Supervised detection of conserved motifs in DNA sequences with `cosmo`

Oliver Bembom, Fabian Gallusser, Sandrine Dudoit

Division of Biostatistics, University of California, Berkeley

April 13, 2011

Contents

1 Introduction

1.1 Overview

The Bioconductor R package `cosmo` implements an algorithm for searching a set of unaligned DNA sequences for a shared motif that may, for example, represent a common transcription factor binding site (Bembom et al., 2007). `cosmo` is extends the popular motif discovery tool `MEME` (Bailey and Elkan, 1995) in that it allows the search to be supervised by specifying a set of constraints that the motif to be discovered must satisfy. Such constraints may, for example, consist of bounds on the information content across certain regions of the unknown motif and can often be formulated on the basis of prior knowledge about the structure of the transcription factor in question. The user is not required to specify *a priori* the width of the unknown motif, the distribution of motif occurrences among the input sequences (OOPS, ZOOPS, or TCM), or a single correct constraint set. Instead these three model parameters can be selected in a data-adaptive manner.

1.2 Motivation

An important goal in contemporary biology consists of deciphering the complex network that regulates the expression of an organism's genome. A central role in this network is played by transcription factors that regulate gene expression by binding to conserved short sequences in the vicinity of their target genes (Davidson, 2001). The discovery and description of these *binding sites* or *motifs* has therefore been at the heart of efforts aimed at understanding gene regulatory networks.

Traditionally, experimental methods have been used for this purpose, leading to a set of target sites from multiple genes that could then be aligned to estimate the position weight matrix (PWM) of the motif - a $4 \times W$ matrix in which position (j, w) gives the probability of observing nucleotide j in position w of a motif of length W. Currently, however, such position weight matrix estimates are more commonly obtained by applying pattern discovery algorithms to functional genomics data. Modern high-throughput methods such as cDNA microarrays (Roth et al., 1998; Eisen et al., 1998; Bussemaker et al., 2001)or SAGE (Powell,

2000), for example, can identify sets of co-regulated genes whose promoter sequences can then be scanned for statistically over-represented patterns that are likely transcription factor binding sites (Lawrence et al., 1993; Bussemaker et al., 2001).

While this approach has proven fruitful for the discovery of such binding sites in yeast, its application to metazoan genomes has met with considerable difficulty since binding sites tend to be spread out over much larger regions of genomic sequence. Efforts at tackling this signal-to-noise problem have concentrated mostly on phylogenetic footprinting, i.e. cross-species sequence comparisons that remove noise by focusing on sequences under selective pressure (Fickett and Wasserman, 2000). Sandelin and Wassermann (2004), however, recently described an alternative approach that is based on prior knowledge about the structural class of the mediating transcription factor of interest. Such knowledge is often available on the basis of genetics or similarities between biological systems. For most structurally related families of transcription factors, there are clear similarities in the sequences of the sites to which they bind (Luscombe et al., 2000). Eisen (2005), for example, has demonstrated that motifs bound by proteins with structurally similar DNA binding domains tend to have similar information content profiles (Schneider et al., 1986). Prior knowledge about the structural class of the mediating transcription factor thus often translates into constraints on the unknown position weight matrix that can be used to enhance the performance of pattern discovery algorithms. Sandelin and Wassermann (2004) show that the benefit of such prior knowledge is comparable to the specificity improvements obtained through phylogenetic footprinting.

Currently, only a few motif finding algorithms such as ANN-Spec (Workman and Stormo, 2000) or the Gibbs motif sampler (Neuwald et al., 1995; Thompson et al., 2003) are capable of incorporating prior knowledge about the unknown motif. These algorithms generally require the user to supply an appropriate prior distribution on the entries of the corresponding position weight matrix. `cosmo` instead allows the user to target the motif search by specifying a set of constraints that the unknown position weight matrix must satisfy. The algorithm is based on a probabilistic model that describes the DNA sequences of interest through a two-component multinomial mixture model as first introduced by Lawrence and Reilly (1990), with estimates of the position weight matrix entries obtained by maximizing the observed data likelihood over the smaller parameter space corresponding to the imposed constraints.

2 Methods

2.1 Probabilistic models

2.1.1 Motifs and background

All of the models described below assume that sequences are generated according to a multinomial mixture model with two components, one that describes the distribution of nucleotides in the motif, and one that describes the distribution of nucleotides in the background. Nucleotides that are part of the length-W transcription factor binding site are assumed to be generated independently of each other according to multinomial distributions that are allowed to be different for each nucleotide in the motif. Nucleotides that are not part of a

motif are assumed to be generated according to a k-th order Markov model that allows the parameter vector of the multinomial distribution of the current nucleotide to depend on the previous k nucleotides.

2.1.2 OOPS

The one-occurrence-per-sequence (OOPS) model assumes that every sequence contains exactly one occurrence of the motif. For a given sequence of length L_i, any of the $L_i - W + 1$ eligible motif starts are equally likely to be the start site of the motif. At a given start site, the motif is equally likely to be present in either one of the two possible orientations. For example, a motif with consensus sequence ATGCCC may be present as ATGCCC or in its reverse complement orientation as GGGCAT.

2.1.3 ZOOPS

The zero-or-one-occurrence-per-sequence (ZOOPS) model assumes that a given sequence contains one occurrence of the motif with probability π and no occurrences of the motif with probability $1 - \pi$. For a given sequence that contains a motif, any of the $L_i - W + 1$ eligible motif starts are equally to be the start site of the motif. At a given start site, the motif is equally likely to be present in either one of the two possible orientations.

2.1.4 TCM

The OOPS and ZOOPS models allow at most one occurrence of the motif per sequence. However, there are many biological examples of DNA sequences that contain multiple occurrences of the same transcription factor binding site. Bailey and Elkan (1995) propose a two-component mixture (TCM) model for this situation that allows each sequence to contain an arbitrary number of non-overlapping occurrences of the motif. This model assumes that a given sequence is generated by repeatedly deciding whether to insert a background nucleotide or a motif of width W. As before, a motif is inserted in either one of the two possible orientations with equal probability. We denote by λ the probability that a motif is inserted at a given position rather than a background nucleotide.

2.2 Constraints

2.2.1 Motif intervals

Many motifs can be conceptually divided into separate intervals that each correspond to a distinct set of constraints on the position weight matrix. In order to specify constraints for `cosmo`, we hence first specify how the motif can be divided into separate intervals. Since the true motif width is usually unknown, forcing `cosmo` to search a range of candidate values, we have to specify how the width of each interval changes with varying motif widths. We offer three possibilities: The length of an interval may be a fixed number of based pairs no matter what the length of the whole motif is; alternatively, the length of an interval may

always be a fixed proportion of the length of the whole motif; finally, a motif may contain one interval that for each motif width is assigned whatever number of base pairs is left after all intervals of the first two kinds have been allocated. Once the motif has been divided into separate intervals, we can add a number of different constraints to individual intervals or to the motif as a whole.

