Supporting Information

Probabilistic Clustering of Sequences: Inferring new bacterial regulons by comparative genomics

Erik van Nimwegen, Mihaela Zavolan*, Nikolaus Rajewsky, and Eric D. Siggia
Center for Studies in Physics and Biology and *Laboratory of Computational Genomics,
the Rockefeller University, 1230 York Avenue, New York 10021 NY

Contents

1. The Weight Matrix Representation of Binding Sites

In this section we briefly explain which assumptions are implied in the weight matrix (WM) representation of transcription factor binding sites. Transcription factors (TFs) will bind specifically to certain DNA segments in the genome. We

wish to mathematically represent the sequence features, shared by these segments, that are responsible for the specific recognition by the TF. It is generally assumed (4) that the binding energy $E(s)$ of the TF to a DNA sequence segment s can be written as a sum of binding energies $E_i(s_i)$ for the different bases in the segment:

$$E(s) = \sum_{i=1}^{l} E_i(s_i) \qquad \textbf{[1]}$$

The probability $P_{\text{bound}}(s)$ that the TF binds to a sequence s in the genome is then proportional to

$$P_{\text{bound}}(s) \propto e^{\beta E(s)}, \qquad \textbf{[2]}$$

where $\beta = 1/(kT)$. We now further assume that binding sites for a particular TF are distinguished from all other DNA segments in that they have higher binding energies. That is, binding sites are characterized by having some expected binding energy $\langle E \rangle$ which is substantially higher than the expected binding energy of random sequences. Under these assumptions, the probability distribution $P(s)$ that segment s is a binding site for the TF is given by the maximum entropy distribution under the condition $\langle E(s) \rangle = \langle E \rangle$,

$$P(s) = \frac{e^{\lambda E(s)}}{\sum_{s'} e^{\lambda E(s')}} = \prod_{i=1}^{l} \frac{e^{\lambda E_i(s_i)}}{\sum_{\alpha} e^{\lambda E_i(\alpha)}}, \qquad \textbf{[3]}$$

where the sum over s' is over all length-l sequences, and the sum over α is over the four bases. The Lagrangian multiplier λ is chosen such that $\sum_s E(s)P(s) = \langle E \rangle$. If we define the WM

$$w_\alpha^i = \frac{e^{\lambda E_i(\alpha)}}{\sum_{\alpha'} e^{\lambda E_i(\alpha')}}, \qquad \textbf{[4]}$$

then we can represent TFs by WMs, and write for the probability $P(s|w)$ that s is a binding site for the TF represented by w as

$$P(s|w) = \prod_{i=1}^{l} w_{s_i}^i. \qquad \textbf{[5]}$$

In summary, under the assumption that the binding energy of a TF to a DNA segment s can be written as a sum of the binding energies of the individual bases in s to the TF, and assuming that binding sites for a TF can be characterized by having a certain expected binding energy $\langle E \rangle$, we can represent TFs by WMs w. For each length-l DNA segment s, the probability of this segments being a binding site for TF w is simply given by a product of the WM probabilities of the individual bases, as described above.

1.1. Information Scores

If we have an alignment of n length-l sequences, with n_α^i occurrences of base α at position i, we can ask for the probability $P(n_\alpha^i|b_\alpha)$ of observing these

base counts under the assumption that all these sequences are random, with probabilities b_α for base α occurring at each position. This probability is given by

$$P(n_\alpha^i|b_\alpha) = \prod_{i=1}^{l} n! \prod_\alpha \frac{(b_\alpha)^{n_\alpha^i}}{n_\alpha^i!}. \quad \textbf{[6]}$$

If we use the Stirling approximation for the factorials (valid for n moderately large):

$$\log(n!) \approx n\log(n) - n, \quad \textbf{[7]}$$

and write for the WM entries

$$n_\alpha^i = n w_\alpha^i, \quad \textbf{[8]}$$

we obtain

$$P(n_\alpha^i|b_\alpha) = \exp\left[-n\sum_{i,\alpha} w_\alpha^i \log(w_\alpha^i/b_\alpha)\right] \equiv e^{-nI}, \quad \textbf{[9]}$$

where we have defined the WM information score I as

$$I = \sum_{i,\alpha} w_\alpha^i \log(w_\alpha^i/b_\alpha). \quad \textbf{[10]}$$

Thus, the higher the information score of the observed alignment, the less likely it is to occur by chance from sampling random bases. This is the main reason that most authors use the information score to assess the quality of alignments of putative binding sites.

Obviously, for small n_α^i and especially if some of the n_α^i become zero, the Stirling approximation breaks down, and it would be best to use the exact Eq. **9**. The information score can thus best be *defined* as

$$I = -\frac{\log[P(n_\alpha^i|b_\alpha)]}{n} \quad \textbf{[11]}$$

from the exact expression for all values of n.

2. The Partition Likelihood Function

We recall that the probability $P(S|w)$ that all sequences of a set S were drawn from a WM w is given by

$$P(S|w) = \prod_{s\in S}\prod_{i=1}^{l} w_{s_i}^i, \quad \textbf{[12]}$$

where s_i is the base occurring at position i in sequence s. To calculate the probability $P(S)$ that all sequences in a set S were drawn from the *same* WM, independent of what this WM may be, we integrated over the hypersimplex

$$\sum_\alpha w_\alpha^i = 1,\ \forall i \in \{1, 2, \ldots l\}. \quad \textbf{[13]}$$

Generally, one would have to choose some integration measure $\pi(w)$ on this space (which is equivalent to choosing a prior). We then have

$$P(S) = \int \pi(w) P(S|w) dw. \quad \textbf{[14]}$$

We chose a uniform prior that normalizes the integrals

$$\pi(w) = (3!)^l. \quad \textbf{[15]}$$

In cases where the sampling distribution is multinomial (as it is in our case), it is customary to use so-called Dirichlet priors

$$\pi(w) = \prod_{i,\alpha} (w_\alpha^i)^{\lambda_\alpha - 1}. \quad \textbf{[16]}$$

Demanding a prior that is invariant under scale transformations of the w_α^i, one would arrive at $\lambda_\alpha = 0$. There are good arguments for suggesting that this corresponds to a complete *ignorance* prior for w (5). It is also easy to derive that setting λ_α to some value larger than zero is equivalent to adding λ_α counts for each letter α to the data. For this reason, the λ_α are also referred to as pseudo-counts. We interpret our use of $\lambda_\alpha = 1$ as representing the knowledge that it is definitely *possible* for each base α to occur at each position. That is, we known $w_\alpha^i > 0$ for all α. In any case, these are subtleties that hardly affect the inference. For instance, some authors (e.g., ref. (6)) prefer to choose λ_α proportional both to the frequency of α in the "full" data set (being for instance the genome or all of its intergenic regions) and proportional to the square root of the number of sequences in the alignment. We, however, fail to see how a representation of our *prior* knowledge should depend on the data set, nor why WMs should *a priori* be likely to be skewed toward bases that have a higher frequency of occurrence in the full genome or its upstream regions (which consist mostly of DNA that is *not* part of a binding site). Still, we have experimented with such priors and noticed that the clustering behavior is hardly affected at all. We thus chose to use the uniform prior since it is simpler, and more easily justifiable theoretically.

Finally, some remarks on the importance of the integration over w in calculating the probability of a cluster. First of all, it is of course simply a matter of probability theory that if we want to calculate the probability that all sequences in the cluster came from the same WM, in absence of any knowledge about this WM, we have to integrate over w. However, one may for instance be tempted to assign a "score" to each cluster by finding the WM w^* that maximizes the likelihood $P(S|w^*)$ that all sequences in the cluster came from this WM. One would find

$$P(S|w^*) = \prod_{i=1}^{l} \prod_{\alpha} \left(\frac{n_\alpha^i}{n} \right)^{n_\alpha^i}, \quad \textbf{[17]}$$

where the maximum likelihood (ML) WM has entries $(w^*)_\alpha^i = n_\alpha^i / n$.

