

as possible (trying not to discard any interesting regions of course). In our case, we only analyze the intergenic regions, i.e., we discard any coding regions and all other annotated features (pseudogenes, repeats, ARS elements,...). We retain known ncRNAs as positive control in the set. The selection was

easily accomplished using the “Table browser” feature of the genome browser. We finally obtained a MAF alignment (`input.maf`) with 10,822 alignment blocks, covering 983,947 bases of the genome (see **Subheading 3.6.11.** for how to get these numbers out of a MAF file.).

3.6.3. Preprocessing Raw Alignments

As described in **Subheading 3.5.3.**, it is necessary to score long alignments in overlapping windows. Given the partly poor quality of automatically generated genome-wide alignments additional preprocessing steps are required to filter out gap-rich regions, dubious aligned fragments or low complexity regions. All preprocessing is done by the `rnazWindow.pl` program, which, per default, performs the following steps:

1. Slice alignments in overlapping windows of size 120 and slide 40.
2. Check each pairwise alignment of the reference sequence (= first sequence) to all other sequences and, after removing common gaps, discard sequences with more than 25% gaps in this pairwise alignment.
3. Discard any sequences which are outside the definition range of RNAz (e.g., <50 nt, GC content >0.75).
4. Discard the complete alignment if either the reference sequence was discarded in a previous step or only the reference sequence is left (i.e., number of sequences <2).
5. If the number of sequences is >6, choose a subset of 6 sequences with mean pairwise identity optimized to a target value of 80%.
6. Remove all sequences which are 100% identical. Never remove the reference sequence and if all sequences are identical retain only a pairwise alignment.

All these steps can be customized with the appropriate command-line parameters. Here, we use the default settings. We define, however, a minimum number of four sequences in the alignment retaining only regions that are well conserved across several species:

```
# rnazWindow.pl --min-seqs=4 input.maf > windows.maf
```

This command will take a few minutes.

3.6.4. Running RNAz

The file `windows.maf` is now ready for being scored with RNAz. We use the `--both-strands` parameter to score both the forward and the reverse complement strand. We also set `--show-gaps`, which means that the output is shown including the gaps. With this option it is possible to recover the complete alignment from the RNAz output file, which is useful in later steps of

the pipeline. Finally, we set a p -value cutoff of 0.5, meaning that only positive predictions are stored resulting in a much smaller output file.

```
# RNAz --both-strands --show-gaps --cutoff=0.5\
windows.maf > rnaz.out
```

This will take approx 1h on a modern desktop computer but may vary depending on the system.

3.6.5. Clustering the Results

The file `rnaz.out` now holds all windows that have a positive RNAz signal with $p > 0.5$. It is possible that several windows cover the same genomic region. Overlapping windows are therefore clustered in *loci*:

```
# rnazCluster.pl rnaz.out > results.dat
```

This command assigns each window a consecutively numbered “window ID” and each group of overlapping windows a “locus ID.” For each window and each locus all relevant data (use `--help` for details) is stored in a tabulator separated text file.

Inspecting the file `results.dat`, we see that we have 1104 windows which can be grouped in 454 loci.

It is important to note that the term “locus” must not be understood in the sense of a genetic unit. It is, of course, possible that several loci of our procedure cover one long ncRNA gene.

At this point we also want to add that we are painfully aware of the fact that the process of first slicing the alignments and the reclustering of them is not optimal. Ideally, one would like to predict conserved RNA structures locally without sliding windows. Although this should be possible ([14](#)) and we are working on a local version of RNAz, the sliding window approach is currently the only reasonable protocol.

3.6.6. Filtering and Sorting the Results

The data file now contains the raw data of all hits. In the following analysis steps, one usually wants to filter and sort candidates by various criteria. For this purpose, the programs `rnazFilter.pl` and `rnazSort.pl` can be used. For example,

```
# rnazFilter.pl "P>0.9" results.dat
```


lists all windows that have a p -value more than 0.9. For hints on how to formulate more complex filtering expressions *see* **Note 8**. The `--count` option will count the hits. We have 670 Windows in 303 loci on the $p > 0.9$ significance level. In addition, we can sort the hits:

```
# rnazFilter.pl "P>0.9" results.dat \  
| rnazSort.pl combPerPair
```

This sorts the output by the “Combinations/Pair” value, i.e., by compensatory mutations supporting the structure (explained in **Note 5**).

3.6.7. Exporting the Results to Standard Annotation Formats

Using different combinations of `rnazFilter.pl` and `rnazSort.pl`, various subselections of the complete data from `results.dat` can be created. There will always be a tabulator delimited data file. The program `rnazIndex.pl` helps to convert these kind of data files into the standard annotation formats GFF (`--gff`) or BED (`--bed`). GFF (<http://www.sanger.ac.uk/Software/formats/GFF/>) is a widely used format supported by many programs. BED (<http://genome.ucsc.edu/FAQ/FAQformat>) is the native annotation format for the UCSC genome browser but is generally useful because of its simplicity (in its simplest form it is a list of genomic locations: `sequenceID start stop`).

The following command creates a GFF file from all results:

```
# rnazIndex.pl --gff results.dat > results.gff
```

3.6.8. Visualizing the Results on a Website

It is often insightful to manually check individual predictions, for example by analyzing different illustrations of consensus structures (*see* **Note 7**). The creation of the necessary files is a tedious task, which, however, can easily be automatized. If the cluster command from **Subheading 3.6.5** is run with the option `--html`,

```
# rnazCluster.pl --html rnaz.out > results.dat
```

the program generates image files for all hits. For the `--html` option to work, the Vienna RNA package must be installed (including the Perl programs of the Utils directory) and the program Ghostscript, *see* **Subheading 2., point 5**, `rnazCluster.pl` creates a subdirectory called `results`, which, in turn, has a subdirectory `locusN` for each locus. In the `locusN` directories, the images

files can be found together with an `index.html`, which arranges the images for each locus on a webpage. The index files can be opened using almost any web browser.

To get an HTML-formatted table of all hits linking to the subpages for each locus, use `rnazIndex.pl` with the `--html` option:

```
# rnazIndex.pl --html results.dat > results/results.html
```

3.6.9. Comparing Hits to Known Annotation

Once there is a list of predicted RNAs, the user may want to add additional annotations to the predictions. Additional fields can be added to the tabulator separated data file. Here, we demonstrate this by comparing our prediction with the known ncRNA annotation from the *Saccharomyces* genome database. The program `rnazAnnotate.pl` checks each predicted locus for overlap with an annotation file in BED format:

```
# rnazAnnotate.pl --bed ../sgdRNA.bed results.dat \
> annotated.dat
```

We find that out of 454 predicted loci, 280 overlap with known ncRNAs (of the 303 loci with $p > 0.9$, 215 are known ncRNAs). We detect all sorts of different ncRNA classes (tRNAs, rRNAs, snRNAs, snoRNAs, RUFs [6], and other ncRNAs like telomerase RNA or RNaseP, ...). Most of the known 373 ncRNAs in yeast are tRNAs (275), which are partly difficult to detect in this screen because most of them are $\approx 100\%$ conserved (i.e., no covariance information).

