BIOINFORMATICS & COMPUTATIONAL GENETICS MSc
PROBABILITY AND STATISTICS

S-PLUS EXERCISE SHEET 3

1. Starting the Splus session

Double click the S-PLUS icon, or start the program from the **Start** menu. When the program has opened up the *Object Browser* window, start a *Commands Window* by using the **Window -> Commands Window** pull down menus.

2. **CHECKING THE BINOMIAL/POISSON APPROXIMATION**

We have seen in lectures that a *Binomial* probability distribution may be well-approximated by a *Poisson* distribution; recall that the *Binomial* (n, θ) distribution models the total number of "successes" in a sequence of n independent and identical binary trials, where the probability of success on each individual trial is θ, for which the mass function is

$$\binom{n}{x} \theta^x (1-\theta)^{n-x} .$$

It can be shown that, if n is large and θ is small, with $n\theta = \lambda$, then

$$\binom{n}{x} \theta^x (1-\theta)^{n-x} \approx \frac{e^{-\lambda}\lambda^x}{x!} \qquad \text{that is} \qquad Binomial\,(n,\theta) \approx Poisson(\lambda)$$

We can verify the accuracy of this approximation for different values of the n and θ using S-PLUS .

For a particular combination of parameters $n = 500, \theta = 0.005$, type the following commands:

```
>n <- 500
>theta <- 0.005
>lambda <- n*theta
>x <- c(0:10)
>y.binomial <- dbinom(x,n,theta)
>y.poisson <- dpois(x,lambda)
>plot(x,y.binomial)
>points(x,y.poisson,pch=3)
```

The two plots overlay each other very closely; numerically, the results are very close also:

```
>y.binomial
[1] 0.0815718614 0.2049544257 0.2569654735 0.2143531085 0.1338360489 0.0667162616
[7] 0.0276587517 0.0098086312 0.0030374719 0.0008344144 0.0002058781
>y.poisson
[1] 0.0820849986 0.2052124966 0.2565156207 0.2137630172 0.1336018858 0.0668009429
[7] 0.0278337262 0.0099406165 0.0031064427 0.0008629007 0.0002157252
```

EXERCISE Find out when this approximation starts to break down by choosing smaller values of n and larger values of θ

3. SHOTGUN SEQUENCING SIMULATION

The shotgun sequencing procedure may be simulated using S-PLUS What we will do below is to fix values of the fragment number N and the fragment length L for a fixed genome length G. We have studied the probabilistic behaviour in lectures via the section from the Ewens and Grant textbook, and the aim here is to simulate the whole process.

We begin by fixing some parameters; we will use $G = 100000$ and $L = 500$ and vary the value of N to see how the coverage, contig lengths, gap lengths etc. vary. For a graphical illustration we shall start with $N = 50$.

```
>G <- 100000
>L <- 500
>N <- 50
>index <- c(1:G)
>plot(c(0:G),xlim=range(0,G),ylim=range(0,N),type=''n'',xlab=''Genome'',ylab=''Fragment'')
>for(i in 1:N){
+ lh <- sample(index,1)
+ lines(c(lh,lh+L),c(i,i))}
```

(remember, the + symbols appear automatically when you hit return) *Try to identify what each command does; you have met most of them previously, and if there is a command that you don't understand, please use the* **Help** *pull down menu).*

If you examine the plot (if you can't see it, check the **Window -> GSD2** on the pull down menu), clearly the coverage is very sparse. This can be improved by increasing L (to 5000 say) or N; try this.

It is useful to be able to generate this plot for different values of the input parameters G, L and N without having to repeat all the same commands. To automate the process we can write an S-PLUS **function**; we do this as follows:

```
>shotgun.plot <- function(G,L,N){
+ index <- c(1:G)
+ plot(c(0:G),xlim=range(0,G),ylim=range(0,N),type=''n'')
+ for(i in 1:N){
+ lh <- sample(index,1)
+ lines(c(lh,lh+L),c(i,i))}
+ }
> shotgun.plot(G,L,N)
```

These commands have created the function **shotgun.plot** which takes three arguments G, L and N . You can now type, for example,

```
>shotgun.plot(10000,500,500)
```

and this will produce a plot for $G = 10000, L = 500$ and $N = 500$ automatically. It is often useful to be able to edit a function that you have created; this may be done easily using the command

```
>shotgun.plot <- edit(shotgun.plot)
```

which brings up an editing window. You can make the changes that you require, then quit the editor. If you make an error in the new code that you type, you will get an error message when you quit the editor.

