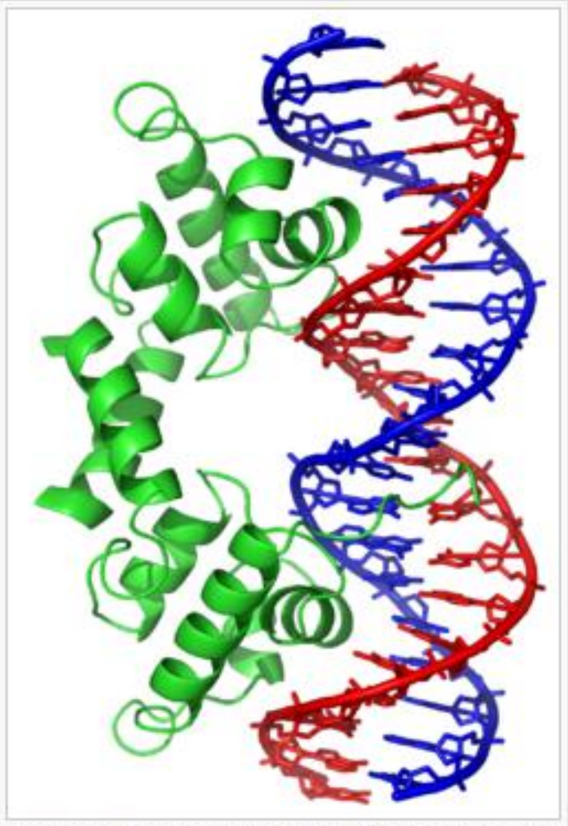
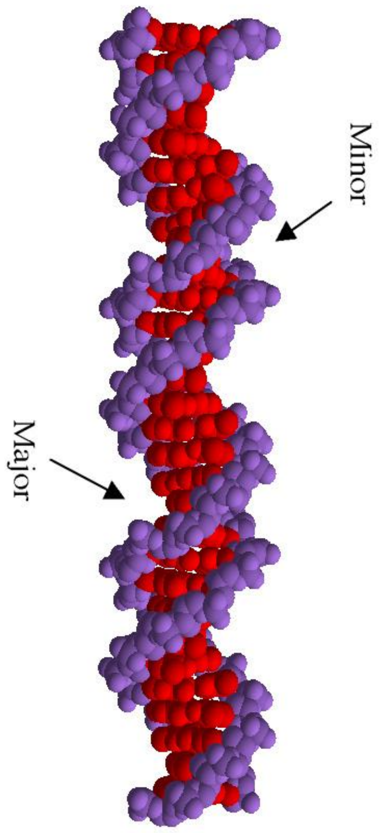
Chapter 9 Class Notes – Signals in DNA

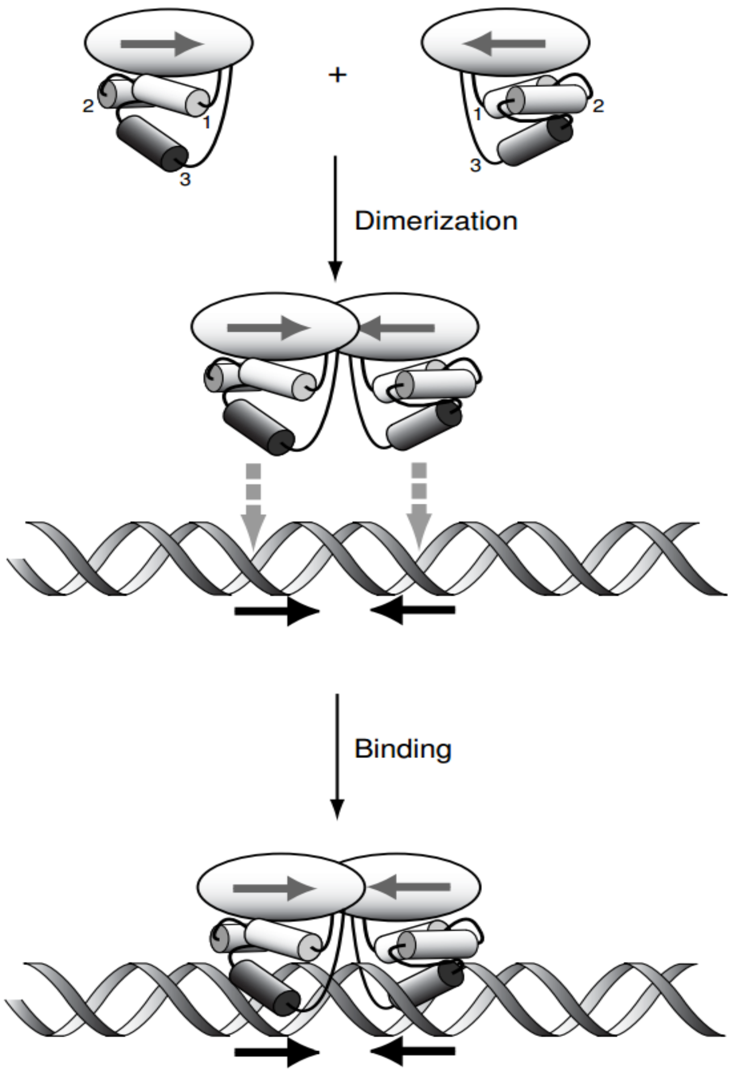
9.1. The Biological Problem: since proteins cannot "read", how do they recognize nucleotides such as `A`, `C`, `G`, `T`? Although only approximate, proteins actually recognize specific atoms or groups of atoms on bases or base pairs; examples:

- **Restriction endonuclease recognition sequences**
- **Binding sites for regulatory proteins**
- **Elements within replication origins and termination regions in genomes**
- **Promoters**

Proteins interacting with DNA: (see link http://en.wikipedia.org/wiki/Protein%E2%80%93DNA_interaction)


The lambda repressor protein interacting with the lambda operator DNA.





Above image uses bacteriophage lambda cro protein as an example to illustrate the relationship between binding protein and binding site structure.

The sites recognized by cro protein are summarized as follows in the table below (taken from text, p.231). Note that the sequences of the six operator sites are similar but not identical.

