BIOINFORMATICS MSc
PROBABILITY AND STATISTICS

SPLUS EXERCISE SHEET 3

1. STATISTICAL HYPOTHESIS TESTING CALCULATIONS

SPLUS is a package with many of the standard tests (Z-test, T-Test, Chi-squared test etc) For example, the function **t.test** allows a one sample or two sample test to be computed, and the pull-down menus **Statistics -> Compare samples** can be used to automate testing procedures. However, the tests can be carried out by construction of the test statistic, critical values and p-values etc from first principles. For example to test the hypothesis H_0 that $\mu = 10$ against a two-sided alternative hypothesis assuming that $\sigma^2 = 2^2$ for a generated sample.of size $n = 20$, type

```
>mu <- 10
>sigma <- 2
>c <- 10
>n <- 20
>x <- rnorm(n,mu,sigma)
>z <- (mean(x)-c)/sqrt(sigma^2/n)
>z
```

which computes the test statistic z, and then

```
>qnorm(0.025)
>qnorm(0.975)
>pvalue <- pnorm(-abs(z))+1-pnorm(abs(z))
>pvalue
```

that computes the critical values and p-value. Recall that, in **SPLUS** the **q-** in functions look up the quantiles (i.e. ordinates) that give a specified probability value in the cdf (and hence are used to find critical values), and the **p-** functions give the probability values for a given quantile (and hence can be used to find p-values).

EXERCISES : For appropriately simulated data sets; think about how to carry out

- a one-tailed one sample **Z-test**
- a one sample **T-test**
- a one sample test on σ^2
- a two sample **Z-test**
- a two sample **T-test** where $\sigma_1^2 = \sigma_2^2 = \sigma^2$ is unknown
- a two sample **F-test** of $\sigma_1^2 = \sigma_2^2$

In each case, think about how the calculation would work for different values of significance level α in both one-and two-sided hypothesis tests. For the two sample t-test, the "pooled" estimate of variance is used in the formula for the test statistic, that is, we need for two samples x and y

$$s_P^2 = \frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{n_X + n_Y - 2}$$

and then use the test statistic t defined by

$$t = \frac{\bar{x} - \bar{y}}{s_P \sqrt{\dfrac{1}{n_X} + \dfrac{1}{n_Y}}}$$

so to compute t

```
>nx <- 10
>ny <- 15
>x <- rnorm(nx,10,1)
>y <- rnorm(ny,10,1)
>sx <- var(x)
>sy <- var(y)
>sp <- ((nx-1)*sx+(ny-1)*sy)/(nx+ny-2)
>t <- (mean(x)-mean(y))/sqrt(sp*(1/nx+1/ny))
```

This code **simulates** the two random vectors x and y of lengths n_X and n_Y; in practice, for real data, we will form the tw vectors by typing or reading in the data

```
>x <- c(43.2,44.3,22.3,31.4,56.6)
>nx <- length(x)
>y <- c(34.7,43.5,65.3,11.3,29.4,19.0,39.6)
>ny <- length(y)
```

2. HYPOTHESIS TESTING FOR GENE EXPRESSION DATA

The (zipped) data set **ALLDATA** and the associated paper **golub.pdf** can be downloaded from

```
http://stats.ma.ic.ac.uk/∼das01/BioinformaticsMSc/Alldata.zip
http://stats.ma.ic.ac.uk/∼das01/BioinformaticsMSc/golub.pdf
```

The data set contains, in two worksheets, relative gene expression measurements for 7130 genes, and for 72 samples from two different tumour types AML and ALL. We wish to discover whether any genes are differentially expressed in one type compared with the other. Use the `importData` command from the command line to import the two worksheets; if the unzipped file is placed in c:\Temp\, then the command will be

```
>all.data <- importData("c:\\Temp\\Alldata.xls",type="EXCEL",pageNumber=1)
>aml.data <- importData("c:\\Temp\\Alldata.xls",type="EXCEL",pageNumber=2)
```

You could also use the **File -> Import Data** pull down menu to import the data into **SPLUS** . Note: take care with the importing. Import each sheet separately into two data sets ALL and AML say, by using the **Options** tab to specify which of the two worksheets is being imported. Take care also to name the new data sets differently on the **Data Specs** tab.