Without providing a detailed sensitivity analysis for this specific yeast screen, we want to add that sensitivity highly depends on the ncRNA class. MicroRNAs, for example are easy to detect because of the high thermodynamic stability of the hairpin precursor. On the other hand, C/D type snoRNAs for example are generally difficult to detect because they lack a pronounced secondary structure. We completely miss ncRNAs, which do not depend on a secondary structure for their function, as for example the yeast *SER3* regulating RNA (2), which, as expected, does not show up in this screen.

3.6.10. Annotating Hits With Database Search

Another possibility to annotate predicted ncRNAs is to compare the sequences to databases of known ncRNAs. In the following, we match the predicted loci against the Rfam database (15) using a simple Blast sequence search. Alternatively, one could use more sensitive methods, which also

incorporate secondary structural information (e.g., Infernal [16]). To run this example, the *S. cerevisiae* sequence files will be needed, the Rfam database file and a working NCBI Blast installation.

First, change into the directory `rfam` and run:

```
# formatdb -t rfam -i rfam -p F
```

This command creates the index files for the file `rfam`, which is a Fasta formatted file with all entries of the database. Now run:

```
# rnazBlast.pl --database rfam --seq-dir=seq \
  --blast-dir=rfam results.dat >annotated.dat
```

This program takes the *S. cerevisiae* reference sequence for each locus and runs a Blast search against the Rfam database. If there is a hit with an expectation value below some cutoff (default: $E < 10^{-6}$), the name of the matching database query is added as a new field to the data file. Please note that the locations of the sequence data files and the blast index files on the command-line must be specified.

3.6.11. Estimating False-Positives and Gathering Statistics

To get an impression of the false-positive rate of a specific screen it is useful to do a control screen on randomized alignments. The command

```
# rnazRandomizeAln.pl input.maf > random-input.maf
```

will produce a randomized version of the input alignments by shuffling the positions in the alignments. The program aims to remove any correlations arising from a natural secondary structure while preserving important alignment and sequence characteristics as for example mean pairwise identity or base composition (17).

We repeated the complete analysis with the randomized alignments and we get 102 and 39 loci, on the $p > 0.5$ and $p > 0.9$ level, respectively.

Table 1 summarizes all results of this example screen. There are a few programs that will help with the gathering statistics on the data. For example,

```
# rnazIndex.pl --bed results.dat \
  | rnazBEDsort.pl | rnazBEDstats.pl
```

Table 1
Statistics of the Yeast Example Screen

	<i>p</i> > 0.5	<i>p</i> > 0.9
Predicted loci	454	303
Known ncRNAs	280	215
Loci without annotation	174	88
Predicted bases	60,834	44,082
Fraction of input alignments (%)	10.6	7.7
Predicted loci random	102	39
Predicted bases random	12,823	6017
Fraction of input alignments random (%)	2.2	1.0

gives detailed information on the predicted loci, including the covered genomic region in nucleotides. This command first exports the results as BED file, sorts the results by the genomic location and, finally, evaluates the coordinates in the BED file. To get statistics on the input alignments, use a command like this:

```
# rnazMAF2BED.pl --seq-id=sacCer windows.maf \  
| rnazBEDsort.pl | rnazBEDstats.pl
```

`rnazMAF2BED.pl` converts a MAF formatted alignment file to coordinates in BED format. With `--seq-id` the user specifies which sequence is used as reference.

Using these tools, the user will find, for example, that in the random control 1.0 % of the input sequences are predicted as RNA on the $p > 0.9$ level. This is exactly the false-positive rate as expected (**Subheading 3.4.3.**). The absolute number of false-positives, however, strongly depends on the specific screen. In this example we have 88 hits $p > 0.9$ without RNA annotation and find that 39 hits should be expected by chance. So we must expect that roughly half of our predictions are false-positives. On the other hand, this implies that the other half of the predicted loci should be real functional RNA structures, either as part of a ncRNA or as regulatory element of a mRNA. However, one always have to bear in mind possible shortcomings of this kind of random control (*see Note 6*).

4. Notes

1. **Custom installation of RNAz.** The installation process using `./configure` and `make` should work on all UNIX-like systems. If there is an error messages,

it may be necessary to install additional “developer packages.” On some Linux distributions, for example, there is no C-compiler installed by default. Also, on OS X it is necessary to install the “XCode” tools.

If the user does not have root privileges or wants to install RNAz into a different location than `/usr/local/` (e.g., the home directory) the following command can be used:

```
#. /configure --prefix=/home/stefan \  
--datadir=/home/stefan/share
```

This installs the executable to `/home/stefan/bin` and the example files, Perl programs, and other data to `/home/stefan/share/RNAz`. Please note that the `bin` directory must be in the `PATH` of executables if they are to be called the RNAz executable without specifying the complete path.

2. **Running the Perl programs.** Because different people usually like to have their scripts in different locations, the Perl programs are not installed to `/usr/local/bin` by default. They are installed to `/usr/local/share/RNAz/perl`. To make them available from other locations, copy all files from this directory to a directory, which is included in the `PATH` of executables, e.g.:

```
# cp /usr/local/share/RNAz/perl/* /usr/local/bin
```

Alternatively, the directory with the Perl programs can be added to the `PATH` variable by editing the `.bashrc` or `.cshrc` file in the home directory.

In any case, it is important that the Perl module file `RNAz.pm` resides in the same directory as the Perl programs (`*.pl`). All the Perl programs depend on this module file.

Another important point is that the Perl programs expect that the path of the Perl executable is `/usr/bin/perl`. This is the standard location on almost all Linux/UNIX systems and OS X. If the Perl installation is different, the user has to customize the first line of all the Perl programs according to the location of the `perl` executable.

On a Windows system the Perl programs should work if Perl has been installed as described in **Subheading 2., Point 3.** and the `Path` variable has been set as described in **Subheading 3.1.2.**

3. **Optional software on Windows on OS X. It might be a bit tricky** to install the necessary software on Windows and OS X for the programs in **Subheadings 3.6.8.** and **3.6.10.**

The Vienna RNA package and NCBI Blast can be installed on OS X without problems by following the instructions in **Subheading 3.2.** However, unlike on a Linux system, Ghostscript is not installed per default. A precompiled package can be obtained from fink.sourceforge.net or darwinports.opendarwin.org.

Alternatively, the source can be downloaded from <http://www.ghostscript.com/> and the package can be built with `./configure` and `make`.

Ghostscript can be installed on Windows through a simple installer file that can be downloaded from <http://www.ghostscript.com/>. Follow the installation instructions. Locate the newly installed file `gswin32c.exe` and copy it to a folder, which is in the `Path` (e.g., the folder, where the `RNAz.exe` executable resides). Rename the file to `gs.exe`.

Windows users do not have to install the Vienna RNA package. The relevant programs are part of the RNAz Windows installer.

To install NCBI blast on windows, create a new folder (e.g., `c:\Program Files\blast`) and download the `blast-2.*-win32.exe` file from <ftp://ftp.ncbi.nih.gov/blast>. Within the new folder, double click on the `blast-2.*-win32.exe` file, which extracts the programs and data. Add the `bin` subdirectory to the `Path`: right-click “My computer”, then click “Properties”. Select “Advanced/ Environment variables/New”. Add the complete path of the blast bin directory to the variable `Path`, use “;” as separator.