2.2.2 Bound constraints on the information content across an interval

An important summary measure of a given position weight matrix is its information content profile. The information content at position w of the motif is given by

$$IC(w) = \log_2(J) + \sum_{j=1}^{J} p_{wj} \log_2(p_{wj}) = \log_2(J) - entropy(w)$$

where J denotes the number of letters in the alphabet from which the sequences have been derived so that here $J = 4$. The information content is measured in bits and, in the case of DNA sequences, ranges from 0 to 2 bits. A position in the motif at which all nucleotides occur with equal probability has an information content of 0 bits, while a position at which only a single nucleotide can occur has an information content of 2 bits. The information content at a given position can therefore be thought of as giving a measure of the tolerance for substitutions in that position: Positions that are highly conserved and thus have a low tolerance for substitutions correspond to high information content, while positions with a high tolerance for substitutions correspond to low information content.

It has been shown that the information content at a given position of a motif is proportional to the number of contacts between the protein and the base pair at that position. We therefore expect higher information content in regions of the motif that are bound by the transcription factor than in the remaining regions. If the transcription factor contains two DNA-binding domains whose target sequences in the motif are separated by a short stretch of sequence that does not interact with the protein, we would expect that the information content of the motif follows a high-low-high pattern. In this case, it may be useful to give bounds on the information content across an individual interval.

2.2.3 Shape constraints on the information content profile across an interval

We may want to exclude position weight matrices from consideration whose information content profile is sharply discontinuous across a given interval. This can be achieved by requiring the information content profile across that interval to follow a linear or monotone shape. In both cases, we may also give bounds on the information content at the left and right edge of the interval.

2.2.4 Lower bounds on nucleotide frequencies across an interval

We may suspect that a given nucleotide occurs with high frequency across a certain interval. In that case, we may require that the average frequency of a given nucleotide j across all

positions in the interval is no less than some lower bound. Similarly, we may require that the GC-content or AT-content across an interval is no less than some lower bound. If the length of the interval does not change with varying motif width, we may also impose lower bounds for nucleotide frequencies at a single position in that interval.

2.2.5 Palindromic intervals

If the DNA-binding domains of the transcription factor are homodimeric, the DNA stretches that are bound by the transcription factor will be palindromes of each other. `cosmo` allows the user to specify two intervals that are thought to be palindromic with respect to each other. In particular, we require that the frequency of nucleotide j at position l in the interval equal the frequency of the palindrome of nucleotide j at position l from the right edge of interval, to within a given error bound.

2.2.6 Submotifs

Families of transcription factors are often characterized by the occurrence of a certain submotif within the motif. The exact location of the submotif within the motif, however, can vary widely. DNA sequences bound by transcription factors with an ETS domain, for example, all contain the stretch GGAA somewhere within the binding site. `cosmo` allows the user to specify such a submotif that is then required to occur somewhere within the motif of interest, with nucleotide frequencies of the consensus nucleotides in the submotif roughly equal to some user-specified frequency.

2.2.7 Bounds on differences of shape parameters

Sometimes we may wish to impose constraints on the shape of the information content that cannot be specified by the shape constraints described above. For example, we may wish to require that the information content across a certain interval is constant, or that the information content profile be continuous at the junction between two intervals.

Such constraints can be formulated by giving bounds on the difference between two shape parameters. Recall that shape constraints on the information content profile across an interval are parameterized using the information content at the left and right edge of the interval. Hence we may require that these two quantities be identical, corresponding to a constant information content profile across that interval. As another example, we might require that the information content at the end of one interval is identical to the information content at the beginning of the next interval, corresponding to the constraint that the information content profile be continuous at the junction between the two intervals.

2.3 Model selection

The probabilistic models described above are indexed by by following four parameters:

1. The order of the background Markov model.

2. The width of the motif.

3. The type of model used to describe the data generating process (OOPS, ZOOPS, or TCM).

4. The set of constraints on the position weight matrix of the motif,

For each one of these four parameters, `cosmo` allows the user to either make a manual selection or to have the appropriate index selected data-adaptively. For data-adaptive selection, the user may choose from among a number of different model selection approaches.

2.3.1 Likelihood-based validity functionals

Apart from the likelihood, we also consider Akaike's Information Criterion AIC (Akaike, 1973) and the Bayesian Information Criterion BIC (Schwarz, 1978). BIC has been found to work fairly well for selecting the unknown motif width, while the likelihood and AIC generally perform quite poorly in the model selection tasks we consider.

2.3.2 E-value of the resulting multiple alignment

The E-value of the multiple alignment consisting of the predicted motif occurrences is an approximate p-value for testing the null hypothesis that this alignment was obtained from a set of sequences that were generated entirely from the background distribution against the alternative hypothesis that the sequences were generated according to the estimated model.

This measure of statistical significance has been found to work well for all three model selection problems and forms the default approach used by `cosmo` for selecting the motif width as well as the distribution of motif occurrences.

2.3.3 Likelihood-based cross-validation

Likelihood-based cross-validation is a popular approach to model selection in the context of density estimation (van der Laan et al., 2003). The general idea of cross-validation is to divide the original dataset into a training set that is used to estimate the parameters of a given model and a validation set that is then used to evaluate the performance of this estimated model. This performance assessment is based on an appropriately specified loss function. In the context of likelihood-based cross-validation this loss function is taken to be the Kullback-Leibler divergence (DKL), which gives a measure of the distance between two densities f and g.

Likelihood-based cross-validation generally performs rather poorly at the model selection problems encountered here, presumably because it is aimed at estimating the entire density of the data-generating distribution well instead of the lower dimensional functionals of this density we are concerned with here.

2.3.4 Cross-validation based on the Euclidean norm

For selection problems in which W is fixed, notably in selecting between candidate constraint sets, we may want to use as loss function the Euclidean norm between a position weight matrix estimate obtained under a candidate constraint set and an independent position weight matrix estimate obtained under no constraints.

Cross-validation based on this loss function has been found to perform well for the purpose of selecting between different candidate constraint sets and forms the default approach employed by `cosmo` for this problem.

2.3.5 Separate model selection criteria for different parameters

The user is allowed to specify different model selection criteria for selecting the different parameters. In fact, the default settings cause `cosmo` to select the motif width and the model type based on the E-value criterion, but the constraint set by cross-validation based on the Euclidean-norm loss function.

