

Detecting regulatory sites using PhyloGibbs

Rahul Siddharthan¹

Erik van Nimwegen²

¹ *The Institute of Mathematical Sciences, Chennai 600 113, India*

² *Biozentrum, University of Basel, Basel, and
Swiss Institute of Bioinformatics, Switzerland*

August 18, 2006

Abstract

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.

Keywords: 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 (messenger RNA or 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 recognising 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 transcription factors is a complex process that can only partially be understood by analysing the DNA sequence itself. However, understanding the process is a hugely important problem. 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 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 maximisation 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 organisation 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 organised. Next we discuss usage of PhyloGibbs on isolated stretches of DNA, on sets of promoters of co-regulated genes, and lastly 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 version 1.0 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 Organisation 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 a few tens to a few

hundred base pairs 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 transcription factor 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 base pairs 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 base pairs 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 transcription factors whose binding specificity is known: for some recent examples, see [4, 5, 6, 7, 8, 9]. Finally, even within a given module, the gene may be under complex combinatorial control of several transcription factors. Some of these may act as co-factors 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 transcription factors (or regulatory sites of another nature), while 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 transcription factor. 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 co-clustered 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 version 1.0, 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 below. 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 below). 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 non-random 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 discussed above, 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.

- **-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 below).
- **-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.

- `-o output_file` : The name of the file into which the simulated-anneal results are written.
- `-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.
- `-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.
- `-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 normalised). For example, `-N 0.3,0.2,0.2,0.3` would be roughly suitable for yeast.

- `-I numlist` or (in newer versions) `-y sites -z factors`: These options are used to specify the 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 twelve 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 example above.

- `-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 co-regulated

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 transcription factors. 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, eg, [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 co-regulated 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 co-regulated 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 don’t 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 3 binding sites per factor in a promoter one may expect 9 binding sites in 3 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 3 of the 6 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 colours, 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 which 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 pre-processing 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” (eg,[11, 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 pre-constructed 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 below 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 which 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 pre-constructed 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 behaviour of PhyloGibbs is highly robust against changes in the proximities that are specified. Therefore, one only needs to get the tree very roughly correct in order 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 tree¹

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

- 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

¹These statements assume colour-changing moves are not used. When colour-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.

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 $q = e^{-s}$.

- Alternatively, one can estimate mutation rates in aligned regions of non-coding 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:

- 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.)
- 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 labelled 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

Pre-aligned input must be provided to PhyloGibbs in multi-fasta format (described below). Any alignment program that provides output in multi-fasta format may be used. Since non-coding 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 non-coding 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 (figure 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 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:

- If a phylogenetic tree in Newick format is specified (see below) 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.)
- 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 don’t discuss these here: they are described in the program manual.
- 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.

- **-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. Since 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.
- **-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 labelling the species concerned. (As noted above, 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 figure 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).

- **-G, -H** : These commands may be used as shortcuts for specifying the phylogeny in simple cases, but since incautious use can lead to errors, we don’t discuss them here; they are documented in the manual.

In addition, the following options have slightly different meanings when running on aligned sequence:

- **-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.

- **-c: Allowing an arbitrary number of windows and colours**

The option `-c` is used to turn on so called “colour moves”, either by `-c n` which specifies `n` colour moves per cycle, or by `-c -1` where PhyloGibbs automatically chooses the number of colour moves per cycle. When colour moves are turned on, PhyloGibbs is no longer restricted to a certain number of sites or maximal number of colours, but can freely vary them. When colour moves are used the values specified in `-I` or `-y` and `-z` are interpreted as the *a priori* expected number of sites and colours. So when using colour moves the user should specify a “best guess” at the total number of sites and motifs represented in the input.

- **-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.

- **-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.

- **-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.

- **-M: Pre-specifying 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 more well defined WMs. For details, see the manual.

8 Output and Track file format

The output file (figure 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 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 0 bits for a completely random distribution of bases to 2 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 (figure 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 below). Finally, after each list of tracked sites the WM 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:

- `-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.
- `-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 and/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, etcetera. 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 his/her input sequences and the prior information that he/she has 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 repre-

sentations 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 weight matrix 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 Licence. By registering you 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 *Saccharomyces 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 which factors are predicted to regulate each gene and which sets of genes are predicted to be regulated by each factor.