The problem with an approach like this is that, trivially, the partition in which each sequence forms its own cluster will be the partition that maximizes the score. That is, for single-sequence clusters one can always find a WM that gives rise to that sequence with probability 1. Obviously, we would not want this totally "unclustered" state to have the highest score. The source of the problem is that to obtain these ML scores, one has to choose all WMs such that the likelihood is maximized. The more clusters there are in the partition, the more WMs have to be set to their ML values. One thus allows oneself to optimize over more degrees of freedom for a partition with many clusters than for a partition with a small number of clusters. One then might be tempted to introduce some kind of "penalty" per degree of freedom that has to be optimized to obtain the ML score. However, these problems are solved automatically when one applies probability theory correctly, that is, when one performs the integration over w.

2.1. Estimating WM Entries

Given an alignment of sequences that we believe to be sampled from the same WM, we would like to estimate the WM entries w_α^i from which these sequences were sampled. In general we have for the expected WM entry $\langle w_\alpha^i \rangle$

$$\langle w_\alpha^i \rangle = \frac{\int w_\alpha^i P(S|w)\pi(w)dw}{\int P(S|w)\pi(w)dw}, \qquad \textbf{[18]}$$

where the integral is again over the hypersimplex, and $\pi(w)$ is again the prior on this space. With the general Dirichlet prior one finds

$$\langle w_\alpha^i \rangle = \frac{n_\alpha^i + \lambda_\alpha}{\sum_\beta n_\beta^i + \lambda_\beta}. \qquad \textbf{[19]}$$

With our prior $\lambda_\alpha = 1$, we thus have $\langle w_\alpha^i \rangle = (n_\alpha^i + 1)/(n + 4)$.

2.2. Relation to the Gibbs Sampler

Consider a situation in which there only two clusters: one very large "reservoir" cluster and one small cluster (which contains the estimated alignment of binding sites for a single TF). Under our model, this partition has a certain probability P. We are now interested in calculating the *change* $P \to P'$ in that probability when a single sequence is moved out of the reservoir cluster into the smaller cluster. With n_α^i, the number of bases α in column i before the move and N_α the number of bases α in the "background reservoir" (which is assumed to have equal base frequencies b_α in each column), this is given by

$$P' = P \prod_{i=1}^{l} \frac{n_\alpha^i + 1}{n + 4} \frac{N + 3}{N_\alpha} \equiv PQ. \qquad \textbf{[20]}$$

In the limit of N goes to infinity the factor Q reduces to

$$Q = \prod_{i=1}^{l} \frac{n_\alpha^i + 1}{(n + 4)b_\alpha} = \prod_{i=1}^{l} \frac{\langle w_\alpha^i \rangle}{b_\alpha}, \qquad \textbf{[21]}$$

with b_α the background base frequencies and $\langle w_\alpha^i \rangle$ the expected WM entries based on the n members of the smaller cluster prior to the addition of the new sequence. This factor Q is precisely (assuming uniform pseudo-counts) the scoring that is used by the Gibbs sampler algorithm. Thus, under the assumption that the data set consists of a large set of "background" sequences plus a set of binding sites for a single TF, our scoring reduces to the scoring of Gibbs sampler. Note, however, that assuming a single cluster plus background is of course not correct for data sets of the type that we cluster in the paper.

2.3. Prior on the Space of Partitions

In the paper, we are using a uniform prior over all possible partitions. The argument for using such a prior is that it is the maximum entropy prior with respect to the space of partitions. In absence of any prior knowledge about which partitions are more or less likely, a uniform prior is the least "assuming" of any prior. We note, however, that there are some subtleties with this reasoning. Instead of saying that we are *a priori* completely ignorant regarding which partition is more or less likely, it may seem that we may just as well have said that we are completely ignorant regarding the number of clusters that underlie the data. These two statements are mutually inconsistent, however. The number of ways of partitioning a set of N objects into n subsets is given by a so-called Stirling number of the second kind S_n^N. The total number of ways of partitioning a set of N objects into subsets is

$$b_N \equiv \sum_{n=1}^{N} S_n^N . \qquad [22]$$

A uniform prior over the partitions thus corresponds to a probability distribution

$$P_n = \frac{S_n^N}{b_N} \qquad [23]$$

over the number of clusters n. This distribution has, for large N, a relatively sharp peak at some value of n. Therefore, complete ignorance regarding the partition "induces" knowledge regarding the number of clusters.

We could of course have chosen a uniform prior over the number of clusters, thereby increasing the relative weight of partitions with cluster numbers that can be realized in fewer ways. We are, however, not trying to infer the cluster *number*, we are inferring which particular partitions are most likely. We therefore feel that a uniform prior of partitions is the most relevant for our purposes.

3. Monte Carlo Sampling

In this section we explain in more detail how our Monte Carlo sampling is implemented. First of all, we want to implement a move set that would sample

all partitions of our set of mini-WMs equally often if the probability distribution $P(D|C)$ were *constant*. That is, we want to implement a move set that respects the uniform prior. This can be done as follows.

Let our data set contain N mini-WMs and imagine that we have N boxes. A clustering state can be specified by assigning the N mini-WMs to the N boxes (leaving some boxes empty of course). There are N^N such states. This state space (call it X) can be easily uniformly sampled by, at each time step, picking one of the N objects at rando, and moving it to one of the $N-1$ other boxes, which is also to be chosen at random. However, a uniform sampling of X does not correspond to a uniform sampling of the space of partitions. For each possible partition C of the data set, there is a multitude of states in the state space X that correspond to this partition C. To be precise, for a partition C containing n clusters there are $N!/(N-n)!$ states in the state space X that correspond to the same partition C (all ways of permuting the N boxes, divided by all ways of permuting the empty boxes). Therefore, partitions with $n+1$ clusters have $N-n$ times as many states in X as partitions with n clusters. We thus bias the move-set on X such that the rate of moving from a state with $n+1$ clusters to a state with n clusters is $N-n$ times as high as the backward rate. This will ensure a uniform sampling of the space of partitions.

A second issue that the Monte Carlo walk has to deal with is the aligning of the mini-WMs with respect to each other. As mentioned in the text, all our mini-WMs are 32 bases long, while we assume that the binding sites are 27 bases long. There are six ways of placing a length 27 window over the length 32 mini-WM. Every time a mini-WM is moved during the random walk, we sample over these six possible shifts *and* over both strands of the DNA (leading to 12 possible ways of picking the "site" from the length 32 window). This sampling is done by so-called importance sampling.

Assume that we want to consider moving mini-WM m from a cluster containing the set of sequences S into a cluster containing the set of sequences S'. Let $P(S')$ be the probability of the set S' without m added, and let $P(S^-)$ be the probability of the set of sequences S when m is *removed*. Further, let $P(S', i, s)$ be the probability of the cluster S' when mini-WM m is added, with a shift i and using strand s (where i runs from 0 to 5, and s is either positive or reverse complement). Finally, let $P(S^-, i, s)$ be the probability of cluster S with m added at shift i and strand s. We then define

$$Z = P(S') \sum_{i=0}^{5} \sum_{s \in \{+,-\}} P(S^-, i, s), \qquad \textbf{[24]}$$

and

$$Z' = P(S^-) \sum_{i=0}^{5} \sum_{s \in \{+,-\}} P(S', i, s). \qquad \textbf{[25]}$$

The move of m from S to S' is accepted when $Z' \geq Z$, and accepted with probability Z'/Z when $Z' < Z$. After m is assigned to either S or S', one of the 12 strand/shift combinations is sampled with probability $P(S^-, i, s)/Z$ (or

$P(S', i, s)/Z'$ depending on which cluster m was assigned to). This extension of the move set will sample over all possible shift and strand combinations that the clusters can take.

With this move set, it still seems that stable clusters may get "trapped" into an unfavorable positioning of their window. That is, assume that some subset of mini-WMs has high similarity and can form a stable cluster. Let us also assume that these sites are not shifted with respect to each other, so that with high likelihood an alignment will be sampled in which all mini-WMs in the cluster occur with the same shift. Still, one may choose different values for this shift, i.e. the alignment may contain bases 0-26, or bases 1-27, etc. Once one of these shifts is chosen our move set makes it very unlikely for the cluster to revert to another shift unless it evaporates. To defeat clusters getting trapped in a particular shift this way we implemented one more move: the "coherent shift". In this move, we sample, according to their probability, from all ways of shifting all mini-WMs in the cluster by the same amount. This sampling is performed (with probability p) on the cluster that contains the mini-WM that was randomly selected to be reassigned. In our implementation we chose $p = 0.05$. The behavior of the algorithm is insensitive to this value. One only has to make sure that the coherent shift is tried at least once during the lifetime of a typical cluster.