4. **Creating the input alignments.** RNAz can only detect a conserved structure if this structure is accurately reflected in the alignment. Therefore, the quality of the alignment is crucial for the success of the analysis. In practice, we found that if the alignment has a mean pairwise identity more than approx. 60% simple sequence-based progressive, global alignment methods yield reasonable results and there is not much difference between methods. One of the best programs for aligning RNAs is Clustal W. For genome-wide alignments we have only experience with Multiz alignments. Also these alignments are of reasonable quality and there is generally no need for realignment. We suppose that also other genome-wide alignment methods produce suitable alignments as long the aligned regions are of sufficient similarity (mean pairwise identity somewhere around 60% or higher). In cases with sequences less than 60% identity, simple sequence-based methods usually do not find an optimal structural alignment. Although, in principle, structural enhanced alignments could help here, this alternative is not relevant in practice. First, there are hardly any structural multiple sequence alignment programs available. Second, current approaches are much too slow to use them for every day analysis. Third, RNAz is not trained on structural alignments. In contrast to pure sequence based alignment, the user would get unusual high SCIs. This could confuse the decision model and the user would get unpredictable results.
5. **Additional output values.** The consensus MFE, which is calculated by the RNAalifold algorithm (see **Subheading 1.2.2.**), can be split in two terms. One is the “energy contribution,” which is the folding energy from the standard energy model. The “covariance contribution” is the part that comes from the additional “bonus” or “penalty” energies for compensatory/consistent and inconsistent mutations, respectively. If the covariance term is negative, there are more compensatory mutations than inconsistent mutations.

RNAz also calculates another value quantifying compensatory/consistent mutations: “Combinations/Pair.” This is the number of different basepair combinations in the consensus structure divided by the number of pairs in the consensus structure. Both the covariance contribution of the consensus MFE and the “Combinations/Pair” are mainly useful for final sorting a set of equally good predictions that have been filtered using other criteria (e.g., p - or z -scores).

RNAz uses a SVM algorithm for classification. The raw output of the SVM is the so-called “decision-value.” This real-valued number is positive if the prediction is “RNA” and negative otherwise. From this value we calculate the more intuitive “RNA class probability” or “ p -value,” which is 0.5 for a decision value of 0. In some cases, the raw decision value can be more convenient than the p -value (e.g., to plot the distribution of RNAz results).

6. **Estimating false-positives.** The RNAz classification model is trained on a test set consisting of natural RNAs as positive examples and randomly shuffled alignments as negative examples. Thus, any signal reported by RNAz is relative to an artificial background. Although this null model of shuffled sequences is probably the most sensible choice possible, one cannot assume that it behaves exactly like the natural background of real sequence data. Also the estimation of false-positive rates is based on shuffled sequences. We want to stress that, therefore, such an estimation of false-positives must be regarded as a lower bond because one cannot rule out the possibility that nonrandom patterns in natural sequences cause a higher rate of false-positives than one observes in synthetic random sequences. In particular, the z -score calculation might be affected by such effects. For example di-nucleotide content could bias the MFE structure prediction. As an opposite effect one must consider the possibility that the shuffling procedure cannot remove all secondary structure signals and, thus, overestimates the real false-positive rate. If an alignment is shuffled with many compensatory mutations, the number of “compatible columns” remains the same, allowing for compensatory mutations also in the shuffled alignment.
7. **Manual inspection of candidates.** If the user has a hit with $p > 0.9$, there is approximately a chance of 1 in 100, that this arises through pure chance (but *see also Note 6*). It makes sense to critically look at a hit. Sometimes the signal only comes from a low z -score of borderline significance and there is no evidence for structural conservation. Sometimes the complete alignment looks pathological (weird gap patterns, low complexity regions, and so on), which suggests that this is not a relevant structure. It is useful to analyze a predicted structure with RNAalifold and its visualization methods (*see Chapter 33 in this book*). Visual inspection of a color-coded alignment and the consensus structure gives an idea about compensatory mutations supporting the structure and inconsistent mutations, which do not support the structure. It must be noted that many ncRNAs in real life-data are not supported by compensatory mutations, still they can be detected based on the stability or the SCI. The SCI implicitly also considers the mutational

pattern outside of stems. To conclude, the p -value efficiently filters the data for candidates, but only the complete picture can help in the decision on the relevance of a hit.

8. **Advanced filtering.** Filtering the tab-delimited data files using standard UNIX tools like `grep` or `awk` is difficult because of the special window/locus grouping of the data. The `rnazFilter.pl` program can be used. The filter statement uses the field names (e.g., `z`, `SCI`, `combPerPair`, see `--help` for a complete list) and standard logical operators as used in the Perl language: `>` (greater than), `<` (smaller than), `==` (equals numerically), `eq` (equals string), `not`, `and`, `or`, `~/regex/` (pattern match). In addition, brackets can be used to group and combine statements. For example the following statement gives all windows with $p > 0.9$ and $z < -3$ on chromosome 13:

```
# rnazFilter.pl "P>0.9 and z<-3 and
seqID=~chr13/"\results.dat
```

It is important that everything put in the filter statement is evaluated by the Perl interpreter. This can be potentially harmful, so take care.

Acknowledgments

The author thanks Ivo L. Hofacker and Peter F. Stadler for helpful discussions and assistance during the development of RNaz. This work was supported by Austrian GEN-AU project “noncoding RNA.”

References

- 1 Frith, M. C., Pheasant, M., and Mattick, J. S. (2005) The amazing complexity of the human transcriptome. *Eur. J. Hum. Genet.* **13**, 894–897.
- 2 Martens, J. A., Laprade, L., and Winston, F. (2004) Intergenic transcription is required to repress the *Saccharomyces cerevisiae* *SER3* gene. *Nature* **429**, 571–574.
- 3 Chooniedass-Kothari, S., Emberley, E., Hamedani, M. K., et al. (2004) The steroid receptor RNA activator is the first functional RNA encoding a protein. *FEBS Lett* **566**, 43–47.
- 4 Rivas, E. and Eddy, S. R. (2001) Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics* **2**, 8.
- 5 Rivas, E., Klein, R. J., Jones, T. A., and Eddy, S. R. (2001) Computational identification of noncoding RNAs in *E. coli* by comparative genomics. *Curr. Biol.* **11**, 1369–1373.
- 6 McCutcheon, J. P. and Eddy, S. R. (2003) Computational identification of non-coding RNAs in *Saccharomyces cerevisiae* by comparative genomics. *Nucleic Acids Res.* **31**, 4119–4128.

- 7 Washietl, S., Hofacker, I. L., and Stadler, P. F. (2005) Fast and reliable prediction of noncoding RNAs. *Proc. Natl. Acad. Sci. USA* **102**, 2454–2459.
- 8 Zuker, M. and Stiegler, P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* **9**, 133–148.
- 9 Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. (1994) Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.* **125**, 167–188.
- 10 Hofacker, I. L., Fekete, M., and Stadler, P. F. (2002) Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.* **319**, 1059–1066.
- 11 Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673–4680.
- 12 Gardner, P. P., Wilm, A., and Washietl, S. (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res.* **33**, 2433–2439.
- 13 Steigele, S., Huber, W., Stocists, C., Stadler, P. F., and Nieselt, K. (2007) Comparative Analysis of Structured RNAs in *S. cerevisiae* Indicates a Multitude of Different Functions. *BMC Genomics*, in press.
- 14 Hofacker, I. L., Priwitzer, B., and Stadler, P. F. (2004) Prediction of locally stable RNA secondary structures for genome-wide surveys. *Bioinformatics* **20**, 186–190.
- 15 Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R., and Bateman, A. (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.* **33**, D121–D124.
- 16 Eddy, S. R. (2002) A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics* **3**, 18.
- 17 Washietl, S. and Hofacker, I. L. (2004) Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *J. Mol. Biol.* **342**, 19–30.