O_L1	5′-T A T C A C C G $\underline{C}$ C A G T G G T A-3′
	3′-A T A G T G G C $\underline{G}$ G T C A C C A T-5′
O_R1	5′-T A T C A C C G $\underline{C}$ C A G A G G T A-3′
	3′-A T A G T G G C $\underline{G}$ G T C T C C A T-5′
O_L2	5′-T A T C T C T G $\underline{G}$ C G G T G T T G-3′
	3′-A T A G A G A C $\underline{C}$ G C C A C A A C-5′
O_R2	5′-T A A C A C C G $\underline{T}$ G C G T G T T G-3′
	3′-A T T G T G G C $\underline{A}$ C G C A C A A C-5′
O_L3	5′-T A T C A C C G $\underline{C}$ A G A T G G T T-3′
	3′-A T A G T G G C $\underline{G}$ T C T A C C A A-5′
O_R3	5′-T A T C A C C G $\underline{C}$ A A G G G A T A-3′
	3′-A T A G T G G C $\underline{G}$ T T C C C T A T-5′

Identifying Signals in Nucleic Acid Sequences – one approach is the unsupervised approach: an example of unsupervised pattern discovery is identifying k-words that are over-represented in a set of functional sequences such as promoters (we did this back in 3.6, p.89). Another approach, supervised learning, includes prior knowledge of the pattern to be found (our focus in this chapter).

Context example: the MERS (Middle East Respiratory Syndrome, 2012), is a coronavirus, akin to SARS (Severe Acute Respiratory Syndrome, 2003). We want to sequence it, to find the proteins (BLAST), and ultimately to make an antiviral drug.

9.2. Representing Signals in DNA: Independent Positions

– the simplest way to represent signals in DNA (specifically the binding site) is as a consensus sequence: a string of characters corresponding to the most common occurrences of bases at each position. One example (glucocorticoid receptor element,

GRE) follows. The breakdown of the percentages of the 4 nucleotides at each position provides an idea of the amount of agreement or variability at the specific position.

$$\begin{array}{ll} \text{Position:} & \texttt{123456} \\ & \texttt{AGAACA} \\ & \texttt{A}\underline{\texttt{C}}\texttt{AACA} \\ & \texttt{AGAACA} \\ & \texttt{AGAA}\underline{\texttt{G}}\texttt{A} \\ & \texttt{AGAACA} \\ & \texttt{AGAAC}\underline{\texttt{T}} \\ & \texttt{AGAACA} \\ \cline{2-2} \text{Consensus:} & \texttt{AGAACA} \end{array}$$

PWM: Positional Weight Matrix

$$\begin{bmatrix} 1.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.86 \\ 0.00 & 0.14 & 0.00 & 0.00 & 0.86 & 0.00 \\ 0.00 & 0.86 & 0.00 & 0.00 & 0.14 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.14 \end{bmatrix}$$

So, we are transitioning away from the "iid" (independent and identically distributed) model – we are removing the "id", and so the different nucleotide positions do not have to have the identical distribution here.

The needed steps here are first to gather together and align representatives of the signal. Then, a training set (collection of bona fide signals or sites) is used to produce the probability model. A second bona fide set of signals/sites – called the validation set – is needed for testing the model. Sometimes the construction of the model requires a challenge set of sequences (the set to which the real sequences in the training set are contrasted). "The process of estimating the probability distributions from the training set is called learning in the machine-learning world."

9.2.1. Probabilistic Framework: our goal is to recognize a signal of length w within a string representing a DNA sequence; to do so, we parse the sequence using windows of width w and ask how well the w letters in that window correspond to a particular signal. This means we must assign a score to the sequence in each window. "We assume that we have a collection of aligned DNA sequences of the same length w and having no gaps, and that we know that all members of this collection are sites for binding a particular DNA-binding protein." We set the scores in analogous manner to as we did previously:

- **the probability of a sequence $A = a_1 a_2 \dots a_w$ given hypothesis $\mathcal{B}$ that it is a binding site is**

$$P(A|\mathcal{B}) = \prod_{i=1}^{w} p_{a_i} = p_{a_1} \times p_{a_2} \times \dots \times p_{a_w}$$

- **If the same sequence is chosen from the random iid model $\mathfrak{R}$ with associated probabilities q_{a_i}, then:**

$$P(A|\mathfrak{R}) = \prod_{i=1}^{w} q_{a_i} = q_{a_1} \times q_{a_2} \times \dots \times q_{a_w}$$

- **A comparison of these probabilities is the ratio:**

$$\frac{P(A|\mathcal{B})}{P(A|\mathfrak{R})} = \prod_{i=1}^{w} \frac{p_{a_i}}{q_{a_i}}$$

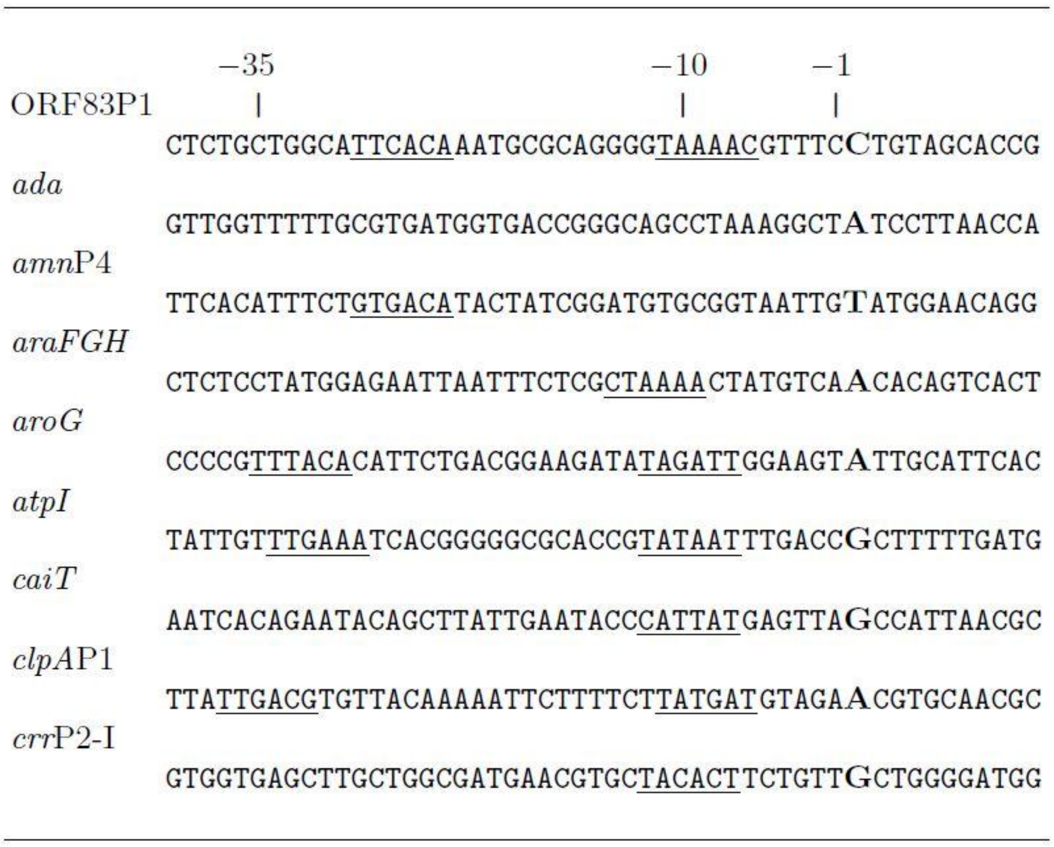
- **A score for the sequence is the log of this ratio:**

$$S = \log_2 \frac{P(A|\mathcal{B})}{P(A|\mathfrak{R})} = \sum_{i=1}^{w} \log_2\left(p_{a_i}/q_{a_i}\right) = \sum_{i=1}^{w} s_i$$

In this sum, $s_i = \log_2\left(p_{a_i}/q_{a_i}\right)$ is the contribution to the sequence score of the base at position i. In the case of 50% G+C, each $q_a = 0.25$ for a in {A,C,G,T}.