Finally, we altered the probability $P(S)$ for clusters consisting of a single mini-WM. For those single mini-WM clusters, we interpret the mini-WM as stemming from a random background distribution as opposed to it consisting of samples from a binding sites. That is, instead of a score 4^{ln} (with l the length and n the number of sequences in the mini-WM) we use a probability

$$P = \prod_{s \in S} \prod_{i=1}^{l} b_{s_i}, \qquad \textbf{[26]}$$

where S is the set of sequences that make up the mini-WM, and b_α is the back ground frequency of letter α in the upstream regions. It must be admitted that this is somewhat of a hybrid procedure. If we had *really* wanted to consider a hypothesis space in which some of the mini-WMs can be "background" sequences as opposed to WM samples, then we should have chosen a prior on the space of partitions that properly reflects this.

A typical Monte Carlo run has on the order of 10^{10} (considered) moves. It takes less than 10^9 moves to reach "equilibrium". That is, for the first $10^8 - 10^9$ time steps the sampled values of $P(D|C)$ are decreasing, whereas they fluctuate around some fixed value after 10^9 time steps. For the data set from ref. (1) this equilibrium value is given by

$$\log[P_{\rm ss}] \approx 0.93 \log[P_{\rm uc}], \qquad \textbf{[27]}$$

where $P_{\rm ss}$ is the steady-state value of the probability $P(D|C)$ and $P_{\rm uc}$ is the probability of the entirely unclustered state where each mini-WM forms a cluster by itself.

3.1. Annealing

The annealing is performed by raising all probabilities to the power β and slowly raising β over time. That is, a move form partition C to C' is accepted with probability $[P(D|C')/P(D|C)]^{\beta}$.

We experimented with different annealing schedules, and a simple linear increase of β with time gives the best results. We let the algorithm run at $\beta = 1$ for 10^8 steps and then start increasing β linearly with time, such that $\beta \approx 3$ in the end phase of the annealing. The last $5 * 10^7$ time steps are then done with a very high value of β, essentially only accepting moves that increase the probability $P(D|C)$, thereby leading to a locally optimal partition at the end of the run.

4. Extracting Significant Clusters

The simplest way in which we could use our assignment of probabilities to partitions to extract clusters is by treating the probabilities simply as a "scoring" function and attempting to find a partition of optimal score. One could for instance use hierarchical clustering:

1. Start with each mini-WM forming its own cluster.
2. Calculate, for all pairs of clusters, the change in the probability of the partition when these clusters are fused (maximizing for each pair over all possible shift and strand combinations).
3. When there is no fusion that increases the partition's probability: stop. Else, fuse those clusters that increase the partition's probability most and go to step 2.

Experiments with the test set of 397 known TF binding sites (3) show that (choosing cut offs as favorably as possible) no more than 10 regulons are correctly inferred using this procedure. Moreover, such greedy-search algorithms only search a minute portion of the state space. But most importantly, such schemes ignore one of the key advantages of our method, which is that it automatically assigns probabilities to clusters.

4.1. Tracking Stable Clusters

One way of identifying significant clusters from the Monte Carlo sampling is to look for clusters that are long-lived. A cluster is "born" when a mini-WM is assigned to form a pair with a mini-WM that is forming a cluster by itself. We then follow this cluster until it "evaporates", which occurs when only a single mini-WM is left. We record which mini-WMs were part of this cluster during its lifetime. That is, we compile a probabilistic membership distribution p, where p_i is the percentage of the lifetime of the cluster that mini-WM i was assigned to this cluster. We only keep track of such lifetime membership distributions

for clusters whose lifetimes are over some threshold (short transient formations of pairs, for instance, are not recorded).

At the end of the Monte Carlo run, we will have a large collection of such clusters c, with their lifetimes T_c, and their membership distributions p^c. There generally will be sets of clusters that have very similar membership distributions. We want to consider such sets of similar clusters as different samples of the *same* cluster. That is, we imagine that there is some set of "real" and "unique" clusters, and that the clusters that we recorded during the Monte Carlo random walk are *samples* from this set of underlying clusters. We introduce a probability $P(p^c, p^{c'})$ that clusters c and c' are samples from the same underlying cluster. We imagine that each underlying cluster has some intrinsic membership distribution p and that our example clusters p^c and $p^{c'}$ are samples of size T_c and $T_{c'}$ of this intrinsic membership distribution. The probability $P(p^c, p^{c'})$ then can be derived analogously to the way in which we derived the probability $P(S)$ for all sequences in a set S to come from the same WM.

For each component i, we have $n_i^c = T_c p_i^c$ for the number of times that mini-WM i occurred in cluster c. Now given the intrinsic membership probability p_i we would have

$$P(n_i^c, n_i^{c'}|p_i) = (p_i)^{n_i^c + n_i^{c'}} (1 - p_i)^{T_c + T_{c'} - n_i^c - n_i^{c'}}. \quad \textbf{[28]}$$

Because we do not known the value of p_i, we again integrate over it

$$P(n_i^c, n_i^{c'}) = \int_0^1 P(n_i^c, n_i^{c'}|p_i) dp_i = \frac{(n_i^c + n_i^{c'})!(T_c + T_{c'} - n_i^c - n_i^{c'})!}{(T_c + T_{c'} + 1)!} \quad \textbf{[29]}$$

Similarly, for the probability that these two samples did *not* come from the same distribution, we have

$$P(n_i^c)P(n_i^{c'}) = \frac{n_i^c! n_i^{c'}! (T_c - n_i^c)!(T_{c'} - n_i^{c'})!}{(T_c + 1)!(T_{c'} + 1)!}. \quad \textbf{[30]}$$

Therefore, the increase in the probability $P \to P'$ when clusters c and c' are fused, is given by

$$P' = P \prod_{i=1}^{N} \frac{P(n_i^c, n_i^{c'})}{P(n_i^c)P(n_i^{c'})}. \quad \textbf{[31]}$$

We have implemented a hierarchical clustering that starts out assuming that all recorded clusters are unique and then iteratively fuses those clusters for which the increase in probability is highest. We then cut this procedure off at a more or less arbitrarily chosen point.

We found this procedure (although workable and giving similar results for the significant clusters) ultimately unsatisfactory because there is no natural way to cut off the hierarchical clustering, and because this is again a greedy algorithm that only searches a small part of the space. In a sense, we end up with the same clustering problem all over again. We sampled the distribution $P(C|D)$ and obtained a set of clusters. However, since these clusters are clearly

not unique, we essentially have to cluster this set of clusters to obtain unique ones! (We could of course sample again all ways of fusing the clusters... and then end up with probabilistic clusters of clusters... and so on *ad infinitum*).

The key conceptual difficulty here is that there is no clear definition of a *probabilistic cluster*. A partition is a rigid assignment of objects into groups, and since our probability assignment gives an essentially fluid assignment of objects into groups, it it not clear how to extract "clusters" from this fluid assignment.

It is therefore that we chose to simply identify the ML rigid assignment of the mini-WMs into clusters by annealing. At the end of the annealing, we have a rigid assignment of mini-WMs into groups. Our Monte Carlo sampling is then used to estimate the significance of these clusters. This is currently the most satisfying approach that we have for identifying significant clusters, and it is the one we used on the test set of 397 known binding sites from ref. (3).

4.2. Estimating Significance of ML clusters

The annealing essentially gives us a set of "candidate" clusters without determining the significance of these clusters. We only know that a search for the ML partition has partitioned the data into these clusters. We measure the significance of these candidate clusters by a Monte Carlo sampling run.