RNA Consensus Structure Prediction With RNAalifold

Ivo L. Hofacker

Summary

The secondary structure of most functional RNA molecules is strongly conserved in evolution. Prediction of these conserved structures is therefore of particular interest when studying noncoding RNAs. Moreover, structure predictions on the basis of several sequences produce much more accurate results than energy directed folding of single sequences.

The RNAalifold program predicts the consensus structure for a set of aligned sequences taking into account both thermodynamic stability and sequence covariation. In this contribution, we provide a tutorial on how to install and use RNAalifold, as well as a guide on how to interpret the results.

Key Words: RNA structure prediction; RNA secondary structure; consensus structure; structural noncoding RNA.

1. Introduction

1.1. *Functional RNAs and Structure Conservation*

A fundamental tenet of molecular biology is that the function of biological macromolecules depends on their structure. As a consequence, these structures tend to be more strongly conserved in evolution than their corresponding sequences. In the case of RNA, structure is most accessible on the level of “secondary structure,” i.e., the pattern of basepairings. A striking example of such structural conservation are the tRNAs, where almost all tRNA sequences, be they from animals, plants, or bacteria, fold into the characteristic “cloverleaf” secondary structure. Similar structural conservation is found for most of the

classical noncoding RNAs, such as 16S, 23S, and 5S rRNAs, RNase P, tmRNA, or group I and II introns. When enough sequences are available, this makes it possible to infer the conserved structure solely on the basis of sequence covariation. In the case of ribosomal RNAs, the secondary structure models inferred by comparative sequence analysis proved to be highly accurate when compared to the tertiary structure of the ribosome as resolved by X-ray crystallography (1).

In recent years the list of functional noncoding RNAs (ncRNAs) has increased vastly, and consequently efficient methods to predict their conserved structures are of great interest. Moreover, the presence of a well-conserved structure is often the best indication that a particular sequence does indeed function as a ncRNA. Consensus structure prediction, therefore, lies at the core of many approaches to detect ncRNAs in genomic sequences.

In typical situations, however, the number of sequences available is too small to infer the common structure from sequence covariations alone. Instead, one must turn to methods that combine the phylogenetic information with energy-based structure prediction methods. Approximately, one may divide such approaches into three broad classes. The most common approach, exemplified by RNAalifold (2) discussed here and pfold (3), starts from a given sequence alignment. The converse approach is taken, e.g., by MARNAs (4) and RNAforester (5), which start from predicted structures and then align the structures. The most rigorous (but also computationally most expensive) approach is to compute alignment and consensus structure simultaneously, as in the Sankoff algorithm (6). For a recent comparison of the various techniques see ref. 7.

1.2. The RNAalifold Method

The thermodynamics of RNA secondary structures can be well described by an energy model that assigns an energy to every loop (and stacked pairs) in the structure. The corresponding energy parameters have been deduced from melting experiments on RNA oligomers and are summarized in ref. 8. Within this loop-based energy model, the structure of minimum free energy (MFE) can be computed efficiently via dynamic programming (9).

RNAalifold generalizes this algorithm for sequence alignments, treating the entire alignment as a single “generalized sequence.” To assign an energy to a structure on such a generalized sequence, the energy is simply averaged over all sequences in the alignment. This average energy is augmented by a covariance term, that assigns a bonus or penalty to every possible basepair (i, j) based on the sequence variation in columns i and j of the alignment.

Sequence covariations are a direct consequence of RNA basepairing rules. RNA helices normally contain only 6 out of the 16 possible combinations: the Watson-Crick pairs GC, CG, AU, UA, and the somewhat weaker wobble pairs GU and UG. Mutations in helical regions therefore have to be correlated. In particular, we often find “compensatory mutations” where a mutation on one side of the helix is compensated by a second mutation on the other side, e.g., a C-G pair changes into a U-A pair. Mutations where only one pairing partner changes (such as C-G to U-G) are termed “consistent mutations.”

Compensatory mutations are a strong indication of structural conservation, whereas consistent mutations provide a weaker signal. The covariance term used by RNAalifold therefore assigns a bonus of 1 kcal/mol to each consistent and 2 kcal/mol for each compensatory mutation. Sequences that cannot form a standard basepair incur a penalty of -1 kcal/mol. Thus, for every possible consensus pair between two columns i and j of the alignment a covariance score is computed by counting the fraction of sequence pairs exhibiting consistent and compensatory mutations, as well as the fraction of sequences that are inconsistent with the pair. The weight of the covariance term relative to the normal energy function, as well as the penalty for inconsistent mutations can be changed via command-line parameters.

Apart from the covariance term, the folding algorithm in RNAalifold is essentially the same as for single sequence folding. In particular, folding an alignment containing just one sequence will give the same result as single sequence folding using RNAfold. For N sequences of length n the required CPU time scales as $O(Nn^2+n^3)$ whereas memory requirements grow as the square of the sequence length. Thus, RNAalifold is in general faster than folding each sequence individually. The main advantage, however, is that the accuracy of consensus structure predictions is generally much higher than for single sequence folding, where typically only between 40 and 70% of the basepairs are predicted correctly (10).

Apart from prediction of MFE structures RNAalifold also implements an algorithm to compute the partition function over all possible (consensus) structures and the thermodynamic equilibrium probability for each possible pair. These basepairing probabilities are useful to see structural alternatives, and to distinguish well-defined regions, where the predicted structure is most likely correct, from ambiguous regions.

1.3. General Remarks and Typographical Conventions

There is no graphical user interface for RNAalifold, instead all steps are carried out on a command-line (terminal). Most programs discussed here read

from standard in and write to standard out. This allows the programs to be used as filters and allows chaining several programs in pipes. Users that only need to do the occasional structure prediction may also see the web interface for **RNAalifold** at <http://rna.tbi.univie.ac.at/>. Here, we will concentrate on using the more flexible command-line tools.

We use **constant width font** for program names, variable names, and other literal text, for example input and output in the terminal window. Lines starting with a **\$** within a literal text block are commands. Type the text following the **\$** into the terminal window finishing by hitting the return-key. (The **\$** signifies the command-line prompt, which may not look the same on different systems.) All other lines within a literal text block are the output from the command recently typed.

2. Materials

1. **Hardware.** **RNAalifold** should run on any hardware for which a C compiler is available. CPU and memory requirements depend on the length of the alignment and, to a lesser extent, the number of sequences in the alignment. Thus, the available memory limits the maximum sequence length that can be folded. For small RNAs even an obsolete Pentium PC will be sufficient, whereas long sequences, such as complete viral genomes, may require large amounts of memory (e.g., about 1 GB for a HIV genome).
2. **Operating system.** We recommend using Linux for bioinformatics work, because a typical installation will have all necessary tools present. In our case, these tools are: a compiler for the C programming language, the Perl scripting language, and a viewer for postscript files such as **gsview** or **gv**. Commercial Unix-like operating systems are also a good choice. This includes Mac OS X, which builds upon a Unix-style operating system. Development tools, however, are not installed by default on Mac OS X, these must be downloaded and installed separately.