For the Escherichia coli promoter data from Appendix C.3 (p.509) in the region of -10 (where it is thought the consensus is TATAAT), we obtain (for just 9 of the many examples):

Table 9.2. A sample of *E. coli* promoter sequences. These sequences have been aligned relative to the transcriptional start site at position +1 (boldface large letter). Sequences from −40 to +11 are shown. Close matches to consensus −35 and −10 hexamers are underlined. See also Appendix C.3 for additional examples and sources of the data.



```
                   −35                         −10       −1
ORF83P1             |                           |         |
           CTCTGCTGGCATTCACAAATGCGCAGGGGTAAAACGTTTCCTGTAGCACCG
ada
           GTTGGTTTTTGCGTGATGGTGACCGGGCAGCCTAAAGGCTATCCTTAACCA
amnP4
           TTCACATTTCTGTGACATACTATCGGATGTGCGGTAATTGTATGGAACAGG
araFGH
           CTCTCCTATGGAGAATTAATTTCTCGCTAAAACTATGTCAACACAGTCACT
aroG
           CCCCGTTTACACATTCTGACGGAAGATATAGATTGGAAGTATTGCATTCAC
atpI
           TATTGTTTGAAATCACGGGGGCGCACCGTATAATTTGACCGCTTTTTGATG
caiT
           AATCACAGAATACAGCTTATTGAATACCCATTATGAGTTAGCCATTAACGC
clpAP1
           TTATTGACGTGTTACAAAAATTCTTTTCTTATGATGTAGAACGTGCAACGC
crrP2-I
           GTGGTGAGCTTGCTGGCGATGAACGTGCTACACTTCTGTTGCTGGGGATGG
```

Clearly, it is tough to pick out the -10 hexamer, so we do so by using the corresponding positional weight matrix:

Frequencies:	**Positional weight matrix (PWM)**
$\begin{bmatrix} 9 & 214 & 63 & 142 & 118 & 8 \\ 22 & 7 & 26 & 31 & 52 & 13 \\ 18 & 2 & 29 & 38 & 28 & 5 \\ 193 & 19 & 124 & 31 & 43 & 216 \end{bmatrix}$	$\begin{bmatrix} 0.04 & 0.88 & 0.26 & 0.59 & 0.49 & 0.03 \\ 0.09 & 0.03 & 0.11 & 0.13 & 0.21 & 0.05 \\ 0.07 & 0.01 & 0.12 & 0.16 & 0.12 & 0.02 \\ 0.80 & 0.08 & 0.51 & 0.13 & 0.18 & 0.89 \end{bmatrix}$

Position-specific scoring matrix (PSSM)

$$\begin{bmatrix} -2.75 & \boxed{1.82} & 0.06 & \boxed{1.23} & \boxed{0.96} & -2.92 \\ -1.46 & -3.11 & -1.22 & -0.96 & -0.22 & -2.22 \\ -1.75 & -4.92 & -1.06 & -0.67 & -1.11 & -3.60 \\ \boxed{1.67} & -1.67 & \boxed{1.04} & -0.96 & -0.49 & \boxed{1.84} \end{bmatrix}$$

(contains scores: $s_i = \log_2\left(p_{a_i}/q_{a_i}\right)$ using all $q_{a_i} = 0.25$)

From bottom of p.5, it is clear that $\frac{P(A|\mathcal{B})}{P(A|\mathfrak{R})} = 2^S$. To illustrate, consider the sequence ACTATAATCG, which (with $w = 6$) we parse into ACTATA, CTATAA, TATAAT, ATAATC, TAATCG; the individuals scores associated with TATAAT are boxed above, and lead to $S = 1.67 + 1.82 + \cdots + 1.84 = 8.56$ and $\frac{P(A|\mathcal{B})}{P(A|\mathfrak{R})} = 2^{8.56} = 377$. Incidentally, the maximum value here is 4096 since this would occur if each boxed value is 1.0, giving individual scores $s_i = \log_2(1/0.25) = 2$, so $S = 12$.

Note that this model is simplistic in the sense of assuming that the state or letter at a given position is unaffected by the state or letter at the previous position. That said, this model extends the *iid* model – that is, that the same distribution applied for all positions.

9.2.2. Practical Issues: these include:

- Sometimes care needs to be exercised with first aligning the sequences
- Care also needs to be exercised with choosing w, the extent of the site (see Section 9.4)
- Working with just the training set to discover the model can result in small sample size; to correct for this small sample size, we use to estimate probabilities:

$$p_{a_i} = \frac{n_{a_i} + 1}{N + 4} = \omega \times \frac{n_{a_i}}{N} + (1 - \omega) \times \frac{1}{4}$$

  This equation adds a "pseudo-count" of 1 for each base at each position. With $\omega = \frac{N}{N+4}$, it is also a weighted average of the usual estimator ($\frac{n_{a_i}}{N}$) and the guess with no prior knowledge ($\frac{1}{4}$)
- With signals of length w, the PWM contains $4w$ parameters (but since the probabilities sum to one, we need to estimate $3w$ parameters); if the training set contains only 6 members, there would be on average only 2 observations to estimate each parameter (i.e., $6w$ observations to estimate $3w$ parameters). Sometimes, this is a challenge in a wet lab setting, and we may wish to use cross-validation (leave-one-out) techniques.