Note: *If you are using* `R`*, the* `importData` *command will not work; you need to use the comma separated data files that are also available on the website, then the function* `read.csv`

```
>all.data <- read.csv("c:\\Temp\\ALLdata.csv")
>aml.data <- read.csv("c:\\Temp\\AMLdata.csv")
```

EXERCISES
(i) Use the two sample hypothesis testing techniques above to test whether there is any evidence of differential expression in any of the genes.
(ii) Explore the different possibilities for testing these hypothesis that there is no differential expression by using the three different tests on the **Statistics -> Compare Samples -> Two Samples** pull down. To find out about the new tests, use the help facility on the dialog box.

3. GOODNESS OF FIT TESTS FOR SEQUENCE COMPARISON

To compare the composition of two nucleotide sequences using a hypothesis test, either using a **chi-squared statistic** or a **likelihood ratio** statistic; for the table

	Nucleotide				
	A	C	G	T	Total
Sequence 1	n_{11}	n_{12}	n_{13}	n_{14}	$n_{1.}$
Sequence 2	n_{21}	n_{22}	n_{23}	n_{24}	$n_{2.}$
Total	$n_{.1}$	$n_{.2}$	$n_{.3}$	$n_{.4}$	n

if we wish to test the hypothesis H_0 that the two sequences have the same (marginal) nucleotide probabilities, $p_A = p_1, p_C = p_2, p_G = p_3$ and $p_T = p_4$ we first compute the estimates of these nucleotide probabilities (using **maximum likelihood estimation**) under the assumption that H_0 **is true**;

$$\hat{p}_1 = \frac{n_{.1}}{n} \qquad \hat{p}_2 = \frac{n_{.2}}{n} \qquad \hat{p}_3 = \frac{n_{.3}}{n} \qquad \hat{p}_4 = \frac{n_{.4}}{n}$$

and then compute the **expected** or **fitted values** if H_0 **is true** are given by

$$\hat{n}_{ij} = n_{i.}\hat{p}_j = \frac{n_{i.}n_{.j}}{n} \qquad i = 1,2,\ j = 1,2,3,4.$$

Two test statistics that are used are the **Chi-squared** and **Likelihood Ratio** (**LR**) statistics:

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{4} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}} \qquad LR = 2\sum_{i=1}^{2}\sum_{j=1}^{4} n_{ij} \log \frac{n_{ij}}{\hat{n}_{ij}}$$

Both of these statistics have an approximate **Chi-squared distribution** $\chi^2_{(r-1)(c-1)} = \chi^2_3$ distribution, again given that H_0 **is true.** Typically, a significance level of $\alpha = 0.05$ is used for this test, and the critical value in a one-tailed test of the hypothesis is at the 0.95 point of this distribution, that is, at 7.81. To perform the calculation, first we produce two simulated DNA sequences with all nucleotide frequencies equal to 0.25.

```
>n1 <- 2000
>n2 <- 2500
>n <- n1+n2
>sequence1 <- sample(c(1:4),n1,prob=c(0.25,0.25,0.25,0.25),rep=T)
>sequence2 <- sample(c(1:4),n2,prob=c(0.25,0.25,0.25,0.25),rep=T)
```

The **sample** command

```
>sample(v,n,prob=p,rep=T)
```

produces a random sample from the elements of the vector **v,** We now have two simulated biological sequences that are essentially very similar. The sequences can be examined by using the **print** and **table** commands

```
>table(sequence1)
>sequence1
>table(sequence2)
>sequence2
```