Let M be the set of mini-WMs making up a particular candidate cluster. At each particular time step during the Monte Carlo sampling this set of mini-WMs will be partitioned over a certain number of clusters. Some members may occur together in a cluster, while other members may be partitioned with some other mini-WMs. In general, the set of members M of the candidate cluster will be spread over a set of clusters c, that each contain m_c members of the set M (with $\sum_c m_c = |M|$, of course). We now find the maximum $m_* = \max_c(m_c)$ of these values m_c and say that at this particular time step m_* members of the candidate cluster M are "coclustered" (that is, we find the cluster that contains most of the members of M and count how many members of M are in this cluster). By recording m_* at each time step, we calculate a distribution $p(m)$ for the probability of m members of M coclustering. We can then of course calculate mean, variance and so forth of this distribution. In particular we will calculate the shortest interval $[m_-, m_+]$ that contains 95% of the probability: $\sum_{m=m_-}^{m_+} p(m) \geq 0.95$. If $m_- > 1$ we consider the cluster to be "significant".

We can also measure, for each member of M, what fraction of the time p it is a member of the set of m_* members. We then obtain a probabilistic membership list for this cluster. That is, for each member m we obtain the probability p_m that m is "in" the cluster.

4.3. Extracting Significant Clusters from Pairwise Statistics

For larger data sets it may be computationally unfeasible to converge all of the statistics just mentioned. We may then need to resort to measuring a simpler

set of statistics to infer significant clusters.

Our approach is to measure, for each pair of mini-WMs i and j, the fraction of the time p_{ij} that these mini-WMs are coclustered, i.e. occur in the same cluster. We do several (20 or so) Monte Carlo runs and measure all p_{ij} in each of them. By comparing the measured p_{ij} from the different runs we can obtain both the mean and the standard deviation between different runs for each p_{ij} (giving us a direct handle on the convergence). For a data set of N mini-WMs, we thus measure mean μ and standard deviation σ for each of the $N(N-1)/2$ pair statistics p_{ij}. In order to illustrate the convergence attained in our runs Fig. 6 shows the cumulative distribution of the ratio $r = \sigma/\mu$ of the standard deviation and mean for all pairs that have a mean of over half, $\mu > 1/2$. (We focus on these because these are the pair statistics that will be used below.) The figure shows, for instance, that 50% of the pairs have a standard deviation

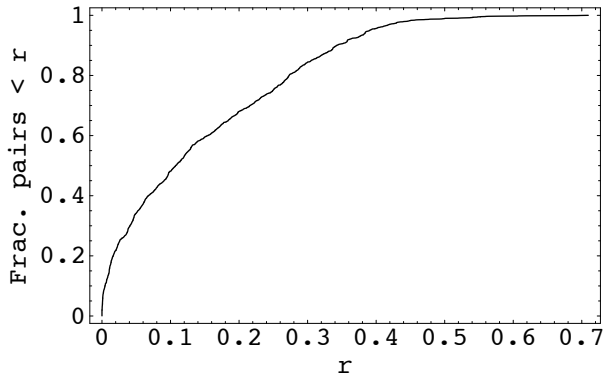

Fig. 6. Cumulative distribution of the ratio r of standard deviation to mean for all pair statistics with a mean larger than 1/2.

smaller than 10% of the mean.

We now use the pair statistics to define "candidate" clusters. We construct a graph where each node represents a mini-WM and connect those nodes i and j for which $p_{ij} > p_{\rm crit}$, where $p_{\rm crit}$ is some connectivity threshold. This graph will contain a certain number of connected components. Some of these components will consist of single nodes. These unconnected mini-WMs are "orphans" in the sense that there are no other mini-WMs with which they cluster more than $p_{\rm crit}$ of the time. All other mini-WMs have at least one partner with which they cluster more than a fraction $p_{\rm crit}$ of the time. Fig. 7 shows the fraction of the data set that has at least one partner over $p_{\rm crit}$ as a function of $p_{\rm crit}$. Fig. 8 shows the number of connected components, i.e. the number of candidate clusters, as a function of $p_{\rm crit}$. We see that when the threshold $p_{\rm crit}$ is set very high, for instance at 95%, that there are slightly less than 100 connected components, containing somewhere between 10 and 20% of the data set. As the threshold $p_{\rm crit}$ is lowered, many new components appear, and the fraction of the data set contained in these components grows along with it. At some point

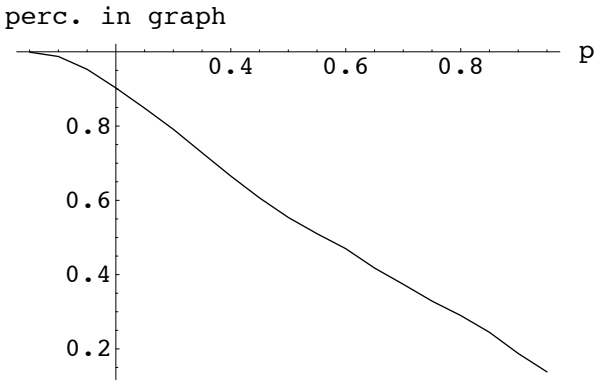

Fig. 7. Fraction of the data set that has at least one other mini-WM with which it clusters more than a fraction p of the time. The data set consists of the top $2,000$ sites from ref. (1).

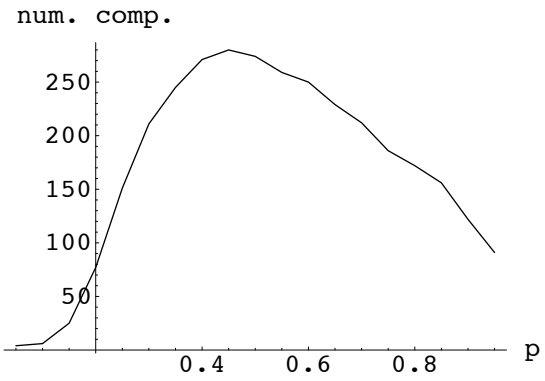

Fig. 8. Number of connected components (with more than 1 member) as a function of the connectivity threshold p. The data set consists of the top 2000 sites from ref. (1).

(around $p_{\text{crit}} = 0.45$), the number of components reaches a maximum. For lower thresholds the number of components drops quickly. What happens here is that the different clusters start fusing rapidly for lower thresholds.

We chose to set the threshold at $p_{\text{crit}} = 1/2$. One reason for setting the threshold at this value is that it is conceptually most appealing: a mini-WM belongs to a cluster if there is a mini-WM within that cluster with which it co-clusters *most* of the time. By setting the threshold at $1/2$ we are guaranteed that each mini-WM cannot cluster more with any other mini-WM than with those in the cluster that it is assigned to. A second reason is pragmatic: Fig. 8 shows that around this value of the threshold the number of different components is maximal, i.e. we will obtain a maximal number of candidate clusters.

Now that we have our candidate clusters, we want to calculate their significance by calculating the distribution $p(m)$ of the number of its coclustering members (see section 4.2). We also want to assign, to each of its members m, a membership probability p_m. Both of these we want to infer from the pairwise statistics (since it is computationally prohibitive to measure all these statistics directly by sampling).

The coclustering probabilities $p(m)$ should in principle be inferred by first estimating the probability of coclustering for all possible subsets of the candidate cluster. We would for instance need to estimate, from the pair statistics, the probability p_{ijk} that members i, j, and k all three co-occur in a single cluster. That is, we need to estimate higher order coclustering statistics from the pair statistics, i.e. given p_{ij}, p_{ik}, and p_{jk} we want to estimate p_{ijk}. This is, in principle, a well defined probability theory problem. We have equalities such as

$$p_{ij} = p_{ijk} + p_{ij,k}, \qquad \textbf{[32]}$$

where $p_{ij,k}$ is the probability that i and j but not k occur together in a cluster. There are two analogous equalities for p_{ik} and p_{jk}. Finally, the probability $p_{i,j,k}$ for all three to occur in separate clusters has to be nonnegative, $p_{i,j,k} \geq 0$, and the sum over all possible ways of partitioning i, j, and k should of course be 1:

$$p_{ijk} + p_{ij,k} + p_{i,jk} + p_{ik,j} + p_{i,j,k} = 1. \qquad \textbf{[33]}$$

Given no other information than the pair statistics and the above constraints one then finds a uniform distribution over p_{ijk} within the interval bounded by $\max(\{(p_{ij}+p_{ik}+p_{jk}-1)/2, 0\})$ from below and $\min(\{p_{ij}, p_{ik}, p_{jk}\})$ from above. The expected value $\langle p_{ijk} \rangle$ would then be given by

$$\langle p_{ijk} \rangle = \frac{\min(\{p_{ij}, p_{ik}, p_{jk}\}) + \max(\{(p_{ij} + p_{ik} + p_{jk} - 1)/2, 0\})}{2}. \qquad \textbf{[34]}$$

In principle, all higher order statistics could be estimated in this way. However, this sort of inference quickly becomes very cumbersome as more indices are involved.