`cosmo` handles such situations using the following profiling approach. For each given combination of constraint set and model, it first finds the optimal motif width based on the criterion selected for choosing the motif width. In the next step, it selects the optimal model type for each given constraint set at the chosen value for the motif width. Finally, it selects the optimal constraint set for the chosen values of the motif width and model type. This approach is computationally attractive since `cosmo` is not required to evaluate the different model selection criteria for all possible candidate models.

3 Software implementation

3.1 Overview

The supervised motif detection algorithm described above is implemented in the Bioconductor R package `cosmo`. This package offers functions for generating random sequences according to the three different probabilistic models, functions for generating R objects representing sets of constraints on the unknown position weight matrix, as well as a function for carrying out the algorithm itself.

Before being able to access these functions, the user is required to load the package using the `library()` command:

```
> library(cosmo)

Welcome to cosmo version 1.18.0

cosmo is free for research purposes only. For more details, type
license.cosmo(). Type citation('cosmo') for details on how to cite
cosmo in publications.
```

3.2 Simulating sequences

The function `rseq()` allows the user to generate random sequences according to the OOPS, ZOOPS, or TCM models:

```
> args(rseq)

function (numSeqs, seqLength, rate, pwm, transMats, model = "ZOOPS",
    posOnly = FALSE)
NULL
```

INPUT.

1. The number of sequences to be generated, `numSeqs`.
2. The number of nucleotides in each sequence, `seqLength`. This may be either a single number, in which case that number is taken to be the common length of all sequence, or a vector of sequence lengths.
3. The intensity parameter for the ZOOPS and TCM models, `rate`. For the ZOOPS model, this corresponds to π; for the TCM model, this corresponds to λ.
4. The position weight matrix of the motif, `pwm`.
5. The transition matrix for the background Markov model, `transMats`. This is a list of matrices, with the first matrix given the transition probabilities for the 0th order Markov model, the second matrix giving the transition probabilities for a 1st order Markov model, and so on.
6. The distribution of motif occurrences, `model`. This is either "ZOOPS" or "TCM"; the OOPS model is a special case of the "ZOOPS" model.
7. A choice for whether motifs may only be inserted in the forward strand orientation instead of allowing the reverse complement orientation as well, `posOnly`.

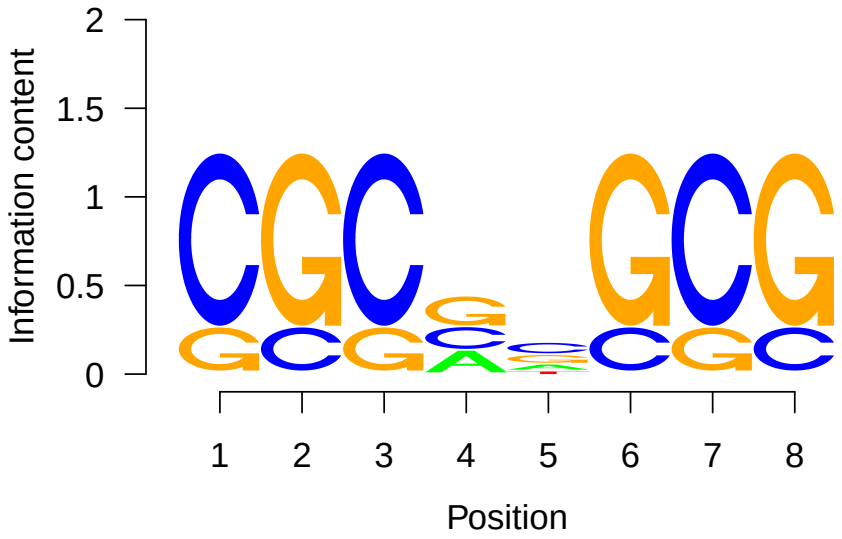
OUTPUT.

1. A list of the generated sequences, `seqs`.
2. An `align` object `motifs` summarizing the positions of the inserted motif occurrences.
3. An object `empPWM` of class `pwm` representing the position weight matrix obtained by aligning the inserted motifs.

EXAMPLE.

The `cosmo` package contains the following example of a position weight matrix for a motif of width 8:

Figure 1: Sequence logo of motif used for simulating sequences.




```
> data(motifPWM)
> motifPWM

    1   2   3   4   5   6   7   8
A 0.0 0.0 0.0 0.3 0.2 0.0 0.0 0.0
C 0.8 0.2 0.8 0.3 0.4 0.2 0.8 0.2
G 0.2 0.8 0.2 0.4 0.3 0.8 0.2 0.8
T 0.0 0.0 0.0 0.0 0.1 0.0 0.0 0.0
```

The `seqLogo()` function found in the `seqLogo` package can be used to produce the sequence logo shown in figure 1 (Schneider and Stephens, 1990). The `cosmo` package also contains the following example of transition matrices needed for a second-order Markov model for the distribution of background nucleotides:

```
> data(transMats)
> transMats

$order0
           A         C         G         T
-- 0.3226181 0.1783398 0.1776999 0.3213423

$order1
          A         C         G         T
```

```
A 0.3750312 0.1604460 0.1771620 0.2873608
C 0.3253088 0.1891273 0.1671617 0.3184022
G 0.3140129 0.2090844 0.1865159 0.2903868
T 0.2732550 0.1739084 0.1795711 0.3732655

$order2
           A         C         G         T
AA 0.4224705 0.1437651 0.1841981 0.2495663
AC 0.3480425 0.1806008 0.1724194 0.2989373
AG 0.3419518 0.1958921 0.1892899 0.2728662
AT 0.3400486 0.1576593 0.1794790 0.3228131
CA 0.3492801 0.1812818 0.1829259 0.2865122
CC 0.3095298 0.1987338 0.1695311 0.3222054
CG 0.2844824 0.2251332 0.1906993 0.2996852
CT 0.2440334 0.1832112 0.1850531 0.3877023
GA 0.4109319 0.1418031 0.1867843 0.2604807
GC 0.3298844 0.1913179 0.1809832 0.2978145
GG 0.3169036 0.2158552 0.1905797 0.2766614
GT 0.3079272 0.1532995 0.2011074 0.3376659
TA 0.3046557 0.1818625 0.1577414 0.3557404
TC 0.3109209 0.1894812 0.1518175 0.3477804
TG 0.2995633 0.2100971 0.1795034 0.3108363
TT 0.2210835 0.1909250 0.1678571 0.4201344
```