Windows users may install CygWin from <http://www.cygwin.com/>, which provides a complete GNU environment running under Windows. The CygWin environment allows the user to compile and run all programs exactly as under Linux/Unix.

Alternatively, one may download the precompiled executables for windows, and separately install Perl and **gsview** (see **Notes 1** and **2**).

3. **Software.** The **RNAalifold** program is part of the Vienna RNA package, which can be downloaded from <http://www.tbi.univie.ac.at/~ivo/RNA/>. The package is normally distributed as source code, see installation instructions next. Precompiled binaries for Windows can be downloaded from <http://www.tbi.univie.ac.at/~ivo/RNA/windoze/>.
4. **Optional software.** **clustalw** or **clustalx** are popular programs for performing multiple sequence alignments. They are available at <http://bips>.

u-strasbg.fr/en/Documentation/ClustalX/. Alternatively, there are several websites where Clustal alignments can be performed online.

Some of the supplied Perl scripts use the Tk library and Perl bindings for visualization purposes, the Perl/Tk module is available from CPAN. It can be installed by typing

```
$ perl -MCPAN -e 'install Tk'
```

Data files for the examples used in this chapter are available in the electronic supplements at <http://www.tbi.univie.ac.at/papers/SUPPLEMENTS/MiMB/>

3. Methods

3.1. Installing RNAalifold and the Vienna RNA Package

3.1.1. Installing From Source

On Unix-like operating systems, such as Linux, SunOS, but also Mac OS X, we recommend installing the complete Vienna RNA Package, which includes RNAalifold, from source code. Simply download the latest version from <http://www.tbi.univie.ac.at/~ivo/RNA/> and follow the steps next:

1. Unpack the tar file by running

```
$ gunzip ViennaRNA-1.6.1.tar.gz
$ tar -xvf ViennaRNA-1.6.1.tar
```

2. To configure, build, and install the package just run

```
$ cd ViennaRNA-1.6.1
$ ./configure
$ make all
$ make install
```

To run the last command, which installs the main programs of the Vienna RNA Package into the default location (`/usr/local/bin/`), the user will need superuser (root) privileges. In addition, several of the scripts and example programs are installed into the directory `/usr/local/share/ViennaRNA/bin/`.

The installation location can be controlled through options to the `configure` script. If the user does not have root privileges, the user should install a directory within the home. For example, to install in the `$HOME/RNA` directory, use:

```
$ ./configure --prefix=$HOME/RNA
```

More detailed installation instructions can be found in the `INSTALL` file in the distribution, as well as on the website. If a nonstandard location is installed, add the directory containing the executables to the shell's search path.

3.1.2. Program Documentation

All programs in the Vienna RNA Package are documented in “man pages” that can be called up using the `man` command, e.g.,

```
$ man RNAalifold
```

Most Perl scripts carry embedded documentation that is displayed by typing, e.g.,

```
$ perl doc coloraln.pl
```

All scripts and programs give a short usage message when called with the `-h` command-line option (e.g., `RNAalifold -h`).

3.2. Using RNAalifold

3.2.1. Basic Usage

As a first example, we will produce a consensus structure prediction for the following three tRNA sequences.

```
$ cat three_tRNAs.seq
>M10740 Yeast-PHE
GCGGAUUUAGCUCAGUUGGGAGAGCGCCAGACUGAAGAUUUGGAGGUCCUGUGUUCGA
UCCACAGAAUUCGCA
>K00349 Drosophila-PHE
GCCGAAAUAGCUCAGUUGGGAGAGCGUUAGACUGAAGAUCUAAAGGUCCCCGGUUCAA
UCCCGGGUUUCGGCA
>K00283 Halobacterium volcanii Lys-tRNA-1
GGGCCGGUAGCUCAUUUAGGCAGAGCGUCUGACUCUUAUUCAGACGGUCGCGUGUUCG
AAUCGCGUCCGGCCCA
```

`RNAalifold` uses aligned sequences as input. Thus, our first step will be to align the sequences. We use `clustalw` in this example, because it is one of the most widely used alignment programs and has been shown to work well on structural RNAs ([11](#)). Other alignment programs can be used, including programs that attempt to do structural alignment of RNAs (see **Note 3**), but the resulting multiple sequence alignment must be in `Clustal` format.


```
$ clustalw three_tRNAs.seq
```

This produces a multiple sequence alignment in the file `three_tRNAs.aln`. Next, we compute the consensus structure from the alignment.

```
$ RNAalifold three_tRNAs.aln
3 sequences; length of alignment 74.
GCGGAAAUAGCUCAGUUGGG_AGAGCGUCAGACUGAAGAUCUGAAGGUCCCCGUGUUCG
AUCCACGGAAUCCGCA
(((((((..((((.....))))).((((.....))))). ....((((.....
...)))))))).
minimum free energy = -31.28 kcal/mol (-24.63 + -6.65)
```

The output contains a consensus sequence and the consensus structure in bracket notation. The bracket notation is convenient string representation for secondary structures. It encodes the structure as a string of dots “.” and matching brackets “()”, such that unpaired positions are symbolized by a “.” whereas a basepair (i, j) is denoted by a “(” at position i and a matching “)” at position j . The consensus structure has an energy of -31.28 kcal/mol, which in turn consists of the average free energy of the structure -24.63 kcal/mol and the covariance term -6.65 kcal/mol. The strongly negative covariance term shows that there must be a fair number of consistent and compensatory mutations, but in contrast to the average free energy it is not meaningful in the biophysical sense.

The predicted structure in this example is in fact 100% correct. On the other hand, if we try to predict the structure using only the first (*Escherichia coli*) sequence we get a rather disappointing result:

```
$ head -2 three_tRNAs.seq | RNAfold
>M10740 Yeast-PHE
GCGGAUUUAGCUCAGUUGGGAGAGCGCCAGACUGAAGAUUUGGAGGUCCUGUGUUCGA
UCCACAGAAUUCGCA
(((((((..(.....((((((((..((((.....))))).)))))))). ..
))))).)))))))). (-21.80)
```

Here, only the outermost helix, the acceptor stem of the tRNA structure, is present, all other predicted basepairs are wrong. The `head -2` command previously described, prints the first two lines of our sequence file, containing the first sequence and its name.

RNAalifold automatically produces a drawing of the consensus structure in postscript format and writes it to the file “alrna.ps.” The resulting structure can be seen in [Fig. 1](#). In the structure graph, consistent and compensatory

mutations are marked by a circle around the variable base(s), i.e., pairs where one pairing partner is encircled exhibit consistent mutations, whereas pairs supported by compensatory mutations have both bases marked. Pairs that cannot be formed by some of the sequences are shown gray instead of black.

Note, that subsequent calls to `RNAalifold` will overwrite any existing output “alirna.ps” (“alidot.ps,” “alifold.out”) files in the current directory. Be sure to rename any files that should be kept.

