School of Mathematical and Computer Sciences

Department of Actuarial Mathematics and Statistics

DATA ANALYSIS

R reference sheets – prepared by and copyright © Roger J.Gray

There are spaces at the foot of some pages for you to record any additional notes

💻 For more information use the extensive *R* help facility e.g. ? plot

Entering data into one-dimensional vectors (in the Commands Window)

> a = 5	← *a single number (a scalar)*
> b = 5:9	← *a simple consecutive sequence 5 6 7 8 9*
> n = c(32, 35, 39, 42, 47, 61)	← *a vector of values of a quantitative variable*
> age = c(12, 13, 14, 15, 16, 17, 18)	← *a vector of values of a quantitative variable*
> age = c(12:18)	← *same as above*
> age2 = c(age, 19)	← *a vector of 8 numeric values*
[redacted]	[redacted]
> agegroupf = factor(c(1, 2, 3, 1, 2, 3))	← *a vector of codes specifying a discrete classification or grouping of components of other vectors*
> gender = c("M", "F", "M", "F", "M", "F")	← *a vector of 6 "character values"*
> genderf = factor(gender)	← *a factor version of the vector gender*

Arithmetic/functions/operations – various illustrations

For a,b,c,d vectors of appropriate lengths:

> a = b + 4   > a = b*c   > a = b*c/sum(d)

> a = sum(b^2/c)   > a = sqrt(b)   > a = log(b)   > pi = exp(d)/(1 + exp(d))

> a = choose(12,5)   ← $\binom{12}{5} = 792$ > a = gamma(6)   ← *5! = 120*

> a = cumsum(1:6)   ← *a is the vector 1 3 6 10 15 21*

> a = cumprod(1:5)   ← *a is the vector 1 2 6 24 120*

Listing objects in use

> ls( ) or > objects( )

Listing and summarising contents of vectors and other objects

```
> age                          ← returns the contents of the vector
> summary(n)                   ← descriptive summary
> length(n)                    ← number of values
> table(n)                     ← frequency distribution of values
> mean(age)   > sd(age)   > median(age)   > max(age)
> quantile(claim)        ← 0%, 25%, 50%, 75%, 100%  quantiles (min, lower quartile, median,
                           upper quartile, max : similar info to "summary" but without the mean)
> quantile(claim, 0.9)         ← 90% quantile (10% of values above it)
> quantile(claim, c(0.25, 0.75))  ← lower and upper quartiles
> cor(weight, height)          ← correlation coefficient

> a = sort(b)                  ← sort into increasing order

> levels(agegroupf)            ← returns the levels of the factor agegroupf
> names(L)               ← give the names of items in a  list or fitted model L :  see ?names

> a = age[4]             ← a is the 4th element of vector age
> agenew = age[26:50]    ← agenew is a vector containing elements 26 to 50 of vector age
> big = claim[claim > 5] ← big contains all elements of claim which exceed 5
> pick = b[a<3]          ← pick contains the elements of b for which the corresponding elements of a
                           are less than 3 : try it with a=c(4,7,1,8,2,5) and b=c(40,70,10,80,20,50)
> pick2 = b[a==2]        ← pick2 contains the elements of b for which the corresponding elements of a
                           are equal to 2
```

Editing vectors

```
> fix(age)               ← opens the vector "age" in a text editor - on exiting saves the changes
```

Plotting data

```
> plot(age)                   ← basic plot, with age on y-axis, against an index
> plot(age, n1)               ← basic scatter plot, with age on x-axis
> plot(age, n1, pch = 16)     ← pch = plotting character (number 16 is a solid circle; try other effects)
> plot(age, n1, pch = "M")    ← uses M as plotting character
> plot(age, n1, type = "l")   ← lines connect the data  (try also types "b" and "o")
> plot(age, n1, type = "n")   ← sets up the plotting scales only – no points shown – useful for plots
                                which include two or more sets of points – e.g. set up the plotting scales
                                and then add the points for men and then add those for women
> plot(dur, n, xlim = c(18,30), ylim = c(0,40), ylab = "number of claims", xlab = "age of
          policyholder", main = "Numbers of claims per 100 policies by age of policyholder")
                              ← sets limits on x and y axes plotting scales, labels axes and plot itself
> plot(x2, y2, log = "y")     ← plots using a log scale on the y axis
```

