20.109 Transcription Factor Motifs

Amanda Kedaigle, Ernest Fraenkel

1/12/2018

Transcription Factor Binding Sites

Regulatory regions in eukaryotic DNA are marked by transcription factor binding sites (TFBSs), or motifs, which are sequence patterns with various degree of degeneracy. That is, certain short sequences will bind specific transcription factors - and the sequences can change somewhat and still make a good binding site.

The Bioconductor package TFBSTools package is designed to be a computational framework for TFBSs analysis [1]. First, we'll look at ways that it stores patterns for known TFBSs. Previous work has identified these patterns for hundreds of TFs. Second, a set of DNA sequences are analyzed to determine the locations of sequences that fit the described binding pattern.

Position Frequency Matrices and other representations

TFBSTools gives us several classes as ways to represent the sequences that define TFBSs. The class PFMatrix is designed to store all the relevant information for one raw position frequency matrix (PFM).

The following examples demonstrate the creation of a PFMatrix, the conversions between these matrices and some assocated methods defined for these classes.

```
library(TFBSTools)

## PFMatrix construction
pfm = PFMatrix(ID="MA0004.1", name="Arnt",
               bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
               tags=list(family="Helix-Loop-Helix", species="10090",
                         tax_group="vertebrates"),
               profileMatrix=matrix(c(4L,  19L, 0L,  0L,  0L,  0L,
                                      16L, 0L,  20L, 0L,  0L,  0L,
                                      0L,  1L,  0L,  20L, 0L,  20L,
                                      0L,  0L,  0L,  0L,  20L, 0L),
                                    byrow=TRUE, nrow=4,
                                    dimnames=list(c("A", "C", "G", "T"))
                                    )
               )

pfm
```

Describe what each of the arguments to PFMatrix is telling you about this TFBD sequence, especially the entries in the matrix?

The position frequency matrix counts the amount of time each nucleotide appeared in each motif position in a set of sequences. This one was created from a set of 20 sequences.

As opposed to position frequency matrix, a position weight matrix (PWM) holds the probability of finding each nucleotide in each position, rather than the counts. These are stored as log likelihoods, based on the background frequencies of each nucleotide. The method toPWM can convert PFMs to PWMs. PWMs are also called position-specific scoring matrices, or PSSMs.

[1] Some of these instructions are based on the TFBSTools package vignette by Ge Tan.

The method toICM can convert PFM to ICM, or information content matrix. The information content matrix has a column sum between 0 (no nucleotide preference) and 2 (only 1 nucleotide used).

```
## convert a FPM to PWM, ICM
pwm = toPWM(pfm)
pwm

icm = toICM(pfm)
icm

seqLogo(icm)
```

Check out the sequence logo created for this motif. Below is the information for another real TFBS, for the transcription factor Barx1. Draw roughly what you'd expect the logo to look like.

```
pfm = PFMatrix(ID="MA0875.1", name="Barx1",
               bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
               tags=list(family="NK-related factors", species="9606",
                         tax_group="vertebrates"),
               profileMatrix=matrix(c(1872L,  590L, 3339L,  6805L,  0L,  0L, 6805L,
                                      1821L, 5138L,  1992L, 207L,  0L,  0L, 0L,
                                      2236L,  246L,  1386L,  192L, 0L,  0L, 0L,
                                      877L,  1667L,  87L,  0L,  6805L, 6805L, 0L),
                                    byrow=TRUE, nrow=4,
                                    dimnames=list(c("A", "C", "G", "T"))
                                    )
               )
```

The JASPAR database

This section will demonstrate how to operate on the JASPAR 2016 database. JASPAR is a collection of transcription factor DNA-binding preferences, modeled as matrices. These can be converted into PWMs, used for scanning genomic sequences. JASPAR is the only database with this scope where the data can be used with no restrictions (open-source). A Bioconductor experiment data package is provided with each release of JASPAR.

This search function fetches matrix data for all matrices in the database matching criteria defined by the named arguments and returns a PFMatrixList object.

```
library(JASPAR2016)
opts = list()
opts[["species"]] = 9606 #This is the ID for Homo Sapiens
opts[["name"]] = "RUNX1"
opts[["type"]] = "SELEX"
opts[["all_versions"]] = TRUE
PFMatrixList = getMatrixSet(JASPAR2016, opts)
PFMatrixList
```

Scan sequences with PWM pattern

Once we have a list of sequences, we want to scan DNA sequences to look for matching TFBSs. The SiteSet class is a container for storing a set of potential binding sites on a nucleotide sequence (start, end, strand, score, pattern as a PWMatrix, etc.) from scanning a nucleotide sequence with the corresponding PWMatrix.

searchSeq scans a nucleotide sequence with the pattern represented in the PWM. The strand argument controls which DNA strand of the sequence will be searched. When it is _*_, both strands will be scanned.

```
library(Biostrings)
subject = DNAString("GAATTCTCTCTTGTTGTAGTCTCTTGACAAAATG")
siteset = searchSeq(pwm, subject, seqname="seq1", min.score="60%", strand="*")
(siteset)
```

How many matches for this TF did we find in this sequence? Which is the best match?

de novo Motif Discovery

So far, we've talked about ways to scan sequences for motifs when we already have a good idea what that motif looks like. Computational biology, however, is also concerned with ways to discover motifs, or highly conserved, repeated sequences, from scratch. This is called "de novo" motif discovery.