We may now generate 20 sequence each of length 100 nucleotides according to the OOPS model and this position weight matrix and background distribution as follows:

```
> simSeqs <- rseq(20, 100, 1, motifPWM, transMats, "ZOOPS")
> simSeqs$motifs

     seq pos orient    motif prob
1   Seq1  65      1 CGCACCCC    1
2   Seq2  75     -1 GGCGCGCC    1
3   Seq3  18     -1 CCGCCGGG    1
4   Seq4  70     -1 CGCGCGCC    1
5   Seq5   8      1 CGCGCGCG    1
6   Seq6  29      1 CCCGGGGG    1
7   Seq7  17     -1 CGCACGGG    1
8   Seq8  55     -1 GGCACGCG    1
9   Seq9   5      1 CGCCGGGG    1
10 Seq10  90      1 CGGCGGCG    1
11 Seq11  17      1 CGCGCGCC    1
12 Seq12  88      1 CGGGAGCG    1
13 Seq13  53      1 GGCCCGCC    1
```

```
14 Seq14  76     -1 CGCCGGCC    1
15 Seq15  54      1 CGCAAGCG    1
16 Seq16  22     -1 CCCCCGCG    1
17 Seq17  37     -1 CCCAGGGG    1
18 Seq18  16      1 CGCGTGCG    1
19 Seq19  54      1 CCGGAGCC    1
20 Seq20  23      1 CCCATGCG    1
```

3.3 Constructing constraint sets

The `cosmo` package defines classes `constraintSet` and `constraintGroup` that represent a single constraint sets and a collection of constraint Sets, respectively. A `constraintSet` object is initially constructed using the function `makeConSet()`

```
> args(makeConSet)

function (numInt, type, length, descrip = "Constraint Set")
NULL
```

that takes as arguments the number of intervals that the motif is to be divided into, the types of those intervals and the lengths of those intervals. A constraint set consisting of a 3-bp interval, a variable-length interval, and another 3-bp interval is constructed as

```
> conSet1 <- makeConSet(numInt = 3, type = c("B", "V", "B"), length = c(3,
+     NA, 3))
```

`constraintSet` objects are displayed in the format employed by the `cosmoweb` web application (`http://cosmoweb.berkeley.edu`):

```
> conSet1

@ ConstraintSet: 1
>IntervalSetup
Length: 3 bp
Length: variable
Length: 3 bp
```

We may now construct a list of constraints that can then be added to this constraint set. To require the information content across the first interval to be bounded between 1.0 and 2.0, we construct the following `boundCon` object:

```
> boundCon1 <- makeBoundCon(lower = 1, upper = 2)
```

Likewise, we construct the following bound constraint for the second interval:

```
> boundCon2 <- makeBoundCon(lower = 0, upper = 1)
```

Lastly, we may construct a palCon object to require that intervals 1 and 3 be palindromes of each other:

```
> palCon1 <- makePalCon(int1 = 1, int2 = 3, errBnd = 0.05)
```

These constraints can now be added to the appropriate intervals of the initially defined constraintSet object:

```
> constraint <- list(boundCon1, boundCon2, palCon1)
> int <- list(1, 2, NA)
> conSet1 <- addCon(conSet = conSet1, constraint = constraint,
+     int = int)
> conSet1

@ ConstraintSet: 1
>IntervalSetup
Length: 3 bp
Length: variable
Length: 3 bp
>IcBounds
Interval: 1
Bounds: 1 to 2
>IcBounds
Interval: 2
Bounds: 0 to 1
>Pal
Intervals: 1 and 3
ErrorTol: 0.05
```

We construct a second constraint set that requires the motif to contain the submotif TATA:

```
> conSet2 <- makeConSet(numInt = 1, type = "V", length = NA)
> subCon1 <- makeSubMotifCon(submotif = "TATA", minfreq = 0.9)
> conSet2 <- addCon(conSet = conSet2, constraint = subCon1, int = NA)
> conSet2

@ ConstraintSet: 1
>IntervalSetup
Length: variable
>SubMotif
Motif: TATA
MinFreq: 0.9
```

3.4 `cosmo` function

The `cosmo()` function carries out the supervised motif detection algorithm described above.

```
> args(cosmo)

function (seqs = "browse", constraints = "None", minW = 6, maxW = 15,
    models = "ZOOPS", revComp = TRUE, minSites = NULL, maxSites = NULL,
    starts = 5, approx = "over", cutFac = 5, wCrit = "bic", wFold = 5,
    wTrunc = 100, modCrit = "lik", modFold = 5, modTrunc = 100,
    conCrit = "likCV", conFold = 5, conTrunc = 90, intCrit = "lik",
    intFold = 5, intTrunc = 100, maxIntensity = FALSE, lstarts = FALSE,
    backSeqs = NULL, backFold = 5, bfile = NULL, transMat = NULL,
    order = NULL, maxOrder = 6, silent = FALSE)
NULL
```

INPUT.

1. A reference to the set of sequence to be analyzed, `seqs`. This may be a list with each element representing a sequence in the form of a single string such as "ACGTAGCTAG" ("seq" entry) and a description ("desc" entry), the path of a file that contains the sequences in FASTA format, or the string "browse", in which case the user is prompted to browse for a FASTA file containing the input sequences.

2. A reference to the constraint sets, `constraints`. This may be a `constraintSet` object, a list of such objects, the name of a file containing the constraint definitions in the format used by `cosmoweb`, the string "None" for no constraints. If the `cosmoGUI` package has been installed, constraint sets may also be defined through an interactive Tcl/Tk-based GUI by specifying `constraints=GUI` (see figure 2).

3. The minimum and maximum motif widths to search through, `minW` and `maxW`.

4. A character vector giving the model types to be considered as candidates for the sequences at hand, `models`. The possible candidates are "OOPS", "ZOOPS", and "TCM".

5. A logical indicator for whether motifs are allowed to occur in the reverse complement orientation, `revComp`.

6. The minimum and maximum number of motif occurrences in the entire set of sequences, `minSites` and `maxSites`.

7. The number of starting values to use for the constrained optimization of the likelihood function, `starts`. In many cases, increasing the number of starting values can help improve the performance of the algorithm, whereas decreaseing the number of starting values will reduce the computing time.