Instead, we estimate membership probabilities by a simpler (approximate) procedure. We assume that one of the members of the connected component,

can be considered the "anchoring" or central member of the cluster and that the probability for a mini-WM to be a member of the cluster is the probability that it coclusters with the anchoring member. We first calculate the probabilities A_i that mini-WM i is the anchoring member. Then, given the A_i, the cluster membership probability p_i of mini-WM i is given by

$$p_i = A_i + \sum_{j \neq i} p_{ij} A_j. \qquad \textbf{[35]}$$

That is, either i is the anchoring member itself, or j is the anchoring member and i coclusters with it. We now define the A_i to be the normalized solution to

$$A_i = \lambda \sum_j p_{ij} A_j. \qquad \textbf{[36]}$$

The idea is that the probability that i is the central member of the cluster, should be proportional to the probability that i coclusters with the central member.

We thus solve the above eigenvector equation for each connected component of the graph, obtain the anchor probabilities A_i and calculate the membership probabilities for all its nodes using Eq. **35**.

Finally, we now use the membership probabilities to calculate the probabilities $p(k)$ that k members from a connected component cocluster at any point in time. To this end, we will assume that for each member m, the probability p_m of this member being part of the cluster, is *independent* of the probability of any of the other members being part of the cluster. That is, each member m has an independent probability p_m to be "in" the cluster. If we then define the generating function

$$G(z) = \prod_m (p_m z + 1 - p_m) = \sum_k p(k) z^k, \qquad \textbf{[37]}$$

the probabilities $p(k)$ can be easily obtained by expanding $G(z)$, and we can again calculate the 95% probability interval $[k_-, k_+]$. Significance of the cluster again is defined by $k_- > 1$.

4.4. Estimating a Cluster's WM

We now want, for each cluster, to estimate a WM from the membership probabilities p_m of its members m. The approach is that if m is in the cluster with probability p_m, that, with the same probability p_m, the cluster WM is given by the alignment of sites that m belongs to. Therefore, we want to calculate the average alignment of sites that each member m belongs to. Thus, during the Monte Carlo sampling, we keep track, for *each* mini-WM i, of the alignment of the mini-WMs of the cluster in which i finds itself. That is, at any point during the run, mini-WM i will find itself in some cluster c, containing some set of other mini-WMs. The alignment of the cluster is simply described by the numbers of n_α^i of bases α at column i of the cluster. We will now keep track,

for each mini-WM, of the running average of this alignment. We of course have to take into account that the mini-WMs under consideration may occur in the cluster with a different length 27 window sampled at different time steps. Thus if at a certain point in time, the mini-WM occurs in the cluster with the window s through $s+27$, then we will add the base counts n_α^i of the current cluster to columns $i+s$ of the mini-WMs WM running average. In this way, we get an average alignment for each of the 32 columns of the mini-WM.

At the end of the sampling, we thus have, for each mini-WM, the averaged alignment of the sites in the clusters that it visited during the run. We will now reconstruct the WM of each "candidate" cluster (be it a ML cluster, or one inferred from pair statistics) by summing the averaged alignments of all members m of the cluster, each weighted with their membership probability p_m. To meaningfully do this sum, we still have to align the averaged alignments of the different cluster members with respect to each other. To this end, we start with the averaged alignment of the member with the largest membership probability p_m. We then align the averaged alignment of the member with the second highest membership probability to this first member. After that, we align the member with the third highest membership probability to this pair, and so on, until all members have been added to the alignment. For each column we then obtain averaged base counts $\langle n_\alpha^i \rangle$. We finally obtain the WM estimates from these averaged base counts:

$$\langle w_\alpha^i \rangle = \frac{\langle n_\alpha^i \rangle + 1}{\langle n \rangle + 4} \qquad \textbf{[38]}$$

4.5. Membership Based on WM Match

For each cluster that occurred at the end of annealing (in the annealing approach) or that forms a connected component of the pairwise clustering graph (in the approach via pair statistics) we reconstruct the WM as described in the previous section. We now *classify* the full data set of mini-WMs in terms of these estimated cluster WMs. Let $P(S|w_j)$ be the probability that the sequences in mini-WM S arise from sampling from the cluster WM w_j. The probability $P(w_j|S_i)$ that mini-WM i was sampled from WM w_j, as opposed to any of the other cluster WMs, is given by:

$$P(w_j|S_i) = \frac{P(S_i|w_j)P(w_j)}{\sum_k P(S_i|w_k)P(w_k)}, \qquad \textbf{[39]}$$

where the $P(w_k)$ are the prior probabilities of cluster WMs, and the sum in the denominator is over all cluster WMs (from either the set of ML WMs or the set of connected components of the connectivity graph.) With respect to the prior, we chose to use the prior that maximizes the likelihood of the full data set D given the cluster WMs. That is, choose $P(w_j)$ such that

$$P(D) = \prod_{S \in D} \left[\sum_j P(S|w_j)P(w_j) \right] \qquad \textbf{[40]}$$

is maximized, where the product is over all mini-WMs in the data set. Once we have determined the prior, we obtain the posterior probabilities of Eq. **39** and in this way obtain another membership list for each cluster, but now for the full data set. For each significant cluster these membership probabilities are also shown on the website (www.physics.rockefeller.edu/∼erik/website.html).

4.6. Finding Matches in Upstream Regions of a Genome

We can search for additional members of our clusters by scanning the upstream regions of all operons in *E. coli* for matches to the cluster WMs. For each cluster WM, and each upstream region U, we will calculate the probability that this upstream region contains *no* sites for the cluster WM. To calculate this, consider first some particular length 27 sequence s in the upstream region U. The probability that s arose from WM w is $P(s|w)$ (given by the usual expression). The probability $P(s|r)$ that this sequence arises "by chance" is $P(s|r) = \prod_i b_{s_i}$, where b_α is the background frequency of base α. Therefore, the probability that sequence s is a binding site rather than a "random" sequence is given by the posterior

$$P(w|s) = \frac{P(s|w)\pi}{P(s|w)\pi + (1-\pi)P(s|r)}, \qquad \textbf{[41]}$$

where π is the a prior probability that a particular segment of the upstream region U is a binding site for a particular TF. We will choose this prior so as to reflect the guess that, on average, an upstream region contains about 2-3 binding sites (a conservative estimate). Since there are ≈ 300 different TFs, the probability that an upstream region has a binding site for a particular TF is, on average, about 1/100. Furthermore, we would like to bias the prior such that binding sites are more likely to be found close to the translation start. We will therefore let the prior depend on the distance d from the sequence s to the translation start. That is, we take $\pi(d) = C/d$, where the constant C is chosen such that $\sum_d \pi(d) = 1/100$ over the whole upstream region. Finally, to obtain the probability that the upstream region contains *no* binding sites for the TF, we take the product over $P(w|s)$ for all ways of picking a length 27 sequence from the upstream region, which gives the probability $\bar{P}$ that the upstream region contains no site for w:

$$\bar{P} = \prod_{s \in U} P(w|s). \qquad \textbf{[42]}$$

On the website we report, for each significant cluster, the 35 operons with the highest scores $-\log[\bar{P}]$ for their upstream regions.

5. Resampling Test for Identifying Significant Clusters

In the paper, we assay the significance of our clusters within a Bayesian framework: given our set of putative sites we find clusters of sites that are more likely to cluster with each other than to cluster in any other combination with other sites. That is, we calculate how likely it is that certain sets of sites belong together relative to these sites clustering with other sites. In contrast, it is quite customary to assay the significance of a cluster by comparing some statistic that measures its "quality" against the typical quality of clusters obtained by clustering "random" sequences. That is, one assumes some null hypothesis reflecting "randomness" and calculates how *unlikely* it is for a cluster of a certain quality to arise from such random data. For instance, to test the significance of a cluster of n sequences having a WM score I that was obtained from clustering a data set of N sequences in total, one may ask: how likely is it that a set of n sequences can be found, in a set of N random sequences, that has a WM score at least as high as I?

