BIOINFORMATICS & COMPUTATIONAL GENETICS MSc
PROBABILITY AND STATISTICS

S-PLUS EXERCISE SHEET 2

PROBABILITY DISTRIBUTION CALCULATIONS

S-PLUS has many facilities for carrying out calculations for probability distributions. There are functions that calculate the probability mass function f_X and the discrete cumulative distribution function F_X for DISCRETE random variables, and the probability density function f_X and the continuous cumulative distribution function F_X for CONTINUOUS random variables. Also, S-PLUS has functions that allows you to simulate random numbers from many standard probability distributions.

The probability distributions that S-PLUS has specially written functions for include the following:

DISCRETE DISTRIBUTIONS	**CONTINUOUS DISTRIBUTIONS**
Binomial	Uniform
Geometric	Exponential
Negative Binomial	Gamma
Poisson	Chi-squared
	Beta
	Normal
	Student-t
	Cauchy
	F

The task today is to use the S-PLUS functions to carry out probability calculations for these distributions

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.

There are four basic functions that are used for probability calculations: using the binomial distribution as an example, the four functions are

```
dbinom
pbinom
qbinom
rbinom
```

notice the first letters d,p,q and r; these letters determine the type of operation that is being carried out. For different distributions, these first letters will always indicate the same type of operation; the last part of the function name determines which distribution is to be used. Specifically `dbinom :`

computes the probability mass function

`pbinom :`

computes the discrete cumulative distribution function

`dbinom :`

computes the inverse cumulative distribution function

`rbinom :`

simulates a random sample from the distribution

If you type

```
>help(dbinom)
```

at the command line, a help screen explains how each function is used. Each function takes a number of arguments that will be described below.

We will initially concentrate on the binomial distribution, but the commands issued are essentially the same for each distribution we wish to use. Recall that the binomial distribution has the following probability mass function:

$$f_X(x) = \binom{n}{x}\theta^x(1-\theta)^{n-x} = \frac{n!}{x!(n-x)!}\theta^x(1-\theta)^{n-x} \qquad x \in \{0, 1, ..., n\}$$

for parameters n (a positive integer) and θ (a probability lying between 0 and 1). In S-PLUS , the parameter θ is written as p

2. Mass and density function calculations.

The probability mass function for the binomial distribution is obtained using the function

```
dbinom(x, size, prob)
```

where x is the vector of points at which we wish to evaluate the mass function, *size* is the parameter n, and *prob* is the parameter p (or θ) Hence to evaluate all the probabilities in a $Binomial\,(10, 0.3)$ distribution, we issue the following sequence of commands at the command line:

```
>x <- c(0:10)
>n <- 10
>p <- 0.3
>dbinom(x,n,p)
```

for which the response is

```
[1] 0.0282475249 0.1210608210 0.2334744405 0.2668279320 0.2001209490 0.1029193452
[7] 0.036756909 0.0090016920 0.0014467005 0.0001377810 0.0000059049
```

What this calculation has done is to create a vector of the integers from 0 to 10 (the range of this distribution), then specified $n = 10$, then specified $p = 0.3$, and then evaluated the mass function at each value in the vector. That is we have evaluated

$$f_X(0) \quad = \mathrm{P}[X = 0] = \binom{10}{0} (0.3)^0 (1 - 0.3)^{10-0} = 0.0282475249$$

$$f_X(1) \quad = \mathrm{P}[X = 1] = \binom{10}{1} (0.3)^1 (1 - 0.3)^{10-1} = 0.1210608210$$

and so on. We can easily assign the probabilities to a vector, and plot the resulting function:

```
>y <- dbinom(x,n,p)
>plot(x,y)
```

for which the response is a point plot of the mass function.

The d- functions for the various distributions, for example,

```
dbinom, dgeom, dnbinom, dpois, dexp, dgamma, dnorm, ...
```

and so on all have this same basic syntax - the only difference is that the parameters for each distribution change. For example, for the *Gamma* distribution, which we have seen with pdf