For any simulated experiment, therefore, we can generate the fragments and study the coverage obtained visually. For a more quantitative analysis, we might wish to compute the coverage for any given experiment, or the number of contigs formed, or the length of the largest gap etc. This can be done routinely using S-PLUS . First, we will create an empty vector **genome** into which the fragment information will be imported. Then we will simulate the N fragments as above and then collate the information.

```
>G <- 100000
>L <- 500
>N <- 50
>index <- c(1:G)
>genome <- rep(0,G)
>for(i in 1:N){
+ lh_sample(index,1)
+ genome[lh:min(lh+L,G)] <- genome[lh:min(lh+L,G)]+1}
>plot(genome)
```

This plot shows how many times each base position is covered by a fragment. The vector **genome** is now a list of G elements that were originally zero, but all now be non-zero if that base position has been covered by at least one fragment.

Create a function **genome.plot** to reproduce this plot (*and output the* **genome** *vector*) for any values of G, L or N. as follows:

```
>genome.plot_function(G,L,N){
+ index_c(1:G)
+ genome <- rep(0,G)
+ for(i in 1:N){
+ lh <- sample(index,1)
+ genome[lh:min(lh+L,G)] <- genome[lh:min(lh+L,G)]+1}
+ plot(genome)
+ genome}
```

This function must be executed by typing

```
>genome <- genome.plot(G,L,N)
```

which assigns values to the vector **genome**.

Next we wish to find the contigs and the gaps. To do this, we must loop through the elements of the vector **genome** looking for contiguous stretches of non-zero elements. Graphically, the contigs and gaps can be found by simply issuing the commands

```
>g <- genome
>g[genome > 0] <- 1
>plot(g)
```

In this plot, the contigs appear at the top of the plot, and the gaps appear at the bottom.

EXERCISE Create the function **contig.plot** that plots the contigs and gaps, by typing

```
>contig.plot <- edit(genome.plot)
```

and editing the function accordingly.

To study the newly formed **g** vector of zeros and ones numerically. We must utilize a **for** loop, recording the beginnings and ends of the contigs and the gaps as we go along. We create vectors **contig.start** and **contig.end** to record the left-hand and right-hand contig end positions, and a vector **contig.length** to record the contig lengths. The value **contig.count** will record the number of contigs.

```
>contig.count <- 0
>g1 <- c(0,diff(g))
>index <- c(1:G)
>contig.start <- index[g1 == 1]
>contig.end <- index[g1 == -1]
>if(length(contig.start) > length(contig.end)) {contig.end <- c(contig.end,G)}
>contig.length <- contig.end-contig.start
>contig.count <- length(contig.start)
> contig.count
[1] 38
> contig.start
 [1]   211   976  3339  5267  6298  9427 11112 12712 13642 17750 22516 26393
[13] 31049 36198 37256 41001 44122 47764 53902 54526 55649 57247 58314
[24] 59550 60071 69248 73233 75398 76136 79575 85163 86097 88488 89139 90619
[36] 92304 94651 96390
> contig.end
 [1]    712   2790   5266   6046   9297  11010  12568  13576  17133  22346  26163
[13]  31000  36134  36774  40958  43913  47708  53840  54403
[20]  55600  57224  58233  59434  60051  68766  73086  75366  75899  79299  84962
[31]  86093  88454  89109  90483  92070  94629  96071 100000
```

That is, there are 38 contigs (in my simulation), the first of which starts at position 211 and ends at 712. This method uses a trick (the use of the **diff** function to find the contig starts and ends; this is not the same as the diff command in MAPLE....).

EXERCISE Find the gap positions.

The actual coverage obtained in this simulated experiment can be computed by dividing the accumulated contig length by the genome length G as follows: in my simulation, this gave the following:

```
>coverage <- sum(contig.length)/G
>coverage
[1] 0.93733
```

All other important statistics (mean contig length etc) can be computed from one of the vectors **contig.start** , **contig.end, contig.length** and the value **contig.count.**

To verify the probabilistic results given in lectures, you need to repeat the whole simulation exercise several times. This is most easily done by creating a function.**contig.simulation**, say, that produces a distribution of, say, of numbers of contigs over n_{rep} replications of the experiment

```
>contig.simulation <- function(G,L,N,nrep){
+  contig.counter <- seq(0, nrep)
+  for(irep in 1:nrep) {
+  index <- c(1:G)
+  genome <- rep(0, G)
+  for(i in 1:N) {
+    lh <- sample(index, 1)
+    genome[lh:min(lh + L, G)] <- genome[lh:min(lh + L, G)] + 1
+  }
+  g <- genome
+  g[genome > 0] <- 1
+  g1 <- c(0, diff(g))
+  contig.start <- index[g1 == 1]
+  contig.end <- index[g1 == -1]
+  if(length(contig.start) > length(contig.end)) {
+    contig.end <- c(contig.end, G)
+  }
+  contig.length <- contig.end - contig.start
+  contig.count <- length(contig.start)
+    contig.counter[irep] <- contig.count
+  }
+  contig.counter
+  }
```