There are several problems with such an approach. First of all, one must be very careful when interpreting such a test. Rejecting the null hypothesis of randomness means no more than: the null hypothesis does not explain the data well. What kind of other hypotheses *would* explain the data better is not addressed by such a test. That is, for our problem, one can not logically make the step: since the null hypothesis is rejected, the cluster must correspond to a "real" regulon. To make that inference one has to introduce a model that explicitly calculates the probability to obtain certain clusters of sites given that they *are* binding sites from the same regulon (as we have done).

A more technical problem is posed by the choice of null hypothesis. Often null hypotheses are obtained by randomizing the data in some way, and then re-running the algorithm with the randomized data. In such "resampling" tests, it is important that the randomization only removes correlations or biases from the data that are relevant for the inference at hand. That is, we only want to remove biases in the data that are due to the occurrence of real binding sites in the data. There may be other, irrelevant biases in the data (such as the relative frequencies of the different bases) that should *not* be removed by randomizing the data. Otherwise, the rejection of the null hypothesis may only imply that the null hypothesis does not reflect certain inherent biases in the data that have nothing to do with the occurrence of binding sites. In general, this is a hard problem; there are probably all kinds of biases in real DNA sequences that have nothing to do with the occurrence of binding sites in these sequences.

In spite of these problems, we did perform some resampling significance test for our clusters. Note also that in section 7 we evaluate the results on a test set of known binding sites, which allows us to explicitly calculate the rate of false positives and false negatives for that test set. For the resampling test, we constructed a randomized data set by permuting the columns of each of the mini-WMs in our data set independently. This retains the distribution of bases

in all the individual columns of the mini-WMs but removes any correlations in the order of these columns between different mini-WMs, which is a conservative randomization procedure.

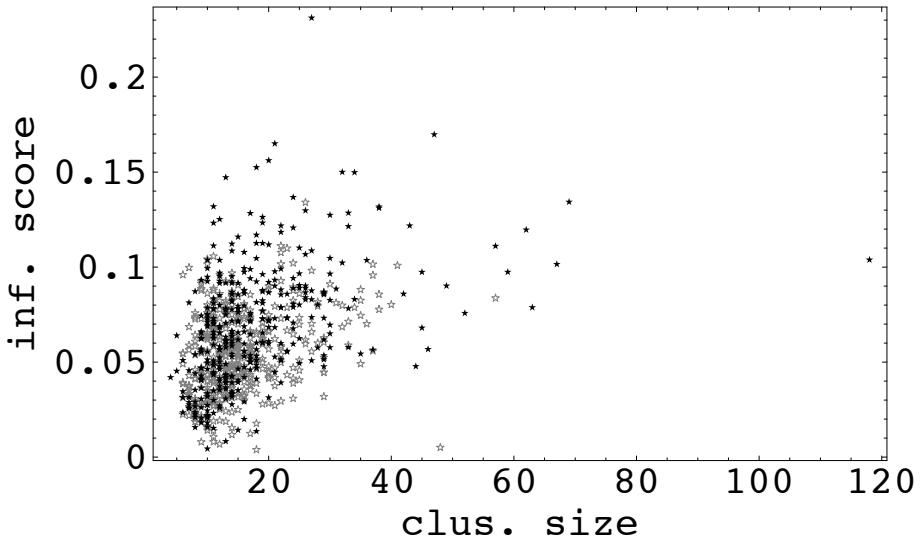

Fig. 9. Resampling test for the clusters of data set from ref. (1). Each filled black star is a cluster obtained by annealing the real data set, while each gray open star is a cluster obtained by annealing the randomized data set. Cluster sizes are shown horizontally, while their WM information scores I are shown vertically.

Fig. 9 shows the results of the resampling test on the results of data from ref. (1). The filled black stars show the clusters that were obtained at the end of annealing for the real data set. The number of sequences in each cluster is shown on the horizontal axis, while each cluster's WM score I is shown on the vertical axis. Similarly, the gray open stars show the clusters obtained by the annealing of the randomized data set. We will call a cluster significant if there is no cluster of the same size and with higher information score in the randomized data set. Of course, it would be better to perform many resampling runs and estimate more precisely the probability of annealing random data producing a cluster with information score larger than I at each possible cluster size. This is computationally rather intensive. Since we are assessing significance of clusters within a Bayesian framework, we provide the data in Fig. 9 as a guideline for comparison with significance tests that are more commonly used. Under our definition of significance for the resampling test we find 88 significant clusters, which are mostly the same clusters as deemed significant by our Bayesian procedure.

6. Operations on the Data Sets

In this section we briefly describe the data sets that were used in our studies and describe in more detail the operations that we performed on these data sets before clustering them using PROCSE.

The data set from ref. (1) contains multiple alignments of putative binding sites from a number of related proteobacteria and was collected in the following way. For each *E. coli* gene, orthologous genes were collected from the species *Actinobacillus actinomycetemcomitans*, *Haemophilus* influenzae, *Pseudomonas aeruginosa*, *Shewanella putrefaciens*, *Salmonella typhi*, *Thiobacillus ferrooxidans*, *Vibrio cholerae*, and *Yersinia pestis*. A modified version[1] of the Gibbs sampler algorithm (6) is then used to find common motifs in the upstream regions of these orthologues. The resulting alignments of putative binding sites contain between 1 and 16 sequence segments with lengths of 15-25 bases. We will interpret these alignments as collections of binding sites for the *same* TF.

The data set from ref. (2) contains similar alignments of putative binding sites for a single TF but was obtained in a somewhat different manner. Again, sets of orthologous genes were collected for each *E. coli* gene from the species *Klebsiella pneumoniae*, *S. typhi*, *V. cholerae*, and *Y. pestis*. Instead of looking for similar sequence motifs in the upstream regions, ref. (2) aligned the upstream regions and found sequence segments that are more conserved than sequences downstream of orthologous genes. The resulting alignments differ from those of ref. (1) in two respects. First, there are more stringent constraints on the selection of conserved sequence segments. Second, since the number of species used is smaller, the number of sequences in the alignment per upstream region is generally smaller.

6.1. Operations on the Data Set from (1)

Ref. (1) have annotated alignments in their data set for which the *E. coli* site overlaps either a known site, or a known repetitive element. We have removed all such alignments from our data set, since for this data set, we are interested in clustering new putative binding sites. We then ordered all the remaining alignments by their MAP value. The MAP value is a quality measure given to the alignments by ref. (1). We then went through the ordered list of alignments and removed alignments whose centers are less than 6 bases away from centers of previous alignments in the list. This is to avoid the possibility of the algorithm aligning the same mini-WM *twice* in a cluster. Overlaps are only checked for the E. coli members of the alignment. We then took the top 2,000 of the remaining alignments as our data set.

After that we symmetrically extended all alignments from their original length (between 15 and 25 bases) to length 32. To do so we had to "pad" bases to the alignments from the genomes. Some of the genomes used by ref. (1) are not yet completed and some cases occurred in which the bases could not

[1] One of the pertinent modifications is the favoring of sites that show palindromicity.

be padded because the sequence is not available. In those rare cases, we added random bases to replace the missing bases. As pointed out in the paper, we would like to consider the sequences in the resulting alignment as *independent* samples from a WM describing the TF. However, some of the species are probably too closely related evolutionary to warrant this assumption of independence. That is, some of the bases in the alignment are conserved not because of selectional constraints but because they simply haven't mutated yet since the species diverged. Ideally we would deal with these evolutionary relationships directly, reconstructing the phylogenetic tree of the species involved, and inferring which bases in the alignment have been retained by selection. However, this is an altogether separate inference problem and we opted for a simple circumvention of this problem. We replaced the sequences from the most closely related species by their consensus. That is, we replaced triplets of sequences from *E. coli*, *S. typhi*, and *Y. pestis* with their consensus, and replaced sequences from the pair *H. influenzae* and *A. actinomycetemcomitans* with their consensus as well. We deem all other species to be sufficiently separated evolutionary such that we can assume their sequences to be independent samples of a WM. There are two more technical details to mention regarding this procedure. First, there may not be a majority base among the sequences. In those cases we select one of the bases among the triplet (or pair) of sequences at random. The second problem arises because the alignments of ref. (1) may contain more than one sequence segment from the same upstream region, i.e. from the same genome. Thus, if we have two segments from *S. typhi* and two from *E. coli* occurring in the same alignment, we have to decide how to pair these before taking their consensus. We do this by picking the pairing that maximizes the number of conserved bases between the paired segments. These operations now give us the mini-WMs on which the PROCSE algorithm operates.