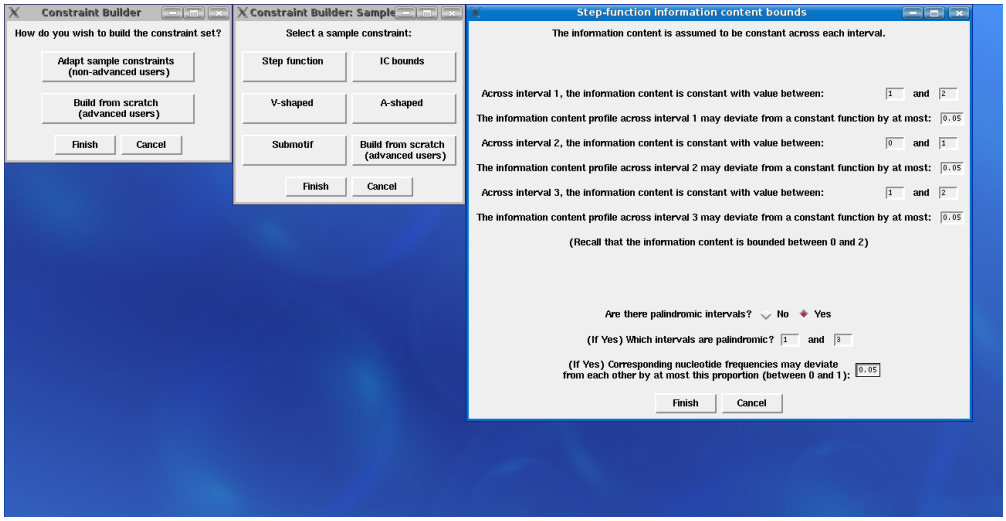


Figure 2: GUI for constructing constraint sets.

8. A number of more advanced parameters, pertaining mostly to the model selection procedure. The default values will be perfectly sufficient for the vast majority of users, with the available options mostly given for testing and simulation purposes.

OUTPUT.

The S4 class/method object-oriented programming approach was adopted to summarize the results of the motif search. Specifically, the output is an instance of the class `cosmo`. A brief description of the class is given next. Please consult the documentation for details, e.g., using `class ? cosmo` and `methods ? cosmo`.

```
> slotNames("cosmo")
```

```
[1] "seqs"      "pwm"       "back"      "tmat"      "cand"
[6] "cons"      "sel"       "motifs"    "probs"     "objectCall"
```

1. A list `seqs` with each element representing one sequence of the input dataset in the form of a single string such as "ACGTAGCTAG" ("seq" entry) and a description ("desc" entry).

2. The estimated position weight matrix, `pwm`. This is an instance of the class `pwm`, containing additionally the information content profile of the position weight matrix and the corresponding consensus sequence. Invoking the `plot()` method on an object of class `pwm` produces a plot of the sequence logo of the position weight matrix (Schneider et al., 1986)

3. A summary of the model selection process for the order of the background Markov model, `back`. This is a data.frame that gives for each candidate order the cross-validated Kullback-Leibler divergence.

4. The estimated transition matrices for the background Markov model, `tmat`.

5. A summary of the model selection process for selecting the motif width, model type, and constraint set, `cand`. This is a data.frame that gives for each candidate model considered the values of the relevant model selection criteria.

6. The selected constraint set, `cons`. This is an instance of the class `constraintSet`.

7. A description of the selected model, `sel`. This is a data.frame that summarized the selections made for the constraint set, the model type, the motif width, and the order of the background Markov model.

8. A summary of the predicted motif occurrences, `motifs`. This is an instance of the class `align` that gives, for each predicted motif occurrence, the sequence name, the position on the sequence, the orientation of the motif, the motif itself, and the posterior probability of this site being a motif occurrence.

9. A list `probs` with each entry giving the posterior probabilities of motif occurrences along a given sequence. If the motif is more likely to occur in the reverse complement orientation, this posterior probability appears with a negative sign.

EXAMPLE.

The `cosmo` package includes the example FASTA file `seq.fasta`. It contains 20 sequences that were simulated as above according to the OOPS model to each contain one occurrence of the motif whose sequence logo is given in figure 1. We can search these sequences for a shared motif, considering as candidate constraint sets the two constraint sets constructed in section 3.3, as candidate model types OOPS and TCM, and as candidate motif widths 7 through 8:

```
> seqFile <- system.file("Exfiles/seq.fasta", package = "cosmo")
> res <- cosmo(seqs = seqFile, constraints = list(conSet1, conSet2),
+     minW = 7, maxW = 8, models = c("OOPS", "TCM"))
```

The `print()` method outputs the estimated position weight matrix:

```
> print(res)
```

```
       1      2      3      4      5      6      7      8
A 0.0000 0.0000 0.0000 0.1591 0.1932 0.0000 0.0000 0.0000
C 0.8165 0.2294 0.7676 0.3531 0.2984 0.1824 0.8206 0.2335
G 0.1835 0.7706 0.2324 0.4878 0.1607 0.8176 0.1794 0.7665
T 0.0000 0.0000 0.0000 0.0000 0.3476 0.0000 0.0000 0.0000
```

A more detailed summary of the results is obtained through the `summary()` method:

```
> summary(res)

Input dataset:
   Sequence Length
1      Seq1    100
2      Seq2    100
3      Seq3    100
4      Seq4    100
5      Seq5    100
6      Seq6    100
7      Seq7    100
8      Seq8    100
9      Seq9    100
10    Seq10    100


Candidate orders for background Markov model:
  order         klDiv
1     0  1.351885e+02
2     1  1.352459e+02
3     2  1.367819e+02
4     3 1.797693e+308
5     4           Inf
6     5           Inf
7     6           Inf


Candidate models considered:
  conSet model width    wCrit   modCrit conCrit
1      1  OOPS     7 2705.698        NA      NA
2      1  OOPS     8 2686.247 -1315.493 133.259
3      1   TCM     7 2720.641        NA      NA
4      1   TCM     8 2706.129 -1324.282      NA
5      2  OOPS     7 2731.628        NA      NA
6      2  OOPS     8 2729.742 -1337.240 142.939
7      2   TCM     7 2736.589 -1342.966      NA
8      2   TCM     8 2738.835        NA      NA


Selected model:
             choice  crit    critVal
Constraint        1 likCV   133.2590
Model          OOPS   lik -1315.4926
```

```
Width             8   bic  2686.2472
NumSites         10   lik -1315.4926
Markov Order      0 likCV   135.1885


Selected constraint set:
@ ConstraintSet: 1
>IntervalSetup
Length: 3 bp
Length: variable
Length: 3 bp
>IcBounds
Interval: 1
Bounds: 1 to 2
>IcBounds
Interval: 2
Bounds: 0 to 1
>Pal
Intervals: 1 and 3
ErrorTol: 0.05


Estimated position weight matrix:
       1      2      3      4      5      6      7      8
A 0.0000 0.0000 0.0000 0.1591 0.1932 0.0000 0.0000 0.0000
C 0.8165 0.2294 0.7676 0.3531 0.2984 0.1824 0.8206 0.2335
G 0.1835 0.7706 0.2324 0.4878 0.1607 0.8176 0.1794 0.7665
T 0.0000 0.0000 0.0000 0.0000 0.3476 0.0000 0.0000 0.0000

Motif occurrences:

E-value: 0.03375495

     seq pos orient    motif      prob
1   Seq1  69      1 CGCCAGCG 1.0000000
2   Seq8  25      1 GGGCTGCC 1.0000000
3   Seq6  21      1 CCCATGGG 1.0000000
4   Seq2  35      1 CGCGCGCG 0.9998827
5   Seq3  86     -1 CGGACGCG 0.9994568
6  Seq10  14     -1 GGCACGCG 0.9949933
7   Seq5  79     -1 GGCCGGCG 0.9729926
8   Seq4  15      1 CCGGAGCG 0.9269565
```

```
9   Seq7  69     -1 CGGGCGGG 0.9082883
10  Seq9   7      1 CGCCCGCG 0.9020536
```