3.2.2. More Colorful Structure Representations

More detailed information can be obtained by computing not only the MFE structure, but also pair probabilities. This is accomplished by adding the `-p` option to the command-line.

The example next also uses the `-mis` command-line, which changes how the consensus sequence is generated. Usually the program simply takes the most frequent nucleotide in each column of the alignment. With the `-mis` option, we instead produce the “most informative sequence” (12). Given the background frequency of each nucleotide (averaged over the whole alignment), we find for each column the set of characters that are overrepresented, i.e., more frequent than in the background. This set is then represented by the corresponding IUPAC code (e.g., R for the purines A and G, Y for C or U).

```
$ RNAalifold -mis -p three_tRNAs.aln
3 sequences; length of alignment 74.
GSSSMDDUAGCUCAKUURGGcAGAGCGYYWGACUSWWRAUYWRRMGGUCSYSKGUUCR
AWYCVSRKHKBSSCA
(((((((..((((.....))))).((((.....))))). ....((((.....
...)))))))).
minimum free energy = -31.28 kcal/mol (-24.63 + -6.65)
(((((((..((((.....))))).((((.....))))). ....((((.....
...)))))))).
free energy of ensemble = -31.69 kcal/mol
frequency of mfe structure in ensemble 0.514564
```

This time we have produced two additional output files apart from the “alirna.ps” file. The file “alidot.ps” contains the pair probabilities in the form of a dot plot. A basepair (i, j) that is predicted with probability p is represented by a square at row i and column j of the dot plot with area p . The lower left half of the plot show only pairs present in the MFE structure, whereas the upper right half shows alternative foldings as well. In our tRNA example, the structure is very well defined resulting in only few alternative pairs with low

probability (see [Fig. 1](#)). The dot plots produced by RNAalifold use color to convey information on sequence variations. The color hue encodes how many of the six possible types of basepairs (GC, CG, AU, UA, GU, UG) occur in at least one of the sequences. Pairs without sequence covariation are shown in red, if two types of pairs occur the square is ochre. Green, turquoise, blue, and violet mark pairs that occur in three, four, five, and six types of pairs, respectively. Unsaturated (pale) colors mark pairs that cannot be formed by all sequences.

The same color annotation can be transferred to normal structure drawing using the `colorrna.pl` utility (see also **Note 4**), e.g.:

```
$ colorrna.pl alirna.ps alidot.ps > tRNA_color.ps
```

Yet another useful representation of RNA structures is the mountain plot. In the mountain plot a pair (i, j) is represented by a colored trapez with baseline from position i to position j and height proportional to the probability. The command

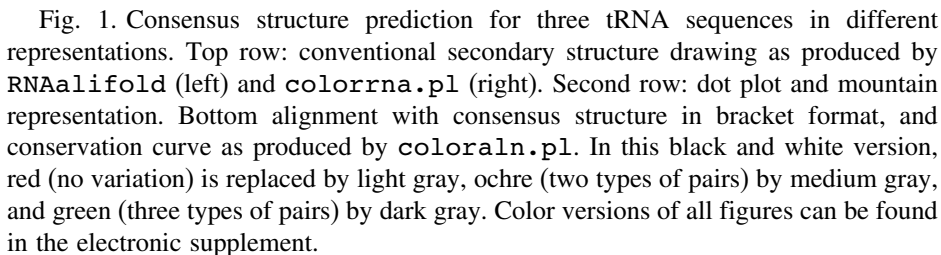
```
$ cmount.pl alidot.ps > tRNA_cmt.ps
```

will produce a color mountain plot from the dot plot in “alidot.ps” (see second row of [Fig. 1](#)).

3.2.3. Using the AliDot Viewer

The last output file produced by RNAalifold `-p`, named “alifold.out,” is a plain text file with detailed information on all plausible basepairs sorted by the likelihood of the pair.

```
$ head alifold.out
3 sequence; length of alignment 74
alifold output
  4 70 0 100.0% 0.000 CG:1 GC:1 GU:1
53 63 0 100.0% 0.001 GC:1 UG:1 UA:1
  6 68 0 99.8% 0.008 GC:1 AU:1 UA:1
  2 72 0 100.0% 0.000 CG:2 GC:1
  3 71 0 100.0% 0.000 CG:1 GC:2
52 64 0 100.0% 0.000 CG:1 GC:2
  5 69 0 100.0% 0.001 CG:1 AU:2
51 65 0 99.9% 0.003 CG:2 UA:1
```



In the previously described example we see that the pair (4,70) has no inconsistent sequences, is predicted with probability 1, and occurs as a CG pair in one sequence, a GC pair in another, and a GU pair in the third. The `AliDot.pl` utility uses this information to display a dot plot equivalent to the postscript version. The viewer has better zoom capabilities than most postscript previewers and shows additional information, but requires the Perl Tk module to be installed.

Start the viewer using `AliDot.pl alifold.out`, and a canvas will open showing the dot plot. The “+” and “-” keys can be used to zoom in and out. The coordinates of the basepair below the mouse pointer is indicated in the upper left corner. Clicking on any basepair will display more detailed information including the probability of the pair, the number of sequences unable to form the pair, and the observed basepair types. A screenshot is shown in [Fig. 2](#).

3.3. Advanced Usage

3.3.1. Structure Predictions for the Individual Sequences

The consensus structure computed by `RNAalifold` will contain only pairs that can be formed by most of the sequences. The structures of the individual sequences will typically have additional basepairs that are not part of the consensus structure. Moreover, ncRNA may exhibit a highly conserved core structure whereas other regions are more variable. It may, therefore, be desirable to produce structure predictions for one particular sequence, while still using covariance information from other sequences.

This can be accomplished by first computing the consensus structure for all sequences using `RNAalifold`, then folding individual sequences using `RNAfold -C` with the consensus structure as a constraint. In constraint folding mode `RNAfold -C` allows only basepairs to form which are compatible with the constraint structure. This resulting structure typically contains most of the constraint (the consensus structure) plus some additional pairs that are specific for this sequence.

Before the constraint folding step one should of course remove any gaps from the sequence, as well as remove the corresponding positions from the consensus structure. To help with this tedious task, there is a Perl script called `refold.pl` available in the electronic supplement. In the example, we will produce a structure prediction for RNase P from *E. coli* using an alignment of 11 sequences, taken from Bralibase <http://www.binf.ku.dk/~pgardner/bralibase/>.