Computational Example 9.1: uses the site to which GATA-1 binds; this transcription factor regulates transcription in hematopoietic cells. The string representing its binding site has $w = 6$, and the consensus is (`A/T`)`GATA`(`A/G`) – hence the name. 49 human sites are analyzed below:

```
gata=read.csv("c:\\hello.csv",sep=",",header = FALSE)
bg<-c(0.295,0.205,0.205,0.295)

makepwm.pwm<-function(x,bg) {
      # x = matrix of N aligned sites coded numerically
      # bg = vector (1x4) of background base frequencies
  L<-length(x[1,])      # Number of positions in each site
  N<-length(x[,1])      # Number of sites
  pwm<-matrix(rep(1,4*L),nrow=4)
      # pwm initialized to 1 for each matrix element (pseudocounts)
  for (j in 1:L) {
   for (i in 1:N) {
    k <- x[i,j]
    pwm[k,j] <- pwm[k,j]+1
   }
  }
  N <- N+4 # Denominator for small sample correction
  pwm<-pwm/N      # PWM in terms of probabilities
  log2pwm<-matrix(rep(0,4*L),nrow=4,ncol=L)
      # Initialize PWM in terms of log(base 2) of p/q
  for(i in 1:4) {
   log2pwm[i,]<-log2(pwm[i,]/bg[i])
       # Scores for each [nucleotide, position], base 2
  }
  return(pwm, log2pwm)
}
```

```
tmp.pwm<-makepwm.pwm(gata,bg)
tmp.pwm
         [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,] 0.56603774  0.03773585  0.92452830  0.03773585  0.69811321  0.43396226
[2,] 0.05660377  0.03773585  0.01886792  0.01886792  0.07547170  0.05660377
[3,] 0.03773585  0.84905660  0.03773585  0.05660377  0.05660377  0.39622642
[4,] 0.33962264  0.07547170  0.01886792  0.88679245  0.16981132  0.11320755
```

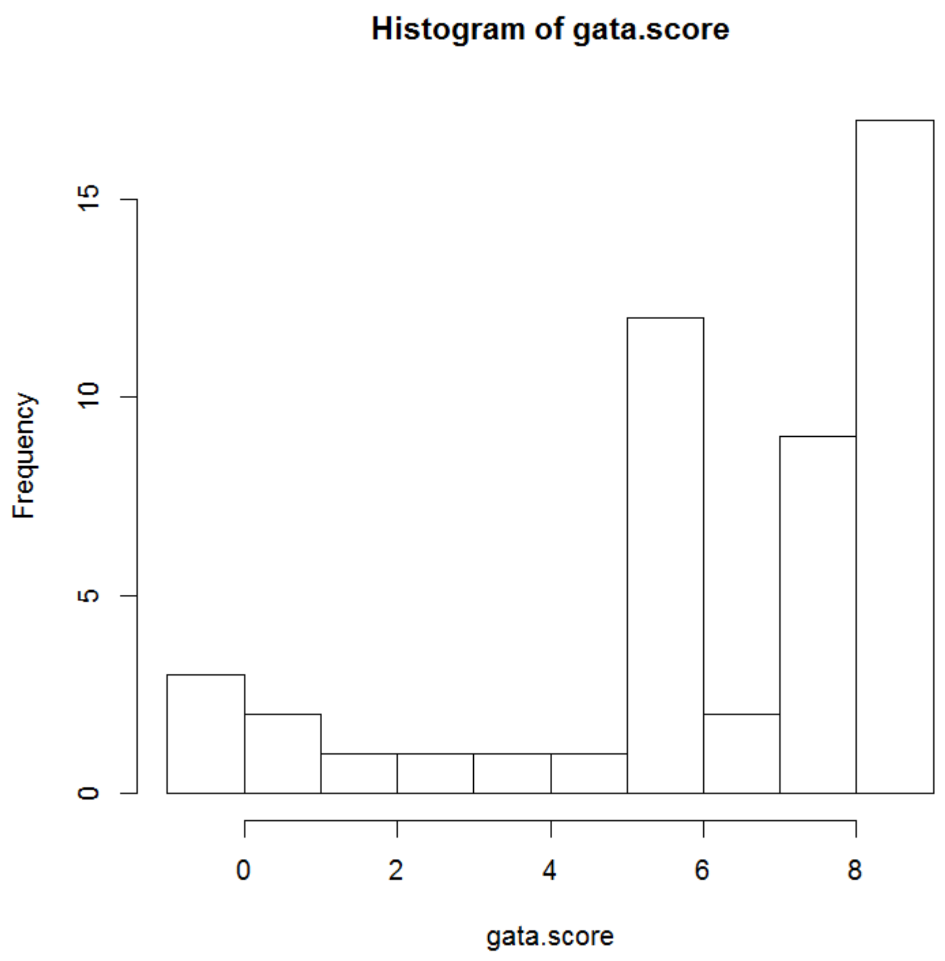
```
tmp.log2pwm<-makepwm.log2pwm(gata,bg)
tmp.log2pwm
           [,1]        [,2]       [,3]       [,4]        [,5]        [,6]
[1,]  0.9401833   -2.966707   1.648003  -2.966707   1.2427461   0.5568546
[2,] -1.8566538   -2.441616  -3.441616  -3.441616  -1.4416163  -1.8566538
[3,] -2.4416163    2.050237  -2.441616  -1.856654  -1.8566538   0.9507012
[4,]  0.2032177   -1.966707  -3.966707   1.587882  -0.7967823  -1.3817448
```

Note: above, we see: (A/T)GATA(A/G)

Next, let's find the scores for the 49 reads (binding sites):

```
calcscore<-function(seq,log2pwm){
# seq is a vector representing input DNA numerically
# log2pwm is a PWM (4xL) with elements as log base 2
score <- 0
for (j in 1:length(log2pwm[1,])){
score<-score+log2pwm[seq[j],j]}
return(score)
}
gata.score<-rep(0,length(gata[,1]))
for(i in 1:length(gata[,1])) {
gata.score[i]<-calcscore(gata[i,],tmp.log2pwm)
}
```

```
> signif(gata.score,3)
 [1]  3.670  6.090  8.420 -0.614  8.420  8.420  8.420  7.290  8.030  8.030  5.620  0.603  7.290
[14]  7.290  5.320  8.420  6.380  8.030  7.290  8.030  8.420  7.680  4.600  0.588  5.320  5.350
[27]  8.420  7.290  2.230  8.030  5.230  7.680  8.030  5.250  8.030  8.420  5.350  8.030 -0.254
[40] -0.687  5.990  7.680  7.680  5.610  5.640  5.990  1.430  8.030  5.990
```