The `cand` slot of the `cosmo` object consists of a data frame that summarizes the model selection process:

```
> res@cand
```

```
  conSet model width    wCrit   modCrit conCrit
1      1  OOPS     7 2705.698        NA      NA
2      1  OOPS     8 2686.247 -1315.493 133.259
3      1   TCM     7 2720.641        NA      NA
4      1   TCM     8 2706.129 -1324.282      NA
5      2  OOPS     7 2731.628        NA      NA
6      2  OOPS     8 2729.742 -1337.240 142.939
7      2   TCM     7 2736.589 -1342.966      NA
8      2   TCM     8 2738.835        NA      NA
```

Note that the E-value criterion was evaluated for all candidate models to select the optimal motif width for each given combination of model type and constraint. The model type critertion then needs to be only evaluated for that each combination of model type, constraint set and selected motif width. The constraint set criterion, lastly, needs to be evaluated only for the optimal motif widht and model type for each candidate constraint set. In this case, `cosmo` selected a motif width of 8, the one-occurrence-per-sequence model, and the first constraint set, choices that agree well with the data-generating distribution described above.

The alignment of predicted motif occurrences is stored in the `motifs` slot of the `cosmo` output object:

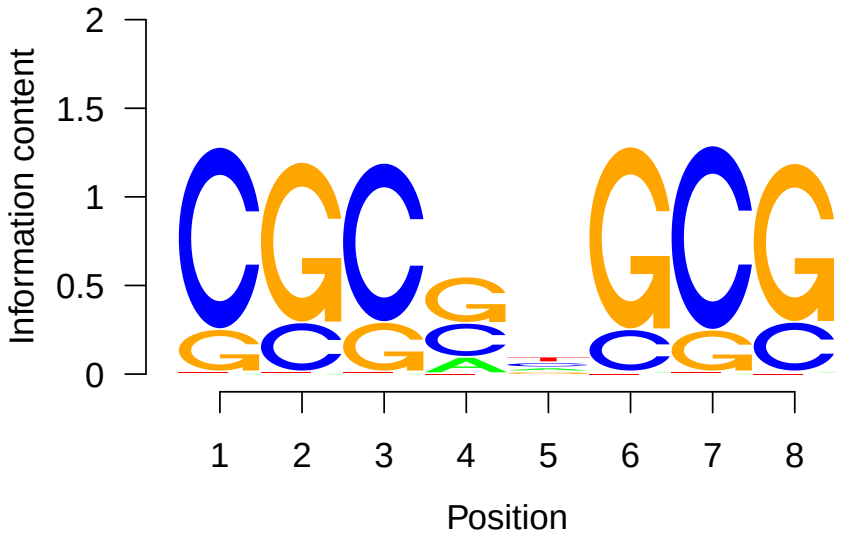
```
> summary(res@motifs)
```

```
Motif sites:

E-value: 0.03375495

     seq pos orient    motif      prob
1   Seq1  69      1 CGCCAGCG 1.0000000
2   Seq8  25      1 GGGCTGCC 1.0000000
3   Seq6  21      1 CCCATGGG 1.0000000
4   Seq2  35      1 CGCGCGCG 0.9998827
5   Seq3  86     -1 CGGACGCG 0.9994568
6  Seq10  14     -1 GGCACGCG 0.9949933
7   Seq5  79     -1 GGCCGGCG 0.9729926
8   Seq4  15      1 CCGGAGCG 0.9269565
9   Seq7  69     -1 CGGGCGGG 0.9082883
10  Seq9   7      1 CGCCCGCG 0.9020536
```

Figure 3: Sequence logo.



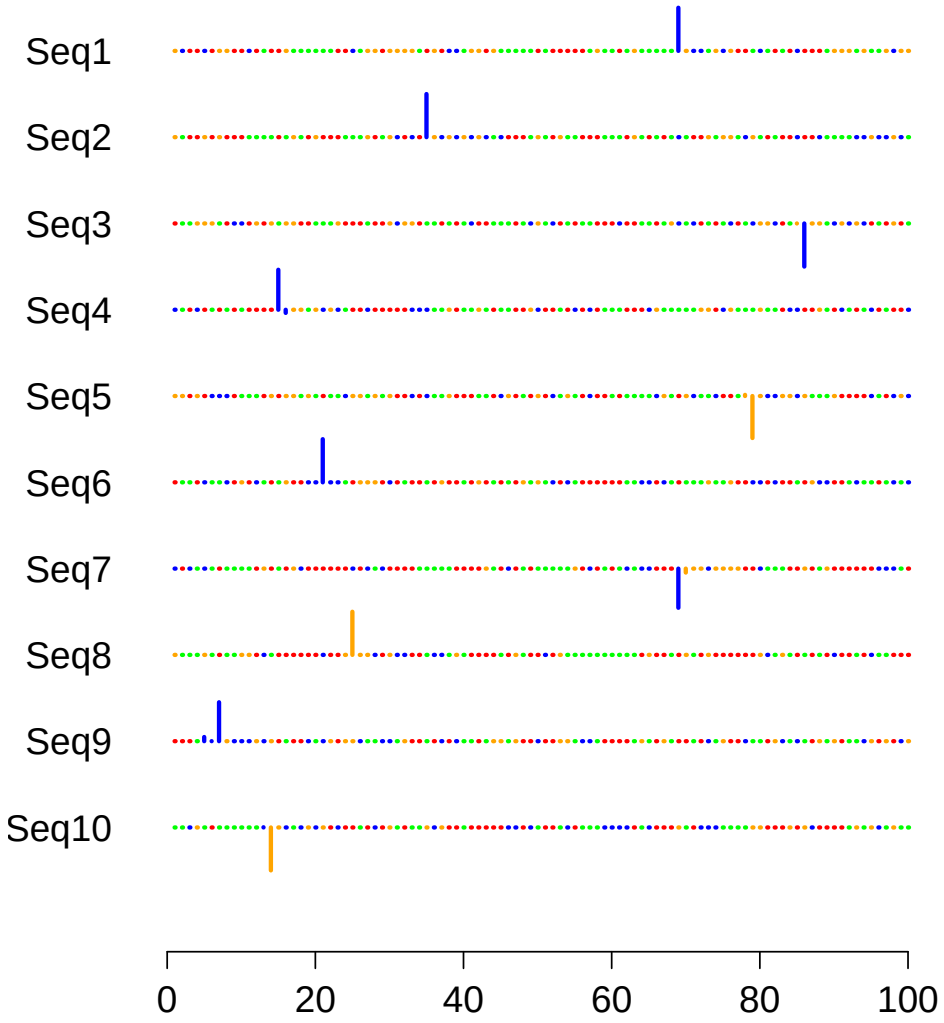
In this, case twenty motif occurrences were predicted, with an E-value for the resulting alignment of 3.3755e-02. Invoking the `plot()` method on the `cosmo` object `res` produces a sequence logo of the estimated motif (figure 3). Note that the sequence logo of the estimated position weight matrix agrees well with the sequence logo of the true position weight matrix shown in figure 1. A plot of the posterior probabilities along each sequence is obtained by invoking the `plot()` method on the `cosmo` object `res` with the argument `type="prob"`(figure 4).