References

- [1] R Siddharthan, E D Siggia, and E van Nimwegen. Phylogibbs: A gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput Biol*, 1(7):e67, 2005.
- [2] C E Lawrence, S F Altschul, M S Boguski, J S Liu, A F Neuwald, and J C Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, Oct 1993.
- [3] T L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, 2:28–36, 1994.
- [4] Nikolaus Rajewsky, Massimo Vergassola, Ulrike Gaul, and Eric D Siggia. Computational detection of genomic cis-regulatory modules applied to body patterning in the early *Drosophila* embryo. *BMC Bioinformatics*, 3:30, Oct 2002.
- [5] Saurabh Sinha, Erik van Nimwegen, and Eric D Siggia. A probabilistic method to detect regulatory modules. *Bioinformatics*, 19 Suppl 1:292–301, 2003. Evaluation Studies.
- [6] Saurabh Sinha, Mark D Schroeder, Ulrich Unnerstall, Ulrike Gaul, and Eric D Siggia. Cross-species comparison significantly improves genome-wide prediction of cis-regulatory modules in *Drosophila*. *BMC Bioinformatics*, 5:129, Sep 2004.
- [7] Benjamin P Berman, Barret D Pfeiffer, Todd R Laverty, Steven L Salzberg, Gerald M Rubin, Michael B Eisen, and Susan E Celniker. Computational identifica-

- tion of developmental enhancers: conservation and function of transcription factor binding-site clusters in *Drosophila melanogaster* and *Drosophila pseudoobscura*. *Genome Biol*, 5(9):R61, 2004.
- [8] Benjamin P Berman, Yutaka Nibu, Barret D Pfeiffer, Pavel Tomancak, Susan E Celniker, Michael Levine, Gerald M Rubin, and Michael B Eisen. Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the *Drosophila* genome. *Proc Natl Acad Sci U S A*, 99(2):757–762, Jan 2002.
- [9] O Johansson, W Alkema, W W Wasserman, and J Lagergren. Identification of functional clusters of transcription factor binding motifs in genome sequences: the MSCAN algorithm. *Bioinformatics*, 19 Suppl 1:169–176, 2003. Evaluation Studies.
- [10] J Quackenbush. Computational analysis of microarray data. *Nat Rev Genet*, 2(6):418–427, Jun 2001.
- [11] Mathieu Blanchette and Martin Tompa. FootPrinter: A program designed for phylogenetic footprinting. *Nucleic Acids Res*, 31(13):3840–3842, Jul 2003.
- [12] Mathieu Blanchette and Martin Tompa. Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res*, 12(5):739–748, May 2002. Letter.
- [13] E T Dermitzakis, C M Bergman, and A G Clark. Tracing the evolutionary history of *drosophila* regulatory regions with models that identify transcription factor binding sites. *Mol Biol Evol*, 20(5):703–714, 2003.
- [14] E Emberly, N Rajewsky, and E D Siggia. Conservation of regulatory elements between two species of *drosophila*. *BMC Bioinformatics*, 4(1):57, 2003.

- [15] B Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–218, Mar 1999.
- [16] R Siddharthan. Sigma: multiple alignment of weakly-conserved non-coding dna sequences. *BMC Bioinformatics*, 7:143, 2006.
- [17] J D Thompson, D G Higgins, and T J Gibson. 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(22):4673–4680, Nov 1994.
- [18] Michael Brudno, Chuong B Do, Gregory M Cooper, Michael F Kim, Eugene Davydov, Eric D Green, Arend Sidow, and Serafim Batzoglou. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res*, 13(4):721–731, Apr 2003. Evaluation Studies.
- [19] V Matys, E Fricke, R Geffers, E Gossling, M Haubrock, R Hehl, K Hornischer, D Karas, A E Kel, O V Kel-Margoulis, D-U Kloos, S Land, B Lewicki-Potapov, H Michael, R Munch, I Reuter, S Rotert, H Saxel, M Scheer, S Thiele, and E Wingender. TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res*, 31(1):374–378, Jan 2003.