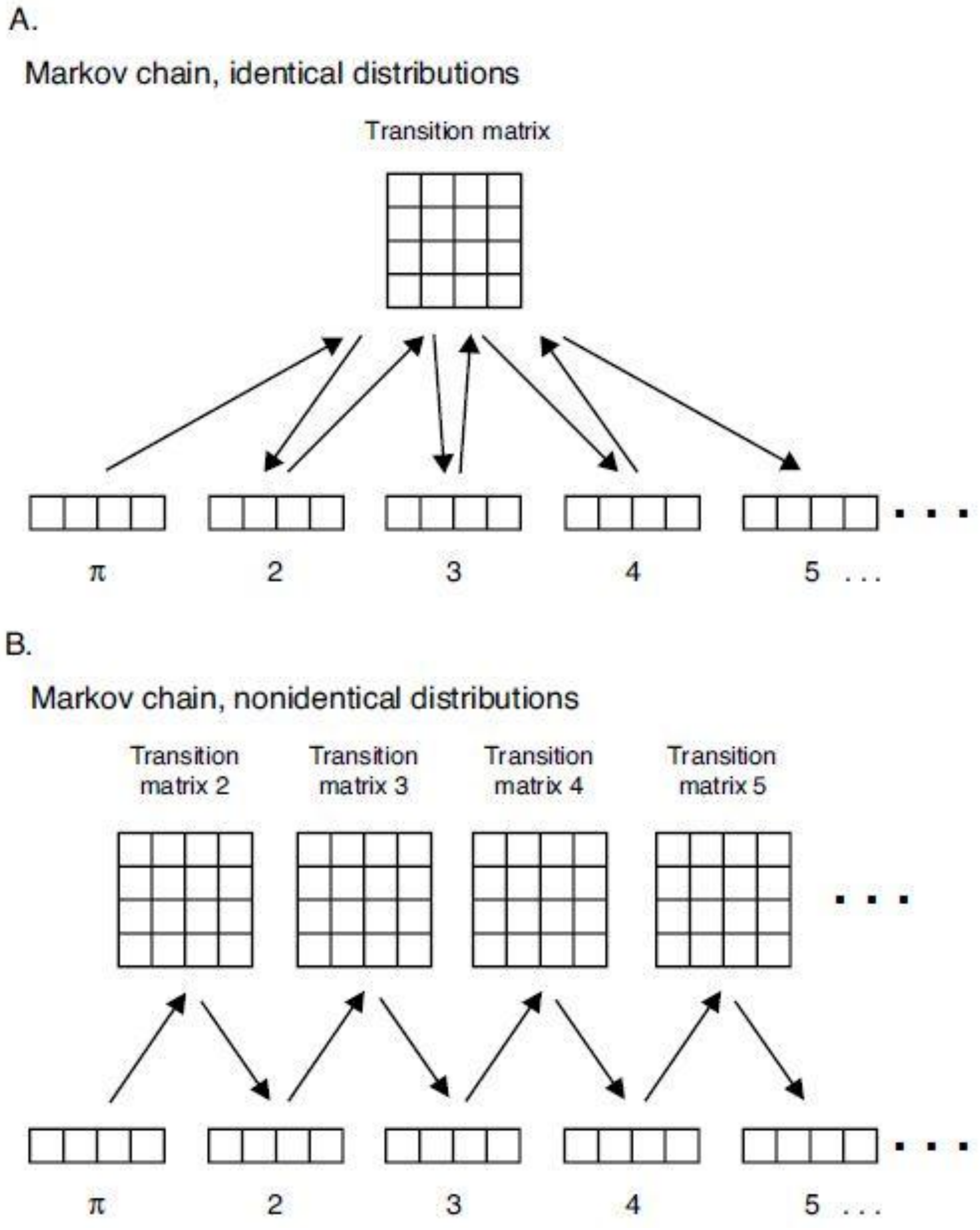


As an exercise, using the above, note that:

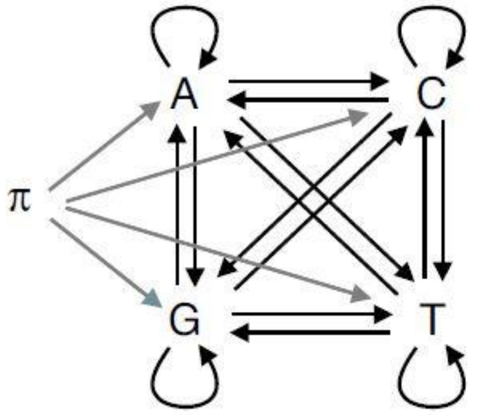
- **P("TTATAG") = P("441413") = (0.33962)*(0.07547)*(0.92453)*(0.88679)*(0.69811)*(0.39623) = 0.0058129**
  So, S = $\log_2$(0.0058129) – $\log_2$(0.000457998) = 3.6658
- **P("TGATAA") = P("431411") = (0.33962)*(0.84906)*(0.92453)*(0.88679)*(0.69811)*(0.43396) = 0.0716221**
  So, S = $\log_2$(0.0716221) – $\log_2$(0.000457998) = 7.2889

9.3. Representing Signals in DNA: Markov Chains –

PWM suffers from the criticism that it assumes states at position k and $k+1$ are independent, and we now extend to the Markov chain strategy used in Chapter 2 – but with the extension that the transition matrix from state k to state $k+1$ are no longer assumed to be homogeneous (i.e. this matrix depends upon $k+1$); see the following Figure.



Think of # of parameters under each of these models. For reads of length L, the # of parameters was $3L$. For a homogeneous Markov model, there are $3 + 12 = 15$ parameters. For our non-homogeneous Markov model, there are $3 + 12(L - 1)$. These respective numbers for $L = 6$ are: 18, 15 and 63.

For this non-homogeneous Markov model, Durbin *et al* (1998) give the diagram at right to demonstrate the process.	

