greater number of large fragments in the case of bacteriophage lambda DNA. To determine whether the distribution for lambda DNA differs significantly from the mathematical model (exponential distribution), we could break up the length axis into a series of "bins" and calculate the expected number of fragments in each bin by using the exponential density. This would create the entries for a histogram based on the mathematical model. We could then compare the observed distribution of fragments from lambda DNA (using the same bin boundaries) to the expected distribution from the model by using the χ^2 test, for example. Further details are given in Exercise 13.

3.6 k-word Occurrences

The statistical principles learned in this and the previous chapter can be applied to other practical problems, such as discovering functional sites in promoter sequences. Recall from Section 1.3.4 that promoters are gene regions where RNA polymerase binds to initiate transcription. We wish to find k-words that distinguish promoter sequences from average genomic sequences. Because promoters are related by function, we expect to observe k-words that are over-represented within the promoter set compared with a suitable null set. These k-words can help identify DNA "signals" required for promoter function. (DNA signals are described in detail in Chapter 9.) Using the approaches of Chapter 2, we determine expected k-word frequencies and compare them to the observed frequencies. Distributions presented in this chapter are used to test whether over-represented k-words appear with significantly higher frequencies.

Consider N promoter sequences of length L bp, which we denote by S_i for $i = 1, \ldots, N$ (Table C.2). The null set might consist of N strings of L iid letters, each letter having the same probability of occurrence as the letter frequencies in genomic DNA as a whole. For the purposes of the discussion here, we take a small word size, $k = 4$, so that there are 256 possible k-words. With no a priori knowledge of conserved patterns, we must examine all 256 words. We ask whether there are an unusual number of occurrences of each word in the promoter regions.

For the 49 promoter sequences shown in Table C.2 in Appendix C, we first evaluate the most abundant observed k-words and their expected values for $k = 4$ using R for the computation described in Computational Example 3.6. The expectation of each word according to the null (iid) model is easy to calculate if words are overlapping. For example, if X_w denotes the number of occurrences of word w in the whole set of sequences, then for $w = \texttt{ACGT}$,

$$\mathbb{P}(w = \texttt{ACGT}) = p_{\texttt{A}}p_{\texttt{C}}p_{\texttt{G}}p_{\texttt{T}}$$
$$\mathbb{E}(\#\text{ times } w \text{ appears in } S_i) = (L-4+1)p_{\texttt{A}}p_{\texttt{C}}p_{\texttt{G}}p_{\texttt{T}}$$

and the expected number of occurrences in N such sequences is

$$\mathbb{E}(X_w) = N(L-4+1)p_{\texttt{A}}p_{\texttt{C}}p_{\texttt{G}}p_{\texttt{T}}. \tag{3.14}$$

Computational Example 3.6: Counting k-words in promoter sequences

The data in Table C.2 are stripped of row labels, and `A`, `C`, `G`, and `T` are coded numerically as 1, 2, 3, and 4 separated by spaces, as has been our usual practice. The result is saved as a text file, `Ec.table.txt`, which is then read into a matrix in R:

```
>ec.prom<-matrix(scan("Ec.table.txt"),nrow=49,byrow=T)
Read 2499 items
```

We must first decide to what we wish to compare the promoters. In this example, we compare them with the average *E. coli* sequence, and in an exercise you will compare them with sequences having the promoter base composition. The base frequencies for the *E. coli* genome are, for `A`, `C`, `G`, and `T`, respectively, (see Table 2.1):

```
> prob.ec
[1] 0.246 0.254 0.254 0.246
```

These values are employed for calculating the expected value of each k-word and later for simulating sequences under the null model.

Since we are concerned with k-words having $k = 4$, it is convenient to store the results of our calculations in four-dimensional arrays. We can think of these as four different three-dimensional arrays, each labeled 1, 2, 3, 4, but this visualization is not required for handling the array objects under R. We represent the word size, k, as w in this function, so $w = 4$ in our example. Because the base frequencies are, in general, all different, the expected word frequencies are not all identical. The code to generate the expected frequencies is:

```
> expect4.ec<-array(rep(0,4^w), rep(4,w))
> for(i in 1:4){
    for(j in 1:4){
      for(k in 1:4){
        for(m in 1:4){
              expect4[i,j,k,m]<-
+  48*49*prob.ec[i]*prob.ec[j]*prob.ec[k]*prob.ec[m]
  } } } }
```