Figures

Figure 1: An example of aligned sequence in multi-fasta format which may be fed to PhyloGibbs: the promoter of NDT80, a gene from *S. cerevisiae*, and orthologous regions from *S. bayanus* and *S. mikatae*.

```
>Scer_YHR124W NDT80 SGDID:S0001166 5' untranslated region, Chr VIII:3555
63..356562, 1000 bp length=999
atcgcaactttgtatctacttttttttatttcgaaaacaaggcacaacaatgaa-----TCTAT
CGCCCTGTGAGATTTTCAATCTCAAGTTTGTGTAATAGATAGCGTTATATTATAGAactataaaggtccttg
aatatacatagtggtttcattcctattactgTATATGTGACTTTACATTGTTACTTCCGCGGCTATTTGACGT
TTTctgctTCAGGTGCGGCTTGGAGGGCAAAGTGTGAGAAAATCGGCCAGGCGTATGACACAAAAGAGTAG
AAAACGAGATCTCAAATATCTCGAGGCCGTCTCTATAC-AACCGCCAGCTCTCTGACAAAAGTCCAGAA
CGGTTGTCTTTTGTTCGAAAAGCCAAGGTCCTTATAATTGCCCTCCATTTTGTGTGTCACctattTAAGCAA
AAAATTGAAAGTTTACTAACCTTTTCATTAAGAGAAATAACAATATTATAAAAA-GCGCTTAAA
>Sbay_Contig514.9 NDT80 YHR124W 5' untranslated region, Contig c514 1530
5 - 16304(revcom), 1000 bp length=999
aaccgcactttggttcacacgttttctgtttgtttgtcttccctttatTTAAATAAAAACCCAATTTTCTCTAT
TGCCCTGCGGGACAACCGGTCTCTAGTCTGTGTAATAGATAACATTATATTATAGAATGATAGAACTATCG
ATATGCATAGTGCTTTTATCGCTGTGCGAGATATATCTGGCCTCACCTTATCACTTCCGCGGCTATTTGACGT
TTTTTGT-TCAAGCGCGGCTTGGACGGCAAAGTGTGAGAAAATCGCCAGGCTGTATGACACAAAAGGGcaa
aaagagatctcaaaagccctctcgagacaagtctcttgctgAACCGCCGAGCTCTCTGCAAACCTCTATTGGA
CAATCATCTTTTGTGTTGAAGAGGTAACCTCCGTTACAGTTGTCCCCATTTTGTGTGTCatcTAC-TAAAGTA
GAAATTAAGTTTAATAAACATTCAATAAAGAGGGAAAACGGTAATATAAAAAaGCACCTTAAA
>Smik_Contig2829.6 NDT80 YHR124W 5' untranslated region, Contig c2829 69
67 - 7966, 1000 bp length=999
aaatcatgtttggtggtttacgcttctctcttttttttctta-----TTAAACAAGGTACAAAGCACTCTAT
TGCTCCGTGAGATTATCAATCTCAACTTTGTGTAATAGATACC GTTATATTATAGAGTTATAGAAATCCGTTT
GATGTACATAGTGCTTCATTGCTGTTGCAAGTATATGTAGTTTACATTGTAACCTTCCGCGGCTATTTGACGT
TTTTTTG-TCCAGTGCGGCTTAAAGGCCAAAGTGTGAGAAAATCGGCCATGCCGAATGACACAAAAGAGTGG
CAACCGATATCTCAAGGTTCTCGAGGTCTATTCTATTCTG-AACCGCCAGCTTTCTAAAAAAGGTCACCAA
CAGTTGTCTTTTGTGTTGACGAGCCAAGGTCGTTATAACTGTCCGCGGTTTTGTGTGTCAC-TAT-TAAAACA
AAAAATAAAGCTTAGTATACTTTTCATTAAGAGGgacaacagtaatattaaa--GCGCTTAAAa
```

Figure 2: A sample phylogenetic tree for mammals (numbers are only approximate). Since 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).

