Elements of R

1 Arithmetic

The expressions $+, -, *, /$ are used in the usual way. Exponents are indicated by expressions like $3 \wedge 4$, which evaluates to 81. There are various common functions that work like sqrt(9) and abs(-4).

2 Logic

Equality is expressed by $==$. Lack of equality is $!=$. The inequalities are $<, <=$ and $>, >=$. The logical operations and, or, not are written &, $|$, !.

3 Vectors

A vector can be generated by c(5, 2, 4). This combines the numbers 5, 2, 4 to form a single vector. The vector 2:5 is the same as the vector c(2,3,4,5). The vector seq(2,5, 0.1) is the same as the vector 20:50/10.

4 Assignment

A variable is assigned a value by the command

variable <- expression

Thus, for instance

x <- c(5,2,4)

makes x stand for the corresponding vector. In this context we can say x "becomes" c(5,2,4).

5 Vector operations

If x is a vector, then length(x) tells how many components it has, and x[3] selects the third component.

The sum of the components is sum(x), and the mean is mean(x). This is the same as sum(x)/length(x). The variance var(x) is defined with the $n - 1$ factor in the denominator. The standard deviation is sd(x).

The largest and smallest elements of a vector are given by max(x) and min(x). The expression sort(x) gives a vector with the same entries, but sorted in increasing order. The expression median(x) gives the same result as quantile(x, 0.5). The quartiles can be obtained by quantile(x, 0.25, 0.5, 0.75)

With two vectors of the same length one can compute the correlation coefficient cor(x,y). The two vectors can be plotted by plot(x,y).

6 Functions

A function is denoted by giving inputs and an expression for an output. Thus

function (x) $x * (1 - x)$

denotes a function that takes input x and gives output $x(1-x)$. If we wanted to give this function a name, such as h, then we would make the assignment

h <- function $(x)x * (1 - x)$.

Thus h(2) would return -2.

7 Probability distributions

For each probability distribution there are three functions and one random sample generator. Thus for the normal distribution these are:

dnorm(x,mean,sd) density: computes density as a function of x

pnorm(q,mean,sd) distribution: computes probability as a function of quantile q

qnorm(p,mean,sd) inverse distribution: computes quantile as a function of probability p

rnorm(n,mean,sd) generates random sample of size n

Similarly, for the binomial distribution there are the functions dbinom(x,size,prob), pbinom(q,size,prob), qbinom(p,size,prob), and rbinom(n,size,prob).

Here are some of the probability distributions that are commonly used. The following listing has the p version of the function, but the d,p,q,and r versions all exist.

pnorm(q,mean,sd) normal distribution

pgamma(q,shape,rate) Gamma distribution

pexp(q,rate) exponential distribution: same as pgamma(x,1,rate)

pchisq(q,df) chi square distribution: same as pgamma(x,df/2,1/2)

pt(q,df) t distribution

pf(q,df1,df2) F distribution

pbeta(q,shape1,shape2) Beta distribution

punif(q,min,max) uniform distribution

pcauchy(q,location,scale) Cauchy distribution

pbinom(q,size,prob) binomial distribution

pnbinom(q,size,prob) negative binomial distribution

pgeom(q,prob) geometric distribution: same as pnbinom(q,1,prob)

ppois(q,lambda) Poisson distribution

8 Example: Empirical distribution

Take a sample; tabulate the results.

Create a sample:

```
x <- rbinom(100,8,1/2)
```

Create a vector:

```
n <- 0:8
```

Tabulate the sample:

```
for(i in 1:9) n[i] <- sum[ x == i-1 ]
```

Plot the table:

```
plot(0:8,n)
```

9 Example: The Bernoulli process

Compare the number of successes up to n with the time of the ith success.

Take an independent Bernoulli sample:

```
x <- rbinom(100,1,1/7)
```

Create a vector:

```
s <- 1:100
```

Find the number of successes in the first n trials:

```
h <- 1:100
for( n in 1:100) s[n] <- sum( x[ h <= n])
```

Create another vector:

```
t <- 1:100
```

Find the time of the ith success:

```
for(i in 1:100) t[i] <- min ( h [ s >= i])
```

Extract the useful part of this vector:

```
tt <- t[1:13]
```

10 File input

To read in a vector:

```
x <- scan(”filename.txt”)
```

To read in a list of two vectors:

```
xy <- scan(”filename.txt”, list(0,0))
```

To extract the individual vectors:

```
x <- xy[[1]]
y <- xy[[2]]
```