6.2. Operation on the data set from (2)

The operations on the data set from ref. (2) are largely the same as those described in the previous section. We used the set of predicted *E. coli* binding sites from (2) and the corresponding local alignments to construct a mini-WM of length 32 for each putative binding site x. In case the length $l(x)$ was < 32, we padded additional bases to the left and to the right using the most significantly conserved *E. coli* base b as an anchor. Similarly, if $l(x) > 32$, sites were clipped off. The sequences for *E. coli*, *K. pneumoniae*, and *S. typhi* were replaced by their consensus. Each mini-WM was scored by the significance score for b (2) and ordered accordingly. Alignments that overlapped an alignment with higher score with 27 or more bases were removed. In contrast to the algorithm employed in (1), the selection of sites in this data set was not optimized to maximize the inclusion of known binding sites or to favor reverse complement symmetry of the sites. For this reason, we did not remove sites that overlap known binding sites (or known repetitive elements) from this set. By comparing the results obtained with this data set and those obtained with the data set from (1), we may (to some extent) assess the effect on the discovery of new vs. known regulons of the

removal/inclusion of known sites.

6.3. Processing of the data set from (3)

The data set of known TF binding sites from Church's lab (3) contains a total of 997 sites. In order to create a test set for our algorithm we filtered this set in several ways. First we removed all σ-factor binding sites. There are several reasons for removing these sites. First of all, σ-70 sites are ubiquitous in the genome, whereas we are interested in finding regulons (sites and their TFs) that specifically regulate (relatively) small sets of genes. Second, there are regulatory sites that overlap σ-sites, and since in our algorithm any piece of genome can only occur in one cluster at a time, these sites would alternatively cluster with the regulon of interest and with the rest of the σ-sites at other times. The presence of the σ-sites would thus negatively effect the inference of other (smaller) regulons. Finally, the primary data set (1) looked for reverse complement symmetry in the binding sites, thus disfavoring the inclusion of σ-sites in their data set. Thus, removing the σ-sites from the tes -set makes it also more comparable with the data set from ref. (1).

After removing σ-sites and sites that overlap each other by more than 27 bases there are 397 unique TF binding sites in this data set.

6.4. Preparation of TF mini-WMs from Data Set (3)

When we cluster the new putative binding sites we want to separate clusters of sites that correspond to TFs for which some binding sites are already known from clusters of sites that form entirely new putative regulons. To this end we took all the known binding sites from the collection from (3) and divided them into groups according to their annotation. That is, all sites that are annotated as binding sites for the same TF form a group. We then aligned these groups of sites into alignments of all known sites for each TF. The multiple alignment is performed by running a hierarchical clustering with our scoring function. That is, we first put all known sites for a TF into separate clusters and then find the pair that, when fused, leads to the largest increase in the score $P(D|C)$ of the partition. We repeat this procedure (i.e. at the second step either a third sequence is added to the pair or another pair is fused) until a single cluster is left. At each step the alignment is maximized over all possible ways of shifting the sites with respect to each other and both strands. Thus, we sometimes have to pad bases from the genome to "complete" the alignment. At the end, the length 32 window with the highest information score is taken from the alignment. The mini-WM so obtained represents all known sites for a particular TF. We constructed 56 different mini-WMs in this way from data set (3). There are only 53 TFs represented in this data set, but three of them (metJ, argR, and phoB) have two different types of sites, which we aligned separately.

We added these 56 mini-WMs to both data sets. The idea is that if additional binding sites for any one of these TFs occur in our data set, then these sites will tend to cluster with one of the 56 mini-WMs that we have just described.

When we collect the significant clusters at the end of the run, we separate them into two sets: those that contain at least one of the 56 mini-WMs constructed from (3) and those that do not. The former clusters contain new putative sites for known regulons, while the latter contain new predicted regulons.

7. Results of Clustering Known Binding Sites

Table 2 shows results from clustering the set of 397 sites from ref. (3). We performed several different tests of our algorithm with this data. First of all, one

fac.	size	int.	size	num. site	int.	size	num. site	int.
arcA	12	2-5	20/15	3/3	1-11/1-11	24	4	1-14
argR2	7	5-7	21/24	7/7	8-20/11-21	26	5	1-11
argR	17	8-14	22/22	10/12	12-21/11-21	-	-	-
crp	41	13-27	22/29	19/20	2-16/7-21	39	19	7-28
dnaA	8	3-6	7/8	6/6	3-7/3-7	-	-	-
fadR	7	4-7	12/9	7/7	3-10/3-8	11	4	1-8
fis	11	2-3	20/15	4/4	1-11/1-11	-	-	-
fnr	12	7-9	11/11	9/9	7-10/7-10	25	6	17-23
fruR	11	8-11	17/13	11/10	8-13/7-12	15	7	1-6
fur	8	5-6	21/24	5/6	8-20/11-21	28	6	19-24
galR	6	3-6	6/14	5/6	3-6/3-8	9	2	1-4
hipB	4	3-4	8/14	3/4	2-5/4-13	-	-	-
ihf	18	3-5	22/22	5/4	12-21/11-21	27	5	5-15
lexA	19	18-19	24/24	19/19	19-23/19-23	28	18	19-24
lrp	14	2-4	-/-	-/-	-/-	-	-	-
malT	10	2-6	9/8	7/6	1-6/1-6	-	-	-
metJ(3)	15	13-15	17/15	14/13	14-17/13-15	17	14	13-16
narL	9	2-6	10/14	5/6	1-5/4-13	-	-	-
ntrC	5	5-5	9/10	5/5	5-7/5-8	18	14	9-14
phoB	9	2-4	9/13	3/4	1-7/1-8	11	4	1-8
purR	16	13-15	15/17	14/16	13-15/13-16	23	13	16-20
trpR	3	2-3	8/-	3/-	1-5/-	-	-	-
tus	5	4-5	11/8	5/5	5-8/5-7	12	5	3-6
tyrR	16	5-11	15/21	9/8	2-10/1-13	-	-	-

Table 2. Clustering of the 397 sites from ref. (3). Column 1 shows the TF name, column 2 shows the number of sites in the data set annotated as binding sites for that TF (according to ref. (3)), column 3 shows the the 90% probability interval for the number of coclustering sites for this TF. Only those TFs are shown for which the lower bound of the 90% probability interval is 2 or higher. We performed two annealing runs of this data set to find the ML partition. For each TF, we found the cluster in this partition with the largest number of sites for that TF. The clusters shown in columns 4-6 resulted from two simulated annealing runs, with the "size" and "int" defined as before. Column 5 shows how many of the sites in column 4 were annotated for the TF. For example, columns 4-6 for the TF dnaA show that at the end of the two annealing runs there were clusters of sizes 7 and 8, respectively that both contained 6 of the 8 known dnaA binding sites, and with 90% probability between 3 and 7 members of these clusters co-cluster. Finally, columns 7-9 show the same statistics when the annealing is done on a larger data set that contains both the 397 sites of ref. (3) as well as the top $2,000$ E. coli sites from ref. (1).

may test how well the known binding sites cluster according to their annotation. That is, do sites annotated to be binding sites for the same TF X significantly cluster with each other? To answer this question, we performed Monte Carlo sampling of the probability distribution $P(C|D)$ for this data set. At each time step of the Monte Carlo random walk, we measure how many sites of each TF "cocluster". This is defined as follows: At any particular time step of the Monte Carlo walk, the sites that are annotated for TF X will be distributed in some way among the clusters that are present at that time during the random walk. For instance, if TF X has 10 binding sites in the data, 6 of those may sit in one cluster, 2 others may sit in another cluster together with some sites of different annotation, and the last 2 may each sit in some cluster that is dominated by sites of other annotation. For such a partition, we would say that 6 of the 10 sites for TF X cocluster. By calculating how many sites cocluster at each time step, we measure, for each TF, the distribution $p(k)$ that k of its sites will cocluster at any point in time. These distributions may be summarized in various ways, but we chose to find the shortest length interval $[k_{\min}, k_{\max}]$ such that at least 90% of the time between $k_{\min}$ and $k_{\max}$ sites cocluster, i.e. $\sum_{k=k_{\min}}^{k_{\max}} p(k) > 0.9$. We will consider the sites of a TF X to cluster "significantly" if $k_{\min} > 1$ for its distribution $p(k)$. Of the 53 TFs represented in the data set, there are 24 that cluster significantly. These are shown in the rows of Table 2. The first column shows the name of the TF, the second column shows the total number of sites for that TF contained in the 397-site data set, and the third column shows the interval $[k_{\min}, k_{\max}]$. The fifth row, for instance, shows that the TF dnaA has a total of 8 sites in the data set, and that at least 90% of the time between 3 and 6 of these sites cocluster.