The number 49 corresponds to N, the number of sequences, and 48 comes from $L - w + 1 = 51 - 4 + 1$, where 51 is the number of bases listed for each string. The expected frequency of each k-word is read from the corresponding array element. For example, if the word is `GATC`, the coded word is represented as 3142, and the expected frequency of that word is contained in the `expect4[3,1,4,2]` array element. The function below is used to perform the

word count, accumulating the total counts for each word in a four-dimensional array, `tcount`. The plug-in portion is included for the computation to be performed in the next box. For counting only, we could "comment out" all lines in the plug-in, and delete the `ncount` in the `return()` statement. For now, we just concentrate on `tcount`.

```
Ncount4<-function(seq,w){
#w=length of word
tcount<-array(rep(0,4^w), rep(4,w))
# array[4x4x4x4] to hold word counts, elements set to zero
ncount<-array(rep(0,4^w), rep(4,w))
# array[4x4x4x4]  holds number of sequences with one or
# more of each k-word
N<-length(seq[1,]) #Length of each sequence
M<-length(seq[,1]) #Number of sequences
#########################################################
#Count total number of word occurrences
for(j in 1:M){   #looping over sequences
  jcount<-array(rep(0,4^w), rep(4,w))
  #array to hold word counts for sequence j
    for(k in 1:(N-w+1)){ #looping over positions
    jcount[seq[j,k],seq[j,k+1],seq[j,k+2],seq[j,k+3]]<-
    jcount[seq[j,k],seq[j,k+1],seq[j,k+2],seq[j,k+3]]+1
    #adds 1 if word at k,k+1,k+2,k+3 appears in sequence j
    }
    tcount<-tcount+jcount
    #Add contribution of j to total
#########################################################
#Plug-in: add 1 to ncount if word occurs >= once in j
  for(k in 1:4){
    for(l in 1:4){
      for(m in 1:4){
        for(n in 1:4){
          if(jcount[k,l,m,n]!=0){
          ncount[k,l,m,n]<-ncount[k,l,m,n]+1}
          }
        }
      }
    }
#########################################################
}
return(tcount,ncount)
}
```

The word count is performed on the promoter sequences:


```
> prom.count<-Ncount4(ec.prom,4)
> sum(prom.count$tcount)
[1] 2352
```

The last two lines verify that the expected number of words have been counted: $48 \times 49 = 2352$. Note that since two items are being returned from the computation, we must specify which item in the list we require; in this case, `prom.count$tcount`. The most frequently occurring word appears 33 times:

```
> max(prom.count$tcount)
[1] 33
```

We find that ten words appear more than 20 times in this set of promoter sequences:

```
>(1:256)[prom.count$tcount[prom.count$tcount[,,,]>20]]
 [1] 23 22 21 21 23 24 24 25 22 33
```

By inspection, we identify these words in the output of the array contents and tabulate the result in Table 3.2. Expected values are taken from corresponding elements of `expect4`.

Table 3.2. Observed and expected k-word frequencies in *E. coli* promoter sequences for $k = 4$. Expected values were computed under the iid null model where $p_A = p_T = 0.246$ and $p_G = p_C = 0.254$. Only the ten most abundant words, based on the total number of apperances in all promoter sequences, are shown. Example promoter sequences are shown in Table 9.2.

Word	Observed frequency	Expected frequency
TTTT	33	8.6
CATT	25	8.9
AATT	24	8.6
TAAT	24	8.6
ATTG	23	8.9
TGAA	23	8.9
ATAA	22	8.6
ATTT	22	8.6
TTTA	21	8.6
ATTC	21	8.9

These results suggest that promoters have unusual word composition compared with the iid null model. These words are composed mostly of As and Ts, the most frequent letters in the promoter set. We must determine whether these elevated abundances are statistically significant.

If a k-word is to be identified as significantly over-represented in promoters compared with the null set, we need to know the expected number of occurrences, $\mathbb{E}(X_w)$, and the standard deviation of this number. In other words, we need to know how the values X_w are distributed. A computational approach is to repeatedly simulate sets of N iid sequences, perform the word counts, and then produce a histogram of the observed values of X_w for each word w. The resulting distributions for each word might not be normal, but we could determine thresholds such that an appropriately small fraction of observations (0.001, for example) fall outside this range. If the distributions of the X_w were approximately normal, then a significance level of three standard deviations would correspond to a probability of approximately 0.0013. If there were $N = 100$ sequences, the expected number of words that are three or more standard deviations above the mean would therefore be 0.13.