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \qquad x > 0$$

has an S-PLUS function

```
dgamma(x, shape, rate=1)
```

where *shape* is the α parameter, and *rate* is the β parameter (which has the default value 1 in the function). Hence to plot four *Gamma* pdfs

$$Gamma(2,2) \qquad Gamma(2,1) \qquad Gamma(2,0.5) \qquad Gamma(4,2)$$

on the range we use the following commands

```
>x <- c(0:1000)/100
>y1 <- dgamma(x,2,2)
>y2 <- dgamma(x,2,1)
>y3 <- dgamma(x,2,0.5)
>y4 <- dgamma(x,4,2)
>plot(x,y1,type=''l'')
>lines(x,y2,lty=2)
>lines(x,y3,lty=3)
>lines(x,y4,lty=4)
```

for which the response is a series of line plots of the pdf. The x vector created is a series of 1000 points equally spaced on 0 to 10, and y_1, y_2, y_3 and y_4 are the four evaluated pdf curves. For information the **type="l"** command produces a line (rather than a point) plot, **lines** adds a line to the current plot, and the **lty=2,3,4** commands produce different line styles.

For the different probability models, you need to use the **help** command to find out the precise syntax and parameter specification for each distribution.

3. Distribution function calculations.

Again we begin with the Binomial model, for which the cumulative distribution function is obtained using the function

```
pbinom(q, size, prob)
```

where q is the vector of points (or *quantiles*) at which we wish to evaluate the cdf, *size* is the parameter n, and *prob* is the parameter p (or θ) Hence to evaluate the cdf in a *Binomial* $(10, 0.3)$ distribution, we use the following commands:

```
>x <- c(0:10)
>n <- 10
>p <- 0.3
>pbinom(x,n,p)
```

(with x acting as q in the specification for convenience) for which the response is

```
[1] 0.02824752 0.14930835 0.38278279 0.64961072 0.84973167 0.95265101
[7] 0.98940792 0.99840961 0.99985631 0.99999410 1.00000000
```

Again, this calculation has created a vector x of the integers from 0 to 10, then specified $n = 10$, then specified $p = 0.3$, and then evaluated the cdf function at each value in the vector. That is we have evaluated

$$F_X(0) = \mathrm{P}[X \leq 0] = \mathrm{P}[X = 0]$$

$$F_X(1) = \mathrm{P}[X \leq 1] = \mathrm{P}[X = 0] + \mathrm{P}[X = 1]$$

and so on. Again, we assign the cumulative probabilities to a vector, and plot the resulting function:

```
>y <- pbinom(x,n,p)
>plot(x,y)
```

for which the response is a point plot of the cdf.

The p- functions for the various distributions, for example,

```
pbinom, pgeom, pnbinom, ppois, pexp, pgamma, pnorm, ...
```

and so on all have this same basic syntax - the only difference is that the parameters for each distribution change. For continuous distributions, we proceed as above

```
>x <- c(0:1000)/100
>y <- pgamma(x,2,2)
>plot(x,y,type=''l'')
```

produces a plot of the required cdf

Again, for the different probability models, you need to use the **help** command to find out the precise syntax and parameter specification for each distribution.

4. Inverse Distribution function calculations.

Again we begin with the Binomial model, for which the inverse cumulative distribution function, that is the function that solves the equation

$$F_X(x) = p_0$$

for x with p_0 fixed, is obtained using the function

```
qbinom(p, size, prob)
```

where $p \equiv p_0$ is the probability at which we wish to evaluate the inverse cdf. This calculation is very important in many statistical problems. To evaluate the inverse cdf in a $Binomial\,(10, 0.3)$ distribution, we use the following commands - we will use x to replace p as the argument of the function, to avoid confusion:

```
>p0 <- 0.265
>n <- 10
>p <- 0.3
>x_qbinom(p0,n,p)
>x
```

for which the response is

```
[1] 2
```

Again, we can use a vector argument to this function, solve for x, and plot the resulting function:

```
>p0 <- c(1:100)/100
>x <- qbinom(p0,n,p)
>plot(p0,x)
>plot(x,p0)
```

The q- functions for the various distributions, for example,

```
qbinom, qgeom, qnbinom, qpois, qexp, qgamma, qnorm, ...
```

and so on all have this same basic syntax but slightly different parameter specifications. For continuous distributions, we proceed as above for the $Gamma(2, 2)$ distribution:

```
>p0 <- c(0:1000)/1000
>x <- qgamma(p0,2,2)
>plot(p0,x,type=''l'')
>plot(x,p0,type=''l'')
```

produces a plot of the required inverse cdf

For the different probability models, you need to use the **help** command to find out the precise syntax and parameter specification for each distribution.