Use this to illustrate plotting characters and colours available:

```
> plot(1:20, pch=1:20, col=1:20)
```

Alternative approach, using a "structure" in place of two vectors (whose names might be duplicated and thus lead to confusion)

```
> plot(wt ~ ht, data = frame4)  ← takes the vectors wt and ht from data frame "frame4" and plots wt on
                                  the vertical axis against ht
> pairs(frame5)                 ← "matrix plot": one scatter plot for each pair of variables in the data
                                  frame "frame5"
```

♦ Adding points, lines, and a "legend" to an existing plot

> points(age, n2, pch = 8) ← *plotting character (number 8 is an asterisk; try various effects)*
> lines(age, n3, lty = 2) ← *lty = line type (number 2 is a dashed line; try various types)*
> abline(a,b) ← *adds line with intercept a and slope b to current plot*

Use this to illustrate the adding of points and a legend to a plot:
> x=1:20
> plot(x,x)
> points(x,sqrt(x),pch=2,col=3)
> points(x,log(x),pch=3,col=4)
> legend(2.5,15,legend=c("x", "sqrt(x)", "log(x)"), pch=c(1,2,3), col=c(1,3,4))

Some displays (many options available)

> stem(claim) > stem(claim, scale=5)
> hist(claim) > hist(claim, seq(0,6000,300), prob=T) > hist(claim, breaks=25)
> lines(density(claim, bw=150)) >rug(claim)
> boxplot(claim) > boxplot(claim, horizontal=T)
> plot(density(claim))
> qqnorm(claim) > qqline(claim)
> plot(income) ← *plot of income against an index 1,2, …, ; basic time series plot*
> plot(age, n) > plot (1:50, sales) ← *scatter plots : see more on plotting below*

Matrices and arrays

> m1 = matrix(c(19, 497, 29, 560, 24, 269), 2, 3) ← *a 2×3 matrix, entries read in by column*
> m2 = matrix(c(19, 497, 29, 560, 24, 269), 2, 3, byrow = T)
← *a 2×3 matrix, entries read in by row*
> m3 = matrix(c(5, 7), 2, 6) ← *a 2×6 matrix with 6 identical columns*
> m4 = cbind(n1,n2) ← *creates a k×2 matrix, with columns n1 and n2, where n1 and n2 are both of length k*

> arr1 = array(a1, c(4, 2, 3)) ← *creates an array of dim 4×2×3, where a1 is a vector of length 24: produces 3 matrices of order 4×2; try array(1:24, c(4,2,3))*
> b1 = as.vector(arr1) ← *returns the contents of the array arr1 as a vector of length 24; here b1 is a copy of a1*
> b2 = c(arr1) ← *same effect as using as.vector*

> m2 = t(m1) ← *matrix m2 is the transpose of matrix m1*
> m3 = m1 %*% m2 ← *matrix multiplication*
> m3 = m1*m2 ← *elementwise multiplication within two matrices*
> m4 = diag(1:6) > m4 = diag(x) ← *matrix m4 is a square matrix with diagonal elements 1 to 6 or the elements in the vector x*
> b = diag(m1) ← *b is the vector containing the diagonal elements of matrix m1*
> m4 = solve(m3) ← *m4 is the inverse of square matrix m3*

For m1 a matrix or arr1 an array of order r×s:

> a = sum(m1) ← *sum of all rs entries in m1*
> rsum = apply(m1, 1, sum) ← *vector of r row sums of m1*
> cmean = apply(arr1, 2, mean) ← *vector of s column means of array arr1*

Patterned data (using replicates and sequences)

> age = c(12:20)	← *vector "age" contains integers from 12 to 20*
> a = rep(1,6)	← *vector "a" contains 1 1 1 1 1 1*
> b = rep(1:3,2)	← *vector "b" contains 1 2 3 1 2 3*
> c = rep(1:3, each = 2)	← *vector "c" contains 1 1 2 2 3 3*
> rep(1:3, each = 2)	← *returns* 1 1 2 2 3 3
> evens = seq(4, 12, 2)	← *vector "evens" contains 4 6 8 10 12*
> rc = factor(c(rep(1:4, each = 3)))	← *reads in row codes for a 4×3 table read in row by row*
> cc = factor(c(rep(1:4, 3)))	← *reads in col codes for a 4×3 table read in row by row*