We do not simply compute $\mathbb{E}(X_w)$ and $\mathrm{Var}(X_w)$ from first principles because both $\mathbb{E}(X_w)$ and $\mathrm{Var}(X_w)$ depend on how the words are counted. For example, if $k = 4$, what is X_w for $w =$ AAAAA? If word overlaps are allowed, $X_w = 2$, whereas if word overlaps are not allowed, $X_w = 1$. If $p_A = p_C = p_G = p_T = 0.25$ and word overlaps are allowed, $\mathbb{E}(X_w)$ is identical for each word of length k (see (3.14), but if overlaps are not allowed, $\mathbb{E}(X_w)$ in general differs for different words having the same k. With either way of counting words, $\mathrm{Var}(X_w)$ is not the same for each word. There are basically two approaches for word counting. One is to count all occurrences of the word in the whole set of N regions S_i as we did above, and the second is to count the number of promoter sequences in which the word occurs at least once.

The simple, naive basis we use for deciding whether w occurs with unusual frequency is to take each word w and tabulate the number of promoter sequences N_w in which the word occurs at least once. This alternative statistic conforms to the normal approximation of the binomial distribution. First, we simulate 5000 sequences with letter probabilities corresponding to the *E. coli* genome. We use the simulations to estimate

$$p_w = \mathbb{P}(w \text{ occurs at least once in a 51-letter sequence}) \approx \frac{\#\text{ of sequences in which } w \text{ appears at least once}}{5000}.$$

The reason for using "at least once" is that the word may appear at multiple locations in the promoter, with only one occurrence at a particular location being sufficient for function.

The simulation provides an estimate of p_w that can be used with the normal approximation of the binomial with $n = 49$ trials and success probability p_w. Let N_w denote the number of promoter sequences in which w appears at least once. Then the statistic

$$Z_w = \frac{N_w - 49p_w}{\sqrt{49p_w(1-p_w)}}$$

has approximately an N(0, 1) distribution, which allows p-values to be computed for each word w. The results of this simulation are shown in Computational Example 3.7.

Computational Example 3.7: Number of promoter sequences containing at least one frequent k-word and statistical significance

Step 1: Compute the number of promoter sequences, N_w, containing each k-word

This time, we want to count the number of promoter sequences that contain at least one occurrence of each word. This quantity is computed by the plug-in in the function provided in Computational Example 3.6. The series of four "for" loops in the plug-in examines counts for all words found in sequence j and adds 1 to appropriate elements of `ncount` if a word appears, regardless of the number of times it appears. The desired counts of sequences N_w for any desired k-word `klmn` are extracted from the result of the previous computation, which is a list, as `prom.count$ncount[k,l,m,n]`. For example, the number of promoter sequences containing at least one instance of AAAA is

```
> prom.count$ncount[1,1,1,1]
[1] 13
```

We check the maximum value for N_w among all k-words.

```
> max(prom.count$ncount)
[1] 20 #Maximum value of Nw among all k-words
```

Step 2: Computation of p_w

This is done by simulation. We simulate 5000 sequences, each 51 nucleotides long and having the base composition of average *E. coli* DNA.

```
> ec.sim<-matrix(nrow=5000,ncol=51)
> for(i in 1:5000){
+ ec.sim[i,]<-sample(x,51,replace=T,prob.ec)
+ }
```

Remember that `x` is `[1,2,3,4]` and `prob.ec` is given in Computational Example 3.6. To get the data needed for p_w, we again apply the function `Ncount4()`:

```
> sim.count<-Ncount4(ec.sim,4)
```

This may take a while to run since there are nested loops operating on 5000 sequences. The values of p_w for the most abundant words listed in Table 3.2 are calculated as shown below for AAAA:

```
> sim.count$ncount[1,1,1,1]/5000
[1] 0.1238
```

The results for all words are presented in Table 3.3. Because the simulated sequences were based on chromosomal values for base frequencies, which are all nearly the same, the fraction of sequences for which each word appears should also be about the same.