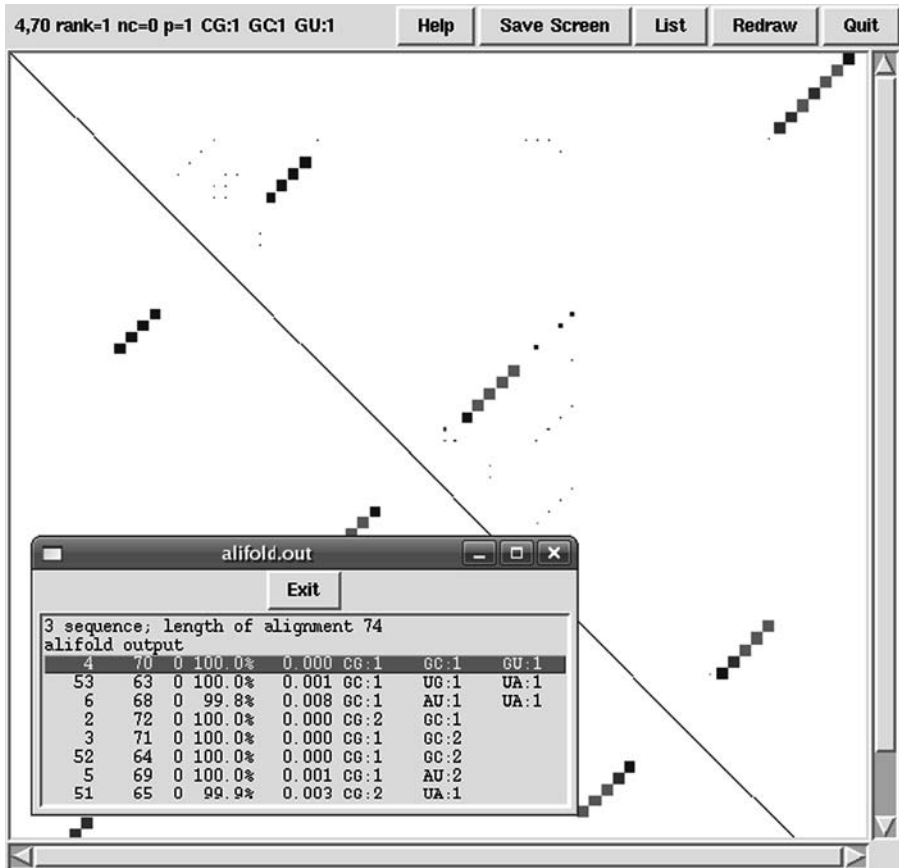


Fig. 2. Screenshot of the AliDot.pl viewer (black and white version).

```
$ RNAalifold RNaseP.aln > RNaseP.alifold
$ refold.pl RNaseP.aln RNaseP.alifold | head -3 > RNaseP.cfold
$ RNAfold -C -noLP < RNaseP.cfold > RNaseP.refold
```

Comparing the consensus structure with the *E. coli* reference structure we find that 73 out of 91 predicted basepairs are correct, but we are nevertheless missing $49/122 \approx 40\%$ of the basepairs in the reference structure. After refolding we have 90 correctly predicted pairs (17 more) and 20 false predictions (vs 18 before). Note that because RNase P forms sizable pseudo-knots, a perfect prediction is impossible in this case.

If constrained folding results in a structure that is very different from the consensus, or if the energy from constrained folding is much worse than from unconstrained folding, this may indicate that the sequence in question does not really share a common structure with the rest of the alignment or is misaligned. One should then either remove or realign that sequence and recompute the consensus structure.

3.3.2. Conserved Structure Motifs in Long Sequences

Longer sequences will often exhibit several short conserved structure motifs separated by regions without conserved structure. In this case, it is recommended to rerun RNAalifold on just the conserved regions.

Identify the conserved region from the dot plot or mountain plot and write a new alignment file for each of them. The `clustalx` program is convenient for cutting a region out of an alignment, but a simple text editor can be used as well.

In the case of very long sequences, one may use a new variant of the program that computes locally conserved structure motifs in a long alignment, in a manner similar to the RNALfold program for single sequences (13).

```
$ RNALalifold -L 100 long.aln
```

The parameter `-L` specifies the maximum size of the conserved structure motif. The output consists of a list of short consensus structure together with their location in the alignment. To obtain the usual structure drawings and other output, simply rerun RNAalifold on each of the regions found by RNALalifold.

3.3.3. Detection of ncRNAs

In the last example, we used RNAalifold essentially to detect structural RNAs within long aligned sequences. The method does not, however, provide a good measure for the statistical significance of predicted consensus structures. For example, it is difficult to decide whether a predicted structure is functionally important or just incidental.

One way of generating such a significance measure is to randomize the alignment through shuffling and compare the MFE of the original alignment with a large sample of randomized alignments (14). If the MFE of the native alignment is significantly better, then we have likely a bona fide functional structure.

An even more sophisticated approach is given by the RNAz program, which is described in Chapter 32.

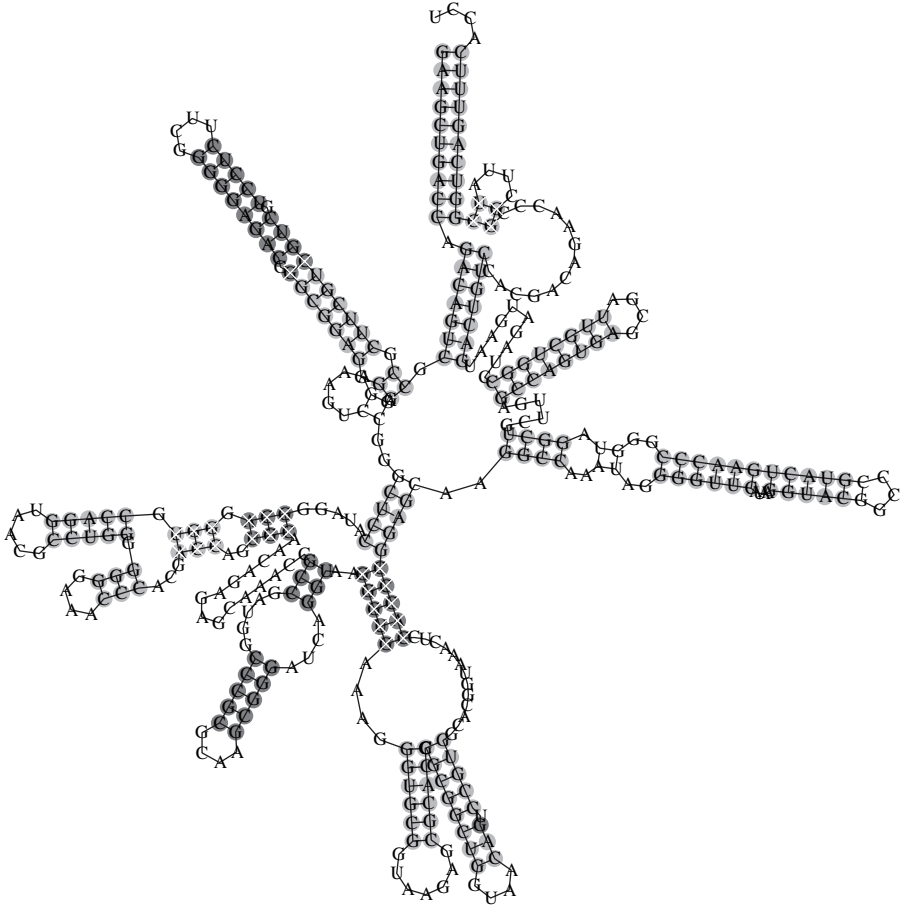


Fig. 3. Structure for *Escherichia coli* RNase P as predicted after refolding. Basepairs with light background are already present in the consensus structure, pairs with dark background are added by the refolding step. Incorrect basepairs are “crossed out” in white.

3.4. Caveats

Although consensus structure predictions tend to be much more accurate than single sequence predictions, a number of circumstances may lead to unexpectedly poor results.