Simulation: generating random observations

> s1 = rnorm(100)   > s2 = rnorm(50, 10, 2)
← *random samples s1: 100 obs from N(0,1) ; s2: 50 from* $N(10,2^2)$
> s3 = rpois(200, 2)   > s4 =rbinom(40,12,0.4)
← *s3: 200 from Poisson(2) ; s4 : 40 from binomial(12,0.4)*
> s5 = rexp(100,0.1) ← *s5: 100 obs from exponential* $\lambda = 0.1$*, mean = 10*
> s6 = rnbinom(500,4,0.6) ← *s6: 500 obs from negative binomial with range x = 0,1,2,… , k=4, p=0.6, mean kq/p = 8/3*

Other distributions available include beta (beta), chi-squared (chisq), gamma (gamma), geometric (geom., or nbinom with k =1), uniform (unif)

Cdf (and hence P-values), quantiles (inverse cdf)

> pnorm(1.5)	← *cdf* $P(X < 1.5)$ *for* $X \sim N(0,1)$
> pnorm(13, 10, 2)	← *cdf* $P(X < 13)$ *for* $X \sim N(10,2^2)$
> qnorm(0.95)	← *value for x for which* $P(X < x) = 0.95$ *for* $X \sim N(0,1)$
> qnorm(0.9, 12, 3)	← *value of x for which* $P(X < x) = 0.9$ *for* $X \sim N(12, 3^2)$
> ppois(5, 2)	← *cdf* $P(X \le 5)$ *for* $X \sim P(\lambda = 2)$
> pbinom(12, 20, 0.6)	← *cdf* $P(X \le 15)$ *for* $X \sim binom(n = 20, p = 0.6)$
> pchisq (4.7, 2)	← *cdf* $P(X < 4.7)$ *for* $X \sim \chi^2$ *with 2df*

Tests of association in a two-way table

> chisq.test(m1)	← *m1 is a matrix of frequencies – chi-squared test*
> fisher.test(m1)	← *m1 is a matrix of frequencies – exact test*

Importing data from files

♦ *Reading data into a single vector*

> rate = scan("h:/intrates.txt") ← *reads a column of data in a text file held in directory h into a vector*

♦ *Reading data into a single vector direct from the web*

> rate = scan("http://www.ma.hw.ac.uk/~roger/f73sj2/data/intrates.txt")
← *reads a column of data in a text file after downloading it from the web*

♦ *Reading data into a data frame*

> claims = read.table("claimsdata.txt") ← *reads a text file containing 2 or more columns of values of variables (numerical, factor codes) of the same length, either (i) with row labels or numbers and a header row (variable names in the first row of the file), or (ii) with no row labels/numbers and no header row, into a data frame*

> claims2 = read.table("h:/project4/ecology.txt", col.names = c("year", "conc", "depth", "type"))

> claims3 = read.table("racestats", header=TRUE) ← *reads a text file containing 2 or more columns of values of variables (numerical, factor codes) of the same length, with no row labels/numbers but with a header row, into a data frame*

♦ *Reading data into a data frame direct from the web*

> claims = read.table("http://www.ma.hw.ac.uk/~roger/f73sj2/data/claims.txt")
← *reads a text file into a data frame after downloading the file from the web*

♦ ***Accessing built-in datsets***

R: Over 50 datsets are supplied and others are available in libraries
> data( ) ← *lists the datasets supplied and available for use – thay are in a package called "base"*
> data(morley) ← *loads the dataset "morley", which is a data frame containing 100 observations of 3 variables*
To access other libraries and data sets use
Packages menu → Load package then highlight the one you want (e.g. MASS) and double-click

> data(package=MASS) ← *lists the data sets, which include a 26x6 data frame called road*
> data(road)
> road

Using data frames

♦ *Creating a data frame from keyboard*

> temp = data.frame(a, b, c) ← *creates data frame temp with 3 columns from vectors of equal length a, b, and c*
> tax = data.frame(mat1) ← *creates data frame tax containing the elements of matrix mat1*

> frame6 = cbind(a,b,c,d) ← *creates data frame (with 4 columns) – different effect*

♦ *Making data available outside a data frame*

> yield$size ← *extracts the column with variable name size from data frame yield*
> sizenew = yield$size ← *as above and defines it as a new object*
> duration = claims[,5] ← *duration is a vector comprising the 5th column of the data frame claims*
> attach(claims) ← *enables all variables in data frame claims to be used outside the data frame (ensure there