Step 3: Computing p-values

We can use N_w and p_w computed above to calculate Z_w and then compute the desired p-value using the R function `pnorm` (previously used in Section 3.3). This is because Z_w is expected to follow (approximately) a normal distribution:

```
> Nw<-c(19,20,20,20,20,19,19,19,17,16)  # See step 1

> pw
[1] 0.1238 0.1680 0.1710  ... 0.1660 0.1626 0.1736
```

(From Step 2 above.) We compute Z_w for the ten top-scoring k-words:

```
> options(digits=4)
> Zw<-(Nw-49*pw)/sqrt(49*pw*(1-pw))
> Zw
[1] 5.610 4.497 4.409  ... 4.172 3.497 2.826
```

Calculate the one-tailed p-value:

```
> 1-pnorm(Zw)
 [1] 1.011e-08 3.452e-06 5.185e-06 2.328e-06
 [5] 1.641e-06 1.589e-05 1.510e-05 1.510e-05
 [9] 2.353e-04 2.354e-03
```

Table 3.3. Number of *E. coli* promoter sequences, N_w, containing indicated k-words for $k = 4$. Data for the ten most abundant words listed in Table 3.2 are shown. For the meaning of other quantities, see the text. The last column corresponds to p-values associated with each N_w. Entries not significant at level 0.001 (one-tailed test) are indicated in italics.

Word	N_w	p_w	Z_w	$\mathbb{P}(X > Z_w)$
TTTT	19	0.124	5.610	10^{-8}
CATT	20	0.168	4.497	0.000003
AATT	20	0.171	4.409	0.000005
TAAT	20	0.165	4.580	0.000002
ATTG	20	0.163	4.652	0.000002
TGAA	19	0.166	4.160	0.000016
ATAA	19	0.166	4.172	0.000015
ATTT	19	0.166	4.172	0.000015
TTTA	17	0.163	3.497	0.000235
ATTC	*16*	*0.174*	*2.826*	*0.002354*

Notice that N_w is lower than the number of overall occurrences shown in Table 3.2, as would be expected. From prior knowledge of *E. coli* promoters, we already expected that words contained within `TATAAT` would be abundant. Note that `TAAT` appears in about 40% of the listed promoters, as does `ATAA`. Testing for the occurrence and significance of `TATA` is left as an exercise.

Why did we earlier refer to this analysis as naive? This comes from the strong correlation between word counts, especially that which comes from word overlaps. If, for example, we knew that `AAAA` occurred in a particular string 49 times out of 51, then the end of the string of `A`s must be a letter not equal to `A`. Of these, $p_{\mathtt{T}}/(p_{\mathtt{T}}+p_{\mathtt{C}}+p_{\mathtt{G}})$ are expected to be `T`s. That means that we expect at least $49 \times p_{\mathtt{T}}/(p_{\mathtt{T}}+p_{\mathtt{C}}+p_{\mathtt{G}})$ occurrences of `AAAT`. In addition, there are occurrences of `AAAT` where the preceding letter was not an `A`. Taking all word overlaps into account in a rigorous statistical analysis is beyond the scope of this book, but you now know that N_w has different variances depending on w and that correlation between words is important.

Functional k-words are generally more complicated and more extensive than the example above, and in practice values to be used for k are not known in advance. Certainly $k = 4$ is too small; nevertheless, larger functional k-words can be decomposed into sets of characteristic 4-words. For promoters, $k = 6$ is a more realistic choice. In addition, our idea of counting exact occurrences does not correspond to what is observed in biological systems (e.g., not all promoters contain exact matches to the most frequent 4-words). However, it is often the case that exact word analysis has allowed researchers to make the initial pattern discovery. The approach above is not limited to exact word analysis, however. We could, for example, use $k = 6$ letter words and allow up to two mismatches in the definition of "occurrence."

In Chapter 9, we illustrate how to describe signals of arbitrary length based upon patterns of letter occurrences observed among a set of aligned subsequences. The approach described in the current chapter could be extended to yield a complementary description of such signal sequences. We could implement this for the promoter data set by making histograms of positions at which each over-represented k-word occurs relative to the transcriptional start sites at $+1$. Any "signal" that appears within a window centered at position x relative to the transcriptional start site would then be represented by a word decomposition yielding the observed k-words preferentially mapping within that window (Galas et al., 1985).

References

Galas DJ, Eggert M, Waterman MS (1985) Rigorous pattern-recognition methods for DNA sequences. *Journal of Molecular Biology* 186:117–128.