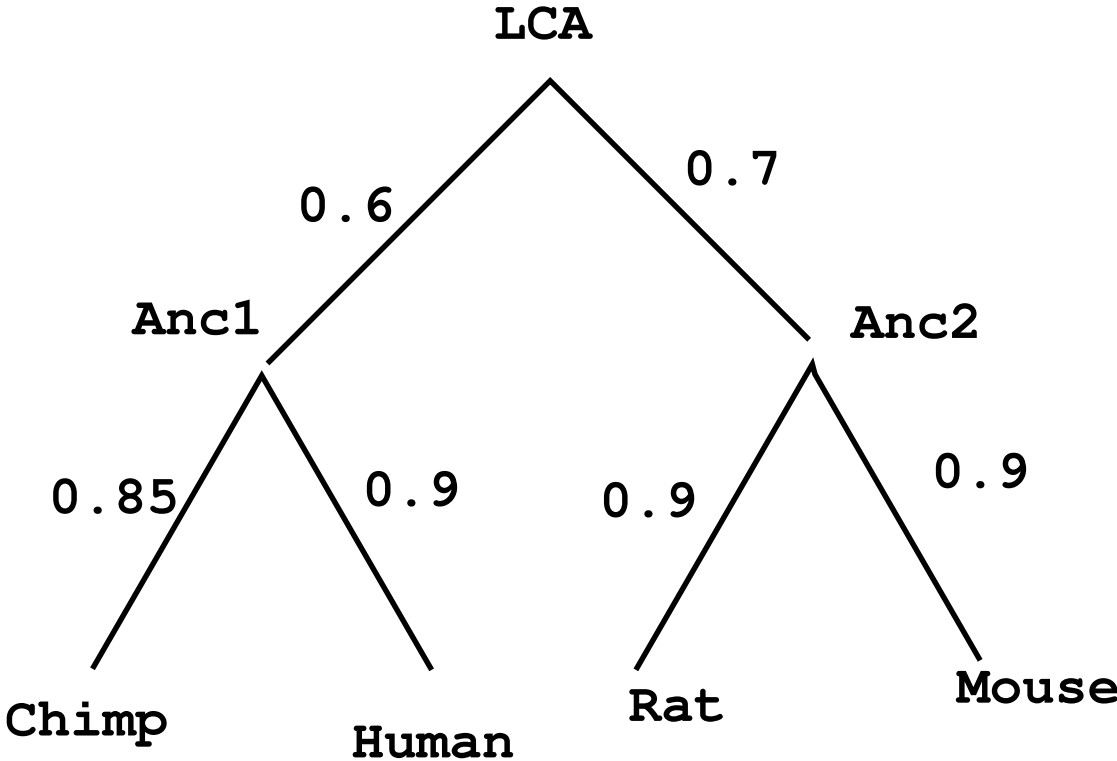


Figure 3: Sample output file. Output sequences that occur in conserved blocks are grouped together.

```

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 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
tcaatttTACATCAATCAAGCTaaagcag | - [fwd] seq 17 Sbay_Contig pos 924
tcaatttTACATCAATTAAGTCaaagcag | - [fwd] seq 18 Smik_Contig pos 920
tcaatttCACATCAACTAAATCaagctag | - [fwd] seq 19 Skud_Contig pos 914
gtataggCTTGTATTTCAGAAATgtgatcc -- [rev] seq 15 Scer_YJR011 pos 277 score 5.725e-07
gtataggCTTATTATTCAGAAATgtgatcc | - [rev] seq 16 Spar_12348 pos 300
gtataggTTTATTATTTAAAAATgtggtcc | - [rev] seq 17 Sbay_Contig pos 351
gtataggCTTATTGTTTAAAAATatgatct | - [rev] seq 18 Smik_Contig pos 311
gtacgggCTTGTATTATTAATgtggtcc | - [rev] seq 19 Skud_Contig pos 364
tatatacCTTATTCATCAACACtttctcc -- [rev] seq 25 Spar_14750 pos 262 score 5.652e-06
tatatacTTATTATTCATCAACACtttctct | - [rev] seq 26 Sbay_Contig pos 261
tatatacCTTATTCATCAACACtttctcc | - [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 ...

```

Figure 4: Sample track file.

```

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 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 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
-----
ctgttttAAAATCCTTATCTTGctctctt -- [fwd] seq 0 Scer_YDL237 pos 49 prob 1.00
ctgttttGGGTTCCCTTATCTTGctctctt |-- [fwd] seq 1 Spar_2881 pos 48
acattttAGATTTCCTTATCTTTctccctt |-- [fwd] seq 2 Sbay_Contig pos 69
ttgcttcACGTGCTTATCTCGctctctt '- [fwd] seq 4 Skud_Contig pos 37
tgggtgaATAATTTACCTAGCTgttggat -- [rev] seq 15 Scer_YJR011 pos 621 prob 0.71
gtcgcaagTAATTTACCTAGTTttcggtt |-- [rev] seq 16 Spar_12348 pos 644
gtcgtaacGAGACTTATCTAGTCatcgatt |-- [rev] seq 17 Sbay_Contig pos 671
ttcttgaATGATTTACTGACTatccctt '- [rev] seq 18 Smik_Contig pos 643
catgcgtAAGGTTTATCTAGTTattgatt |-- [rev] seq 19 Skud_Contig pos 672
tacttatCTTAACCTTATCGTTtctctog -- [fwd] seq 5 Scer_YEL062 pos 333 prob 0.68
tacttatCTTAACCTTATCGTTtctctog |-- [fwd] seq 6 Spar_5973 pos 344
tacttatCTCGACCTTATCGCTtctctog |-- [fwd] seq 7 Sbay_Contig pos 289
aacctatCATAACTTTATCGTAttctctog |-- [fwd] seq 8 Smik_Contig pos 200
tagctatCGCAACCTTATCGTTtctctog '- [fwd] seq 9 Skud_Contig pos 372
ttttatgCTGCACCTTATCTAAGtaaata -- [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 ...

```