5. Random number simulation

It is often useful to be able to generate a random sample from a given probability distribution. Concentrating first on the $Binomial(10, 0.3)$ we use the function

```
rbinom(n, size, prob)
```

where n is the required simulated sample size. To simulate a sample of size 500 from this Binomial model and store it in vector x, and then to plot a histogram of this simulated data, we can issue the following commands:

```
>n <- 10
>p <- 0.3
>x <- rbinom(500,n,p)
>hist(x)
```

which produces a histogram. We can change the number and/or positions of bars or "bins" in the histogram by using the commands

```
>hist(x,nclass=5)
>hist(x,breaks=c(0:10))
```

The r- functions for the various distributions, for example,

```
rbinom, rgeom, rnbinom, rpois, rexp, rgamma, rnorm, ...
```

and so on all have this same basic syntax but slightly different parameter specifications. For continuous distributions, we proceed identically to the discrete case: for the $Gamma(2, 2)$ distribution:

```
>x <- rgamma(500,2,2)
>hist(x)
>hist(x,nclass=20)
```

For the different probability models, you need to use the **help** command to find out the precise syntax and parameter specification for each distribution.

6. Tranformations

For simulated data, generating a "transformed" sample is straightforward. If we wish to generate a sample from a continuous $Uniform(0, 1)$ distribution, and then to transform it using a $-\log$ transformation, we can proceed as follows:

```
>x1 <- runif(5000,0,1)
>hist(x1)
>x2 <- -log(x1)
>hist(x2)
```

EXERCISES:

1. Evaluate $\mathrm{P}[X = 5]$ if $X \sim Geometric(0.6)$ (note: take care with the parameterization - for example check $\mathrm{P}[X = 0]$ and compare this with the parameterization given in lecture notes; the S-PLUS Geometric functions are based on the mass function

$$(1-\theta)^x\,\theta \qquad\qquad x = 0, 1, 2, ...$$

which is a slightly different model to ours)
2. Evaluate $\mathrm{P}[X = 15]$ if $X \sim Poisson(9)$
3. Evaluate $\mathrm{P}[X \leq 12]$ if $X \sim Binomial(20, 0.6)$
4. Evaluate $\mathrm{P}[X > 20]$ if $X \sim Poisson(15)$
5. Evaluate $\mathrm{P}[30 < X \leq 45]$ if $X \sim Binomial(100, 0.35)$
6. Plot the pmf of the $Poisson(8)$ distribution on the range $0 \leq x \leq 20$
7. Plot the pdf of the $Gamma(5, 2)$ distribution on the range $0 \leq x \leq 10$
8. Plot the pdf of the $Normal(-5, 5^2)$ distribution on the range $-20 \leq x \leq 20$
9. Plot the cdf of the $Normal(0, 1)$ distribution on the range $-3 \leq x \leq 3$
10. Produce a sample of 5000 values from a $Normal(0, 1)$ distribution, plot a histogram, and then plot a histogram of the squares of these values.

It is also possible to generate a random sequence that is similar to a biological sequence using the S-PLUS function **sample**: we proceed by issung the following command:

```
>bases <- c(''A'',''C'',''G'',''T'')
>pvec <- c(0.25,0.25,0.25,0.25)
>x <- sample(bases,size=50,replace=T,prob=pvec)
>x
```

that will produce (something like) the following output

```
[1] ''C'' ''A'' ''T'' ''G'' ''A'' ''G'' ''C'' ''C'' ''A'' ''A''
[11]''G'' ''G'' ''C'' ''T'' ''C'' ''T'' ''C'' ''C'' ''C'' ''C''
[21]''G'' ''C'' ''T'' ''A'' ''A'' ''T'' ''C'' ''G'' ''T'' ''G''
[31]''C'' ''A'' ''C'' ''A'' ''A'' ''G'' ''A'' ''T'' ''C'' ''A''
[41]''C'' ''T'' ''T'' ''G'' ''A'' ''G'' ''C'' ''G'' ''C'' ''T''
```

The commands created some base labels A, C, G and T, and then a probability for each label (in this case the probability is 0.25 for each label), and then produced a sample of size 50 independently sampled from this distribution.

The **prob** vector determines how probable each label is; in nature, it is unlikely that each base is observed with equal probability, and also that the base sequence is not independentlt generated (that is, the base observed at one position is influenced by bases observed in previous positions). In light of the analysis carried out last week for the BRCA2 sequence, how could a more realistic biological sequence be generated ?