3.5 External estimates of the background Markov model

By default, the Markov model for the distribution of background nucleotides is estimated from the input sequences supplied by the user. It is also possible, however, estimate this model from a separate, usualy larger set of sequences. The user may, for example, want to use the set of intergenic sequences of the organism of interest for this purpose. Estimating the background model from a larger data set allows for more precise estimation and may thus increase the performance of the algorithm in finding shared motifs in the input sequences.

A separate set of sequences for the estimation of the background model may be passed to `cosmo()` through the (backSeqs) argument. As with the `seqs` argument, this may be either the string `browse`, in which a GUI allows the user to browse the file system for a FASTA file containing the sequences to be used, another string pointing to this FASTA file, or a list with each element representing a sequence in the form of a single string such as "ACGTAGCTAG" ("seq" entry) and a description ("desc" entry).

Figure 4: Posterior probability plot. If the motif is more likely to occur in the forward strand orientation, the bar extends upward from the horizontal, otherwise it extends downward.



If the background data set is large, one may wish to estimate the background Markov model in a preliminary step and then pass the obtained estimates to all following calls to `cosmo()`. The function `bgModel()` can be used for this purpose:

```
> args(bgModel)

function (seqs, order = NULL, fold = 5, maxOrder = 6)
NULL
```

Its main argument consists of the sequences from which the background model is to be estimated. If a Markov model of a specific order is desired, this order may be specified through the `order` argument. If `order==NULL`, the appropriate order is chosen data-adaptively through likelihood-based cross-validation. This approach will larger orders, with corresponding models that become more difficult to estimate, as the amount of available data increases. The `maxOrder` argument gives the largest candidate order that is to be considered. To obtain an estimate of the background Markov model from a set of example sequences contained in the file `bgSeqs`, we might use the call

```
> bgFile <- system.file("Exfiles", "bgSeqs", package = "cosmo")
> tm <- bgModel(bgFile)

cvOrder: Order of background Markov model estimated as order = 2 by CV
```

The output produced by `bgModel()` is a list containing the selected order, a summary of the Kullback-Leibler divergences for the different candidate orders, as well as the estimated transition matrices:

```
> tm

$transMat
$transMat$order0
           A         C         G      T
-- 0.3210294 0.1994118 0.1820588 0.2975

$transMat$order1
          A         C         G         T
A 0.3581267 0.1864096 0.1769972 0.2784665
C 0.3621262 0.2045035 0.1513474 0.2820229
G 0.2983023 0.2215036 0.1843169 0.2958771
T 0.2675569 0.1965875 0.2064787 0.3293769

$transMat$order2
           A         C         G         T
AA 0.4163449 0.1570142 0.1898327 0.2368082
```

```
AC 0.3797781 0.1997534 0.1553637 0.2651048
AG 0.3328999 0.2015605 0.1768531 0.2886866
AT 0.3649876 0.1775392 0.1725846 0.2848885
CA 0.3159509 0.2361963 0.1758691 0.2719836
CC 0.4007220 0.2003610 0.1624549 0.2364621
CG 0.2926829 0.2243902 0.1658537 0.3170732
CT 0.2172775 0.2185864 0.2028796 0.3612565
GA 0.3717775 0.1763908 0.1818182 0.2700136
GC 0.3345521 0.2230347 0.1590494 0.2833638
GG 0.3114035 0.2609649 0.1600877 0.2675439
GT 0.2855191 0.1680328 0.2745902 0.2718579
TA 0.3022181 0.1903882 0.1561922 0.3512015
TC 0.3362720 0.1989924 0.1347607 0.3299748
TG 0.2598802 0.2179641 0.2131737 0.3089820
TT 0.1966967 0.2162162 0.2027027 0.3843844


$order
[1] 2

$klDivs
  order          klDiv
1     0  1.085906e+03
2     1  1.082700e+03
3     2  1.077639e+03
4     3  1.078179e+03
5     4 1.797693e+308
6     5            Inf
7     6            Inf
```

The `transMat` element of this list contains one estimated transition matrix for each order between zero and the selected order. The entry in cell (i, j) of a given transition matrix gives the estimated probability of observing nucleotide j in a given position after having observed the tuple i in the previous k positions. The Kullback-Leibler divergences summarized in the `klDivs` element of the output give the estimated risk for each candidate order corresponding to the minus log loss function; likelihood-based cross-validation selects the order with the minimal Kullback-Leibler divergence. The estimated transition matrix may be passed to `cosmo()` through the `transMat` argument:

```
> res <- cosmo(seqs = seqFile, constraints = "None", minW = 8,
+     maxW = 8, models = "OOPS", transMat = tm$transMat)

> res
```

```
       1      2     3      4      5 6     7      8
A 0.0000 0.0000 0.000 0.2700 0.2000 0 0.000 0.0000
C 0.7544 0.1999 0.578 0.3818 0.4513 0 0.778 0.1752
G 0.2456 0.8001 0.422 0.3482 0.1189 1 0.222 0.8248
T 0.0000 0.0000 0.000 0.0000 0.2299 0 0.000 0.0000
```

MEME allows the user to specify the background Markov model in a slightly different format. The files it accepts for specifyint a 1st-order Markov model, for example, are of the form

```
# tuple   frequency_non_coding
a       0.324
c       0.176
g       0.176
t       0.324
# tuple   frequency_non_coding
aa      0.119
ac      0.052
ag      0.056
at      0.097
ca      0.058
cc      0.033
cg      0.028
ct      0.056
ga      0.056
gc      0.035
gg      0.033
gt      0.052
ta      0.091
tc      0.056
tg      0.058
tt      0.119
```