Following is Computational Example 9.2 (starting in the text on p.244): it is helpful to understand the R code to see how this non-homogeneous Markov model process works.

```
gata2=read.table("c:\\gata.txt",header = FALSE)
length(gata[gata[,1]==1,1])
[1] 29
length(gata[gata[,1]==2,1])
[1] 2
length(gata[gata[,1]==3,1])
[1] 1
length(gata[gata[,1]==4,1])
[1] 17
vectorn<-c(29,2,1,17)
```

```
vectorn
[1] 29  2  1 17
vector1<-(vectorn+1)/(49+4)
vector1
[1] 0.56603774 0.05660377 0.03773585 0.33962264

Above is π (initial probabilities for state 1)

matrix0<-matrix(nrow=4,ncol=4,rep(0,16))

transmatp.matrixp<-function(sites,col,matrix0) {
    #sites = numeric matrix of n binding sites, w positions
    #col = column that transition matrix produces
    #matrix0 = matrix of counts for n = col, initialized to 0
matrixn<-matrix0
for(i in 1:length(sites[,1]))  {
  j<-sites[i,(col-1)]
  matrixn[j,sites[i,col]]<-matrixn[j,sites[i,col]]+1
  }
    #Change counts to probabilities
matrixp<-matrixn
matrixp<-matrixp+1 #Adds 1 to every element
for(i in 1:4)  {
  matrixp[i,]<-matrixp[i,]/sum(matrixp[i,])
   #Denominator=sum(matrixn[i,])+4
  }
return(matrixp)
}
tmp.matrixn<-transmatp.matrixn(gata,2,matrix0)
tmp.matrixn
     [,1] [,2] [,3] [,4]
[1,]    1    0   27    1
[2,]    0    0    2    0
[3,]    0    0    1    0
[4,]    0    1   14    2

tmp.matrix2<-transmatp.matrixp(gata,2,matrix0)
tmp.matrix2
           [,1]       [,2]      [,3]       [,4]
[1,] 0.06060606 0.03030303 0.8484848 0.06060606
[2,] 0.16666667 0.16666667 0.5000000 0.16666667
[3,] 0.20000000 0.20000000 0.4000000 0.20000000
[4,] 0.04761905 0.09523810 0.7142857 0.14285714

tmp.matrix3<-transmatp.matrixp(gata,3,matrix0)
tmp.matrix3
          [,1]       [,2]       [,3]       [,4]
[1,] 0.2000000 0.20000000 0.40000000 0.20000000
[2,] 0.4000000 0.20000000 0.20000000 0.20000000
[3,] 0.9375000 0.02083333 0.02083333 0.02083333
[4,] 0.5714286 0.14285714 0.14285714 0.14285714

tmp.matrix4<-transmatp.matrixp(gata,4,matrix0)
tmp.matrix4
           [,1]       [,2]       [,3]      [,4]
[1,] 0.03846154 0.01923077 0.05769231 0.8846154
[2,] 0.25000000 0.25000000 0.25000000 0.2500000
[3,] 0.20000000 0.20000000 0.20000000 0.4000000
[4,] 0.25000000 0.25000000 0.25000000 0.2500000
```

```
tmp.matrix5<-transmatp.matrixp(gata,5,matrix0)
tmp.matrix5
          [,1]       [,2]       [,3]       [,4]
[1,] 0.2000000  0.2000000  0.2000000  0.4000000
[2,] 0.2500000  0.2500000  0.2500000  0.2500000
[3,] 0.3333333  0.1666667  0.1666667  0.3333333
[4,] 0.7200000  0.0800000  0.0600000  0.1400000

tmp.matrix6<-transmatp.matrixp(gata,6,matrix0)
tmp.matrix6
          [,1]        [,2]        [,3]        [,4]
[1,] 0.4000000  0.07500000  0.4250000  0.1000000
[2,] 0.4285714  0.14285714  0.1428571  0.2857143
[3,] 0.1666667  0.16666667  0.5000000  0.1666667
[4,] 0.5000000  0.08333333  0.2500000  0.1666667
```

Now, using this model, find P("TTATAG") and P("TGATAA") as we did above for the PWM model. It is an exercise to turn these probabilities into scores (see text p.248). Here:

- **P("TTATAG") = P("441413") = (0.33962)*(0.14296)*(0.57143)*(0.88462)*(0.7200)*(0.42500) = 0.007504754**
- **P("TGATAA") = P("431411") = (0.33962)*(0.71429)*(0.93750)*(0.88462)*(0.7200)*(0.4000) = 0.057941115**

9.4. Entropy and Information Content: there are two measures given here. These are:

- **Shannon's Entropy: measures uncertainty associated with a set of possible outcomes. Discrete RV X has outcomes $x_1, x_2, \ldots, x_J$ with probabilities $p(x_1), p(x_2), \ldots, p(x_J)$.**

Then Shannon's Entropy is defined as

$$H(X) = -\sum_{j=1}^{J} p(x_j) \log_2 p(x_j)$$

If we look at a sequence $X_1, X_2, \ldots$ of iid bases, the entropy of the i^{th} position is

$$H(X_i) = -\sum_{a \in \{A,C,G,T\}} p(a) \log_2 p(a)$$

If the outcomes are equally likely (all $p(a) = \frac{1}{4}$), then $H(X_i) = 2$ (two bits: one to determine purine versus pyramidine, and the other to determine which purine or which pyramidine); if instead we know the base is `A`, so $p(a) = 1$ if $a = A$, then $H(X_i) = 0$ (no uncertainty).

Information is a measure of how much the entropy is reduced after a "signal" has been received:

$$I(X_i) = H_{before} - H_{after}$$

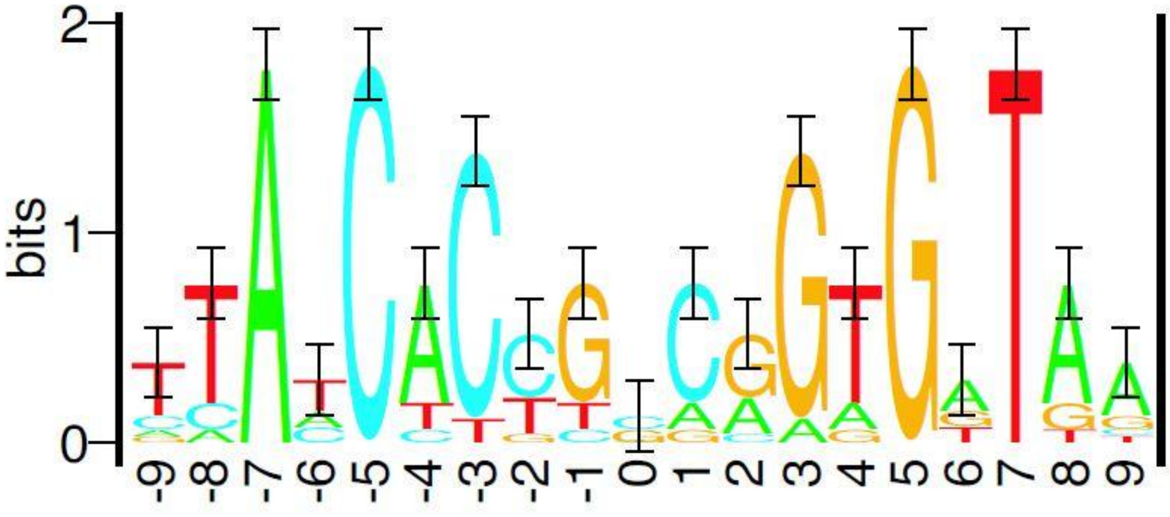
So, if we move from the equally likely case above to one where `A` is certain, the information is $I(X_i) = 2$ bits.

- **Relative Entropy (also called Kullback-Leibler distance): again let p_{ai} denote the probability of finding base a at position i, and q_a represents the background distribution of bases in a genome or in a random model of the genome. Then, the Relative Entropy is**

$$H(p_i \parallel q) = \sum_{a \in \{A,C,G,T\}} p_{ai} \log_2 (p_{ai}/q_a)$$

Notice in the above that p_i and q correspond to distributions (probabilities at each $a \in \{A, C, G, T\}$). This expression is also the expected score since $E(S_i) = \sum_a p_{ai} s_a = \sum_a p_{ai} \log_2(p_{ai}/q_a)$.