1. RNAalifold predicts pseudo-knot free structures only. Thus, RNAs containing many pseudo-knots may be predicted poorly, try, e.g., [hxmatch \(15\)](#) in such cases.

2. For reasons of efficiency, gaps are not discarded in the calculations of loop energies. Thus, a long insertion that is present in only one of many sequences produces a large energetically unfavorable loop in the consensus structure. Manually removing columns that are gaps in almost all sequences can improve prediction accuracy in these cases.
3. Ideally all sequences in the alignment should have similar distances from each other. If the alignment contains a large group of very similar sequences and only few dissimilar ones, the consensus structure will be biased toward the large group.
4. If sequences are very similar (more than 95% sequence identity), there will be few covariations. Consequently, prediction accuracy will be no better than for single sequences.
5. RNAalifold is limited by the accuracy of the input alignment. For sequences with pairwise identities less than approx 60%, sequence alignments (as produced by clustalw) start to differ significantly from structurally correct alignments.
6. RNAalifold is not well tuned for alignments containing hundreds of sequences. A mutual information score will probably work better for such alignment than the alifold covariance score.

4. Notes

1. **Microsoft Windows.** To run Alifold (and any other programs of the Vienna RNA package) natively on Microsoft Windows, download the installer package `ViennaRNA-1.6-win32.msi`. Double click on the file and follow the instructions. The package will install all necessary executables and Perl scripts on the system. It also sets the correct Path variable and all the programs should be able to be run immediately on a console window.

To run the Perl programs, install the Perl interpreter from www.activestate.com. Choose the latest ActivePerl MSI installer package for Windows and simply follow the installation instructions. Be sure to select the “Add Perl to the PATH environment variable” and “Create Perl file extension association” options during installation.

To view postscript files, ghostscript (www.ghostscript.com) and GSview (<http://www.cs.wisc.edu/~ghost/gsview/>) need to be installed. Get the latest Windows packages and follow the installation instructions.

2. **Viewing postscript files under Windows and OS X.** On Windows, a postscript file can be opened on the command-line by simply typing the name of the file, e.g.:

```
$ alirna.ps
```

This will open GSview. If there is a “Document Structuring convention Error,” the error can be safely ignored by clicking “Ignore all DSC.” Postscript files can be easily converted to other formats through the File/Convert menu. Note however, that postscript (and pdf) files contain resolution independent vector graphics that

will print in much better quality than bitmap formats, such as tiff, jpeg, or gif.

Unlike most Linux systems, Ghostscript is not installed per default on OS X. To view a postscript file, simply click on the file in a Finder window. This will automatically convert the postscript to a PDF file and display it. Alternatively, the user can try to install ghostscript. Either get a precompiled package (e.g., from <http://fink.sourceforge.net/>) or build the package from the sources available at <http://www.ghostscript.com/>.

3. **Structural alignment of RNAs.** A variety methods for structural alignments of RNA sequences have been proposed lately. Most of these methods are very recent, and a consensus of which approach is most suitable is still lacking. In general sequence-based alignments work well for sequence similarity more than 60%. We, therefore, recommend using structural alignment programs only in the low homology region.

Notable programs for structural alignments include *foldalign* (16), *dynalign* (17), *stemloc* (18), and *pmcomp* / *pmmulti* (19), which however are suitable only for relatively short (<200 nt) alignments, see ref. 11 for a recent comparison.

4. Running the Perl programs. Although RNAalifold itself is installed in `/usr/local/bin` by default, the Perl scripts are installed to `/usr/local/share/ViennaRNA/bin`. This location is usually not in the search PATH of the shell. To avoid typing long pathnames such as `/usr/local/share/ViennaRNA/bin/colorrna.pl` the user can either copy all files to a directory, which is included in the PATH, e.g.:

```
$ cp /usr/local/share/ViennaRNA/bin/* /usr/local/bin
```

or add the script location to the PATH environment variable. For Bourne shells, such as `bash` type (assuming the default install location)

```
$PATH={PATH}:/usr/local/share/ViennaRNA/bin
```

```
export PATH
```

for C-shells (`csh`, `tcsh`) use

```
$ setenv PATH ${PATH}:/usr/local/share/ViennaRNA/bin
```

To permanently add alter the PATH add the previously listed commands to the `.bashrc` or `.cshrc` file in the home directory.

On a Windows system the Perl programs should work if Perl is installed and set the Path variable as previously described.

References

- 1 Gutell, R. R., Lee, J. C., and Cannone, J. J. (2002) The accuracy of ribosomal RNA comparative structure models. *Curr. Opin. Struct. Biol.* **12**, 301–310.
- 2 Hofacker, I., Fekete, M., and Stadler, P. (2002) Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.* **319**, 1059–1066.
- 3 Knudsen, B. and Hein, J. (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucl. Acids Res.* **31**, 3423–3428.

- 4 Siebert, S. and Backofen, R. (2005) MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics* **21**, 3352–3359.
- 5 Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S. (2003) Local similarity in RNA secondary structures. *Proc. of the Computational Systems Bioinformatics Conference, Stanford, CA, August 2003 (CSB 2003)*, pp. 159–168.
- 6 Sankoff, D. (1985) Simultaneous solution of the RNA folding, alignment, and proto-sequence problems. *SIAM J. Appl. Math.* **45**, 810–825.
- 7 Gardner, P. P. and Giegerich, R. (2004) A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatic* **5**, 140.
- 8 Mathews, D., Sabina, J., Zuker, M., and Turner, H. (1999) Expanded sequence dependence of thermodynamic parameters provides robust prediction of RNA secondary structure. *J. Mol. Biol.* **288**, 911–940.
- 9 Zuker, M. and Stiegler, P. (1981) Optimal computer folding of larger RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* **9**, 133–148.
- 10 Doshi, K., Cannone, J., Cobaugh, C., and Gutell, R. (2004) Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics* **5**, 105.
- 11 Gardner, P. P., Wilm, A., and Washietl, S. (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res.* **33**, 2433–2439.
- 12 Freyhult, E., Moulton, V., and Gardner, P. (2005) Predicting RNA structure using mutual information. *Appl. Bioinformatics* **4**, 53–59.
- 13 Hofacker, I. L., Priwitzer, B., and Stadler, P. F. (2004) Prediction of locally stable RNA secondary structures for genome-wide surveys. *Bioinformatics* **20**, 186–190.
- 14 Washietl, S. and Hofacker, I. L. (2004) Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *J. Mol. Biol.* **342**, 19–39.
- 15 Witwer, C., Hofacker, I. L., and Stadler, P. F. (2004) Prediction of consensus RNA secondary structures including pseudoknots. *IEEE/ACM Trans. Comp. Biol. Bioinf.* **1**, 65–77.
- 16 Hull Havgaard, J., Lyngsø, R., Stormo, G., and Gorodkin, J. (2005) Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics* **21**, 1815–1824.
- 17 Mathews, D. and Turner, D. (2002) Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.* **317**, 191–203.
- 18 Holmes, I. (2005) Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics* **6**, 73.
- 19 Hofacker, I. L., Bernhart, S. H. F., and Stadler, P. F. (2004) Alignment of RNA base pairing probability matrices. *Bioinformatics* **20**, 2222–2227.