As mentioned before, 24 of the 53 TFs cluster significantly. One might call this an overall "false negative" rate of 55%. All the TFs that cluster significantly (with the exception of trpR) have more than three binding sites in the data set. There are 22 TFs that have 3 or less sites in the data. Therefore, it is clear that it is mostly TFs with a small number of samples in the data that do not cluster significantly. In other words, there is a false negative rate of less than 23% for TFs with more than three sites in the data set. Table 3 shows the TFs whose sites did not cluster significantly. Note that some of these sites (such as cpxR, flhCD, glpR, metR, modE, nagC, ompR) *do* cluster significantly better than sets of random sequences would, but did not pass our significance threshold[2]. That is, we could have changed the significance thresholds such that *for this particular data set* the number of false negatives would be further decreased. Still, it is quite clear that the sites for a TF like soxS do not cluster at all. The variability among the 8 soxS sites is so large that one might wonder if a WM model is appropriate at all for TFs like this.

[2]One would assume that for random data all partitions are equally likely. One can then ask: what is the expected co-clustering distribution $p(k)$ for some particular set of n sites, and what is the sampling distribution of, for instance, the mean size $\langle k \rangle = \sum_k kp(k)$. Although cumbersome, these quantities can be calculated analytically, and one would find that for $n = 6$, $\langle k \rangle = 1.39$ and that $\langle k \rangle = 2$ constitutes a significant clustering of the 6 sites. By such measures, some of the clusters in table 3 would come out significant.

In this first test, we used the annotation of the sites to measure to what extent sites with equal annotation cluster together. Summarizing, we found significant clustering for slightly less than half of the TFs, and for over 75% of the TFs that have more than 3 sites in the data set. After this, we wanted to test to what extent our annealing approach will be able to identify the "correct" clusters. That is, we wanted to compare the ML clusters that result from annealing with the annotation of the sites. We performed two annealing runs to identify a ML partition. We tested the significance of the ML clusters obtained by annealing by performing two sampling runs. Again we measure the coclustering distribution $p(k)$ for each cluster and consider a cluster significant if the 90% probability interval $[k_{\min}, k_{\max}]$ has $k_{\min} > 1$. In order to compare these

fac.	size	int.
ada	2	1-1
araC	4	1-2
carP	2	1-1
cpxR	6	1-3
cspA	4	1-2
cynR	2	1-1
cysB	3	1-1
cytR	5	1-2
deoR	1	1-1
farR	3	1-2
fhlA	3	1-1
flhCD	3	1-3
gcvA	4	1-2
glpR	7	1-3
hns	3	1-1
hu	1	1-1
iclR	1	1-1
ilvY	2	1-1
lacI	1	1-1
marR	2	1-2
metR	7	1-4
modE	3	1-3
nagC	6	1-3
ompR	8	1-3
oxyR	3	1-2
pdhR	2	1-1
rhaS	2	1-1
soxS	8	1-2
torR	1	1-3

Table 3. TFs from data set (3) whose sites did not cluster significantly. First column is TF name, second is the total number of binding sites for that TF in the data, and the third column is the 90% probability interval $[k_{\min}, k_{\max}]$ of its coclustering distribution $p(k)$.

clusters with the annotation, we had to identify, for each TF, which of the ML clusters "corresponds" best to the cluster of binding sites of the TF. We do this by simply finding the ML cluster that contains the largest number of sites for the TF. That is, the 10 sites of TF X may be spread over 4 clusters in the ML partition, with 5 sites in one cluster, 3 in another, and 2 more in a third and fourth cluster. We would then identify the ML cluster containing the 5 sites as the cluster "corresponding" to TF X.

These clusters are shown in columns 4 through 6 of table 2. The column entries are best explained by example. Focus, for instance, on the line for TF narL. In the first annealing run, there was a ML cluster that contained 5 of the 9 narL sites. This cluster had a total size of 10 sequences (that is, apart from the 5 narL members, 5 members had different annotations). Under sampling, between 1 and 5 sites of this cluster cocluster 90% of the time. (Note that this means: between 1 and 5 of the full set of 10 sites.) In the second annealing run, there was a ML cluster of size 14 that contains 6 of the 9 narL sites. Between 4 and 13 members of this cluster cocluster 90% of the time. To give an example of a TF for which essentially the same cluster was found in both annealing runs, focus on the lexA line. In both runs there was a cluster of size 24 that contained all 19 of the known lexA sites (and 5 sites with a different annotation). During sampling, between 19 and 23 members of this cluster cocluster (being one of the most stable clusters in the set).

In general, one can see that there is good agreement between the annotation and the clusters inferred by annealing. Looking in more detail at some of the annealed clusters, we note that narL and narP sites cluster together, as do fur and argR2 sites. The ihf sequences sit mostly in the tail of the argR cluster and the fis sequences occupy the tail of the arcA cluster. The many crp sites sit distributed over two clusters (with only the largest represented in the table).

This test also allows us to assess the amount of *false positives* that our algorithm produces. At the end of the annealing there were 30 and 29 clusters in the two respective runs. After sampling, it turns out that 20 of the 30 we deemed significant using our cut-offs, and 16 of the 29 in the other run. None of these clusters are false positives. That is, our false positive rate is 0 for this particular test set.

Finally, one may think that the annealing had not really found the ML clustering (or something close to that), but had gotten stuck in some local optimum, and that if we had had more time, and cooled slower, we may have found clusterings with better score that correspond even better to the known annotation. This is unlikely. One can easily calculate what the probability $P(C|D)$ is of the partition in which all sites are clustered according to their annotation. We find that this probability is substantially *lower* than the probability of the partitions that annealing finds. We believe that this has two main reasons. First, we believe that some of the sites are mis-annotated. Some sites of annotation X consistently cluster with sites of some other annotation Y. Second, for some TFs (such as lrp, fis, ihf and soxS mentioned above) it seems that the WM representation of the sites is unsatisfactory. Their WM scores are extremely low: when we, for instance, run the clustering algorithm on *only* the fis sites, the algorithm prefers partitions in which these sites are not partitioned into a single cluster at all! Possibly, these TFs recognize something else than a sequence motif that can be represented by a WM.

References

1. McCue, L. A., Thompson, W., Carmack, C. S., Ryan, M. P., Liu, J. S., Derbyshire, V., & Lawrence, C. E. (2001) *Nucleic Acids Res.* **29**, 774–782.

2. Rajewsky, N., Socci, N. D., Zapotocky, M., & Siggia, E. D. (2002) *Genome Res.* **12**, 298–308.

3. Robison, K., McGuire, A. M., & Church, G. M. (1998) *J. Mol. Biol.* **284**, 241–254

4. Berg, O. G. & von Hippel, P. H. (1987) *J. Mol. Biol.* **193**, 723–750.

5. Jaynes, E. T. (1983) *Papers on probability, statistics, and statistical physics* vol. 158 of *Synthese library* (D. Reidel publishing company, Dordrecht Holland) R.D. Rosenkrantz, editor.

6. Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., & Wootton, J. C. (1993) *Science* **262**, 208–214.