It should appear that the nucleotide frequencies are roughly in proportion. We now proceed to carry out the tests.

```
>obs.table <- matrix(0,nrow=2,ncol=4)
>fit.table <- matrix(0,nrow=2,ncol=4)
>obs.table[1,] <- table(sequence1)
>obs.table[2,] <- table(sequence2)
>chi.stat <- 0
>lr.stat <- 0
>for(i in 1:2){
+ for(j in 1:4){
+ fit.table[i,j] <- sum(obs.table[i,])*sum(obs.table[,j])/n
+ chi.stat <- chi.stat + ((obs.table[i,j]-fit.table[i,j])^2)/fit.table[i,j]
+ lr.stat <- lr.stat + 2*obs.table[i,j]*log(obs.table[i,j]/fit.table[i,j])
+ }}
>chi.stat
>lr.stat
```

EXERCISE Repeat for different sample lengths, and also by changing the probability vectors in lines 4 and 5, and attempt to get a significant result.

4. SIMULATION-BASED HYPOTHESIS TESTING FOR SEQUENCE ALIGNMENT

We have seen in Exercise 2 how to compute p-values for "run-test" statistics - statistics measuring the longest run of matches in two aligned sequences - by simulating i.i.d. $Geometric(\theta)$ random variables and studying the distribution of the maximum in the sample. We can repeat the exercises in experiments concerned with matching or aligning sequences. We first using a $Binomial\,(N, p_{MATCH})$ model under H_0 for the number of matches in two sequences of length n:

```
>n <- 5000
>pmatch <- 0.25
>sequence1 <- sample(c(1:4),n,prob=c(0.25,0.25,0.25,0.25),rep=T)
>sequence2 <- sample(c(1:4),n,prob=c(0.25,0.25,0.25,0.25),rep=T)
>b <- length(sequence1[sequence1 == sequence2])
>pvalue <- pbinom(b,n,pmatch)
```

EXERCISE Repeat for different sample lengths, and also by changing the probability vectors in lines 3 and 4, in order to get a significant result. To allow for the possibility of mismatches in a sequence alignment, in a run-test situation, the *Negative Binomial* probability sampling model needs to be used. In fact Monte Carlo simulation is the only way that a $p-$value may be computed. The calculation may be achieved as follows, similarly to those methods used in Exercise 3. Here we allow $k = 3$ mismatches, with $p = 0.25$

```
>n <- 5000
>k <- 3
>p <- 0.25
>nits <- 100000
```

```
>xmax_rep(0,nits)
>for(i in 1:nits){
+ x <- rnbinom(n,k,p)
+ xmax[i] <- max(x)+k
+ }
>hist(xmax)
```

The histogram produced essentially contains the frequencies with which the maximum run length is equal to a specified value. In line 7, the value k is added to the maximum value due to the way that the Negative Binomial samples are simulated by **SPLUS**.

EXERCISES (i) Repeat for different sample lengths, and also by changing the probability vectors in lines 3 and 4, in order to get a significant result

(ii) An even more accurate $p-$value can be obtained from repeated simulation of pairs of sequences, with the maximum run lengths being computed by trawling through the sequences looking for runs of consecutive matches allowing up to k mismatches. Repeated simulation of pairs and sequences can be achieved fairly easily

```
>nits <- 10
>n <- 100000
>match.count_rep(0,nits)
>for(i in 1:nits){
+ pmatch <- 0.25
+ sequence1 <- sample(c(1:4),n,prob=c(0.25,0.25,0.25,0.25),rep=T)
+ sequence2 <- sample(c(1:4),n,prob=c(0.25,0.25,0.25,0.25),rep=T)
+ match <- length(sequence1[sequence1 == sequence2])
+ match.count[i] <- sum(match)
+ }
>match.count
```

which reports successive numbers of matches overall. However, computation of the maximal run lengths (with mismatches) for each generated match sequence is not straightforward. Can you construct some code to do this ?