Such files contain estimates of all relevant tuple frequencies. Note that these frequencies are different from the conditional probabilities given in a transition matrix: The tuple frequencies give an estimate of the probability of observing a given tuple, while the frequencies contained in a transition matrix give estimates of the probality of observing a given nucleotide given the previous k nucleotides. Thus, the entries in each row of a transition matrix must sum to 1.0, not the entries in an entire matrix, as is the case with a MEME-style tupe frequency matrix. A MEME-style background file may be passed to `cosmo()` through the `bfile` argument. Alternatively, a MEME-style background file may be converted into a transition matrix by using the function `bfile2tmat()`:

```
> tmat <- bfile2tmat(bfile)
> tmat
```

```
$order0
       A     C     G     T
-- 0.324 0.176 0.176 0.324

$order1
          A         C         G         T
A 0.3672840 0.1604938 0.1728395 0.2993827
C 0.3314286 0.1885714 0.1600000 0.3200000
G 0.3181818 0.1988636 0.1875000 0.2954545
T 0.2808642 0.1728395 0.1790123 0.3672840
```

3.6 Software Design

The following features of the programming approach employed in `cosmo` may be of interest to users.

Class/method object-oriented programming. Like many other Bioconductor packages, `cosmo` has adopted the *S4 class/method objected-oriented programming approach* presented in Chambers (1998). In particular, a new class, `cosmo`, is defined to represent the results obtained by the constrained motif search algorithm. As discussed to some extent above, several methods are provided to operate on this class.

Calls to C. The R package was derived from an earlier stand-alone application that was written entirely in C. This design was necessary to ensure that the computationally intensive constrained optimization algorithm does not take too much time. The constrained optimiziation itself is carried out using the `donlp2()` function by Peter Spellucci, available at `http://plato.asu.edu/ftp/donlp2/donlp2_intv_dyn.tar.gz`.

3.7 License

The `cosmo` package incorporates two sources of foreign code whose free use has been restricted to research purposes. Commercial purposes require permission and licensing by the owners of the copyright to that code. This is true for the `donlp2()` function that is used here to perform the constrained optimization of the likelihood function as well as of code that is used by `cosmo` to compute the E-value criterion. The copyright to `donlp2()` is owned by its author, Peter Spellucci; the copyright to the second piece of code, written by Timoty Bailey, is owned by the Regents of the University of California. For these reasons, the `cosmo` package must likewise be restricted to research purposes, with commercial uses requiring permission by Oliver Bembom, Peter Spellucci, as well as the Regents of the University of California.

The license under with `donlp2()` is distributed furthermore requires that its use must be acknowledged in any publication containing results obtained with `donlp2()` or parts of it. The same is hence true for publications containing results obtained with `cosmo`. Citation of the author's name and netlib-source is suitable for this purpose.

4 Discussion

The Bioconductor package `cosmo` implements a constrained motif detection algorithm that includes as an important special case the popular motif detection tool `MEME`, but also allows the user to enhance the performance of the algorithm by specifying constraints on the position weight matrix to the be estimated.

We note that the same algorithm has also been implemented in the form of a web application, accessible at `http://cosmoweb.berkeley.edu`, that allows users to submit jobs to designated web servers, with results posted in HTML as well as XML format on a temporary web page. In this case, constraint definitions are supplied in a text file according to a straightforward standard. In particular, the R function `writeConFile()` in the `cosmo` package can be used to convert a `constraintSet` or `constraintGroup` object into such a text file. Lastly, we have also posted the source code of the original stand-alone C implementation of the algorithm on this web site.

References

H. Akaike. *Information theory and an extension of the maximum likelihood principle.* Academiai Kiado, 1973.

T.L. Bailey and C.P. Elkan. Unsupervised learning of multiple motifs in biolpolymers using expectation maximization. *Machine Learning*, pages 51–80, 1995.

O. Bembom, S. Keleş, and M.J. van der Laan. Supervised detection of conserved motifs in DNA sequences with `cosmo`. Statistical Applications in Genetics and Molecular Biology: Vol. 6 : Iss. 1, Article 8. URL `http://www.bepress.com/sagmb/vol6/iss1/art8`.

H.J. Bussemaker, H. Li, and E.D. Siggia. Regulatory element detection using correlation with expression. *Nature Genetics*, 27:167–171, 2001.

J.M. Chambers *Programming with Data: A Guide to the S Language.* Springer Verlag, New York, 1998.

E. Davidson. *Genomic Regulatory Systems. Development and Evolution.* Academic Press, San Diego, 2001.

M.B. Eisen. All motifs are not created equal: structural properties of transcription factor - DNA interactions and the inference of sequences specificity. *Genome Biology*, 6:P7, 2005.

M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science*, 95: 14863–14868, 1998.

J.W. Fickett and W.W. Wasserman. Discovery and modeling of transcriptional regulatory regions. *Current Opinions in Biotechnology*, 11:19–24, 2000.

C. Lawrence and A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function and Genetics*, 7:41–51, 1990.

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*, 262:208–214, 1993.

N.M. Luscombe, S.E. Austin, H.M. Berman, and J.M. Thornton. An overview of the structures of protein-DNA complexes. *Genome Biology*, 1:1–37, 2000.

A.F. Neuwald, J.S. Liu, and C.E. Lawrence. Gibbs motif sampling: detection of bacterial outer membrane repeats. *Protein Science*, 4:1618–1632, 1995.

J. Powell. SAGE. The serial analysis of gene expression. *Methods of Molecular Biology*, 99: 297–319, 2000.

F.P. Roth, J.D. Hughes, P.W. Estep, and G.M. Church. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology*, 16:939–945, 1998.

A. Sandelin and W.W. Wassermann. Constrained binding site diversity within families of transcription factors enhances pattern discovery bioinformatics. *Journal of Molecular Biology*, 338:207–215, 2004.

T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, 188:415–431, 1986.

T. D. Schneider, and R. R. Stephens. Sequence Logos: A New Way to Display Consensus Sequences *Nucleic Acid Research*, 18:6097–6100, 1990.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

W. Thompson, E.C. Rouchka, and C.E. Lawrence. Gibbs recursive sampler: finding transcription factor binding sites. *Nucleic Acid Research*, 31:3580–3585, 2003.

M.J. van der Laan, S. Dudoit, and S. Keleş. Asymptotics optimality of likelihood based cross-validation. Technical Report 125, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper125`.

C.T. Workman and G.D. Stormo. ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. In *Proceedings of the Pacific Symposium on Biocomputation*, pages 467–478, 2000.