The reason we consider these two measures is to define the extent of signals of a binding site; one approach is to plot information content as a function of position along the set of sequences. The DNA sequence logo of lambda operator sites (given in Table 9.1, p.231) is below: this is a graphical representation of a signal *in which the total height corresponds to the relative entropy*. Additionally, the height of each *letter* at each position is calculated by multiplying the relative entropy at that position by the frequency of the corresponding letter. Logos indicate both the amount of relative entropy at each position (the height of the stack of four letters) and the relative contribution of each base (the relative height of the letter) at that position.



9.5. Signals in Eukaryotic Genes: read through.

9.6. Using Scores for Classification: here we'll use the scores from above to classify a candidate string as either a site or nonsite. The null hypothesis $\mathcal{H}$ is that the sequence is a site; when assigning the sequence to a site or not, we get:

		Assigned Class is:	
		True	False
$\mathcal{H}$ is:	True	correct	Type I error
	False	Type II error	correct

So, a Type I error is rejecting $\mathcal{H}$ (concluding the sequence is a nonsite) when $\mathcal{H}$ is true (the sequence is a site), and a Type II error is classifying the sequence as a site when it is not. Also, Sensitivity (Sn) and Specificity (Sp) are:

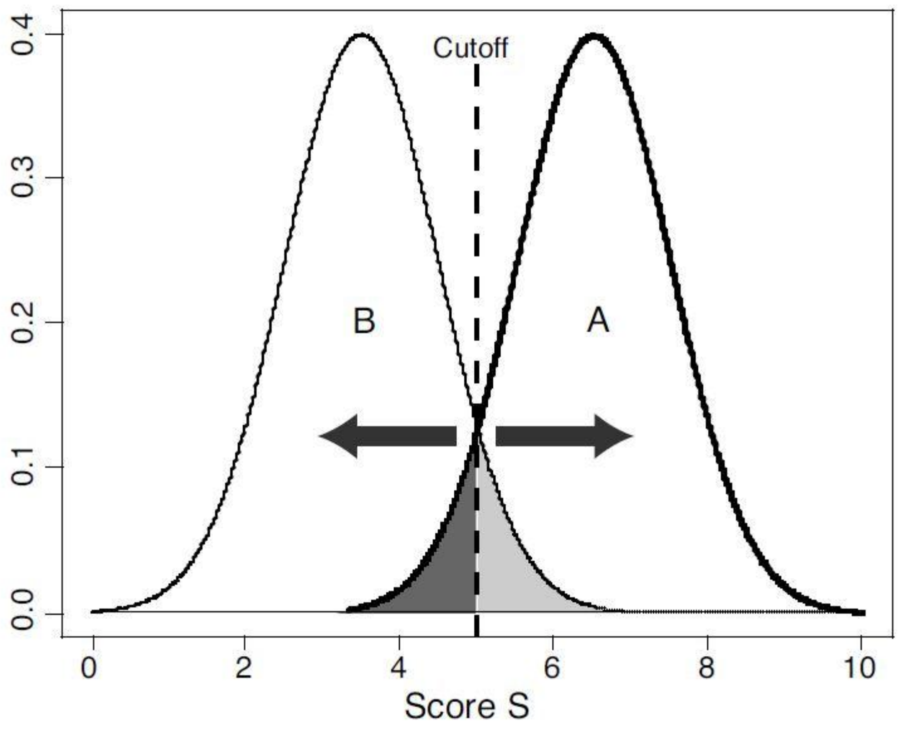
$$Sn = 1 - P(Type\ I\ error)$$
$$Sp = 1 - P(Type\ II\ error)$$

If $\#TP$ is the number of true positive predictions, $\#FP$ is the number of false positive predictions, $\#TN$ is the number of true negative predictions, $\#FN$ is the number of false negative predictions, then we can estimate as follows:

$$Sn \approx \frac{\#TP}{\#TP + \#FN} \quad and \quad Sp \approx \frac{\#TN}{\#TN + \#FP}$$

Given a dataset of sites, scoring gives us a distribution of scores for sequences as in the idealized distribution A below;

also, given a set of nonsites, the same scoring method may give the other distribution B below. We'll then place a cutoff C (vertical line) and classify any score below C as a nonsite and any score above C as a site. Below, the area under the A curve to the left of C (dark shaded region) is the fraction of FN, and the area under the B curve to the right of C (light shaded region) is the fraction of FP. We can move C to lower one of these, but we then increase the other.



In practice, we're not so much interested in the *areas* above but the corresponding *numbers*: when many more nonsites than sites are scored, $\#FP$ tends to be very large and we will want to lower the sensitivity and increase the specificity.

Another useful measure is the false discovery rate:

$$FDR \approx \frac{\#FP}{\#FP + \#TP}$$

Computational Example 9.3: here we classify the GATA-1 sites of p.238 using the PWM matrix from C/E 9.1 (p.240); we do so by simulating in R.

First, simulate GATA-1 sites:

```
simmotif<-function(pwm){
   # pwm is a PWM matrix of probabilities (4xL)
L<-length(pwm[1,]) #Number of positions in the motif
motif<-rep(0,L) #Create and initialize motif vector
dna<-c(1,2,3,4) #Numeric codes for A, C, G, T
for (j in 1:L)  {
   motif[j]<-sample(dna,1,p=pwm[,j])
}
return(motif)

N<-5000
gata.motifs.score<-rep(0,N)
   #vector to hold the results of computation
gata.motifs<-matrix(nrow=N,ncol=6)
for(i in 1:N) {
  gata.motifs[i,]<-simmotif(tmp.pwm)
}
gata.motifs[1:5,]
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    3    1    4    4    3
[2,]    1    3    1    4    4    3
[3,]    4    3    1    4    1    1
[4,]    4    3    1    4    1    3
[5,]    1    3    1    4    4    1
```

Now get the scores for the simulated results:

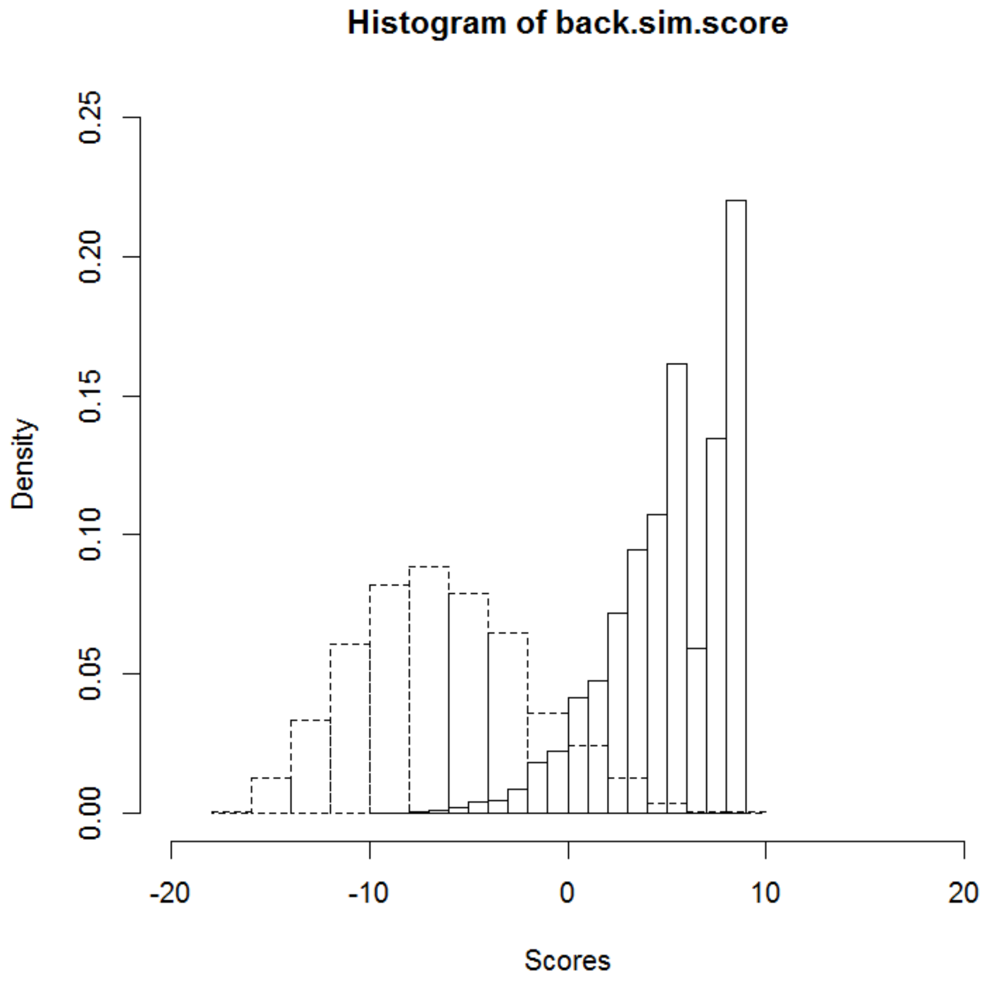
```
gata.motifs.score<-rep(0,N)
   #vector to hold the results of computation
for(i in 1:N) {
  gata.motifs.score[i]<-calcscore(gata.motifs[i,],tmp.log2pwm)
}
```

Now simulate the nonsites (background):

```
simbg<-function(bg,L){
   #bg is a vector of probabilities for A, C, G, T (1x4)
   #L = length of sites to be simulated
seq<-rep(0,L)
dna<-c(1,2,3,4) #Numeric codes for DNA
seq<-sample(dna,L,replace=T,p=bg)
return(seq)
}

back.sim<-matrix(nrow=N,ncol=6)
for(i in 1:N) {
  back.sim[i,]<-simbg(bg,6)
}
back.sim.score<-rep(0,N)
for(i in 1:N) {
  back.sim.score[i]<-calcscore(back.sim[i,],tmp.log2pwm)
}
```

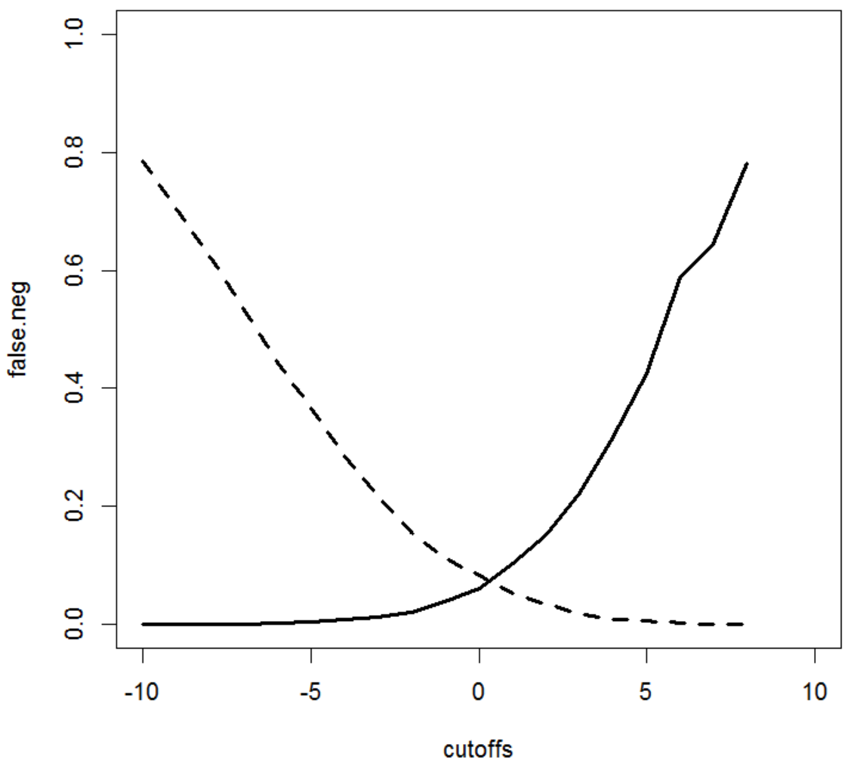
```
hist(back.sim.score,prob=T,xlim=c(-20,20),ylim=c(0,0.25),lty=2,xlab="Scores")
hist(gata.motifs.score,prob=T,xlim=c(-20,20),ylim=c(0,0.25),lty=1,xlab="",add=T)
```



Lastly, we choose different cutoffs and calculate the estimated false negative and false positive rates:

```
cutoffs<-c(-10:8)
false.neg<-rep(0,19)
for(i in 1:19) {
  false.neg[i]<-
    length(gata.motifs.score[gata.motifs.score<cutoffs[i]])/N
}
```

```
false.pos<-rep(0,19)
for(i in 1:19) {
false.pos[i]<-
    length(back.sim.score[back.sim.score>cutoffs[i]])/N
}
plot(cutoffs,false.neg,type="l",
  xlim=c(-10,10),ylim=c(0,1),lwd=3)
points(cutoffs,false.pos,type="l",lty=2,lwd=3)
```



See the discussion on pp.258-9.