In this section we'll implement a small example of the algorithm that a popular de novo motif discovery tool called MEME uses. This algorithm is an example of an Expectation/Maximization optimization algorithm. These algorithms are useful when you want to learn the best parameters for your problem - in our case, what the motif looks like - *and* the optimal answer to your problem - here, where in the sequences the motif lies. The EM algorithm alternates between performing an expectation (E) step, which estimates the best guess of motif locations using the current estimate for the motif, and a maximization (M) step, which computes an optimal motif based on our current gues about locations. The algorithm goes back and forth between these two steps until it converges on an optimal set of motif parameters and locations.

We'll run through just one round of the EM algorithm here. Let's start with three short sequences that we want to find a motif in. Each location in the sequences is called "l1","l2", etc:

Table 1: Sequences

	L1	L2	L3	L4	L5
X1:	A	C	A	G	C
X2:	A	G	C	A	G
X3:	T	C	A	G	T

And we'll assume that in our organism, there is equal chance of each of the nucleotides appearing (the "background probabilites" of each nucleotide is equal).

Table 2: Background

	BG
A	0.25
C	0.25
G	0.25
T	0.25

To start an EM algorithm, we need a starting point for the parameters; a first guess at what the motif might look like. In this case, we want a position probability matrix. Each entry in the matrix gives the probability of that nucleotide being in that position of the motif.

Table 3: M

	M1	M2	M3
A	0.1	0.5	0.2
C	0.3	0.2	0.1
G	0.3	0.1	0.4
T	0.3	0.2	0.3

E Step

To estimate the locations of the motif in the sequence, let's make a matrix Z, such that the entries $Z_{i,j}$ (the number in row i and column j of Z) is the probability of our motif M starting at position j in sequence Xi. To calculate that probability, assume the rest of sequence Xi is background. Fill in the following equations to fill in the matrix. (Tip! Remember you can copy and paste equations into the R console to use it like a calculator.)

$Z'_{1,1} = M_{A,1} * M_{C,2} * M_{A,3} * BG_G * BG_C = 0.1 * 0.2 * 0.2 * 0.25 * 0.25 = 0.00025$

$Z'_{1,2} =?*?*?*?*? = 0.25 * 0.3 * 0.5 * 0.4 * 0.25 = 0.00375$

$Z'_{1,3} =$

$Z'_{2,1} =$

$Z'_{2,2} =$

$Z'_{2,3} =$

$Z'_{3,1} =$

$Z'_{3,2} =$

$Z'_{3,2} =$

Table 4: Z prime

	Z1	Z2	Z3
X1	0.00025	0.00375	
X2			
X3			

Now, we want Z to properly represent probabilities, so we normalize Z' so that, for each sequence (row), the sum of probabilities of the motif starting at each position in that sequence is equal to one. Do this by dividing each entry in Z' by the sum of that row.

```
Z = Zprime/rowSums(Zprime)
kable(Z, caption = 'Z')
```

M Step

Now, we re-estimate M assuming that Z is correct. In the matrix M, $M_{k,l}$ is the probability that nucleotide k at position l, in the motif. We can estimate that by summing the probabilities from Z that refer to incidences of the motif with nucleotide at position l of the sequence.

$M'_{A,1} = Z_{1,1} + Z_{1,3} + Z_{2,1} + Z_{3,3} = 0.0615385 + 0.0153846 + 0.0149254 + 0.04 = 0.1318485$

$M'_{A,2}$ =?+?+? $= 0.9230769 + 0.8855224 + 0.8 = 2.608599$

$M'_{A,3} =$

$M'_{C,1} =$

$M'_{C,2} =$

$M'_{C,3} =$

$M'_{G,1} =$

$M'_{G,2} =$

$M'_{G,3} =$

$M'_{T,1} =$

$M'_{T,2} = None = 0$

$M'_{T,3} =$

Table 5: M prime

	M1	M2	M3
A	0.1318485	2.608599	
C			
G			
T		0	

To make M properly represent probabilities, normalize M' so that for each position in the motif (column), the sum of the probabilities of each nucleotide is equal to one. Do this by dividing each entry in M' by the sum of that column.

```
M = Mprime/rowSums(Mprime)
kable(M, caption = 'M')
```

After one round of EM, what is the most likely motif, and what are the starting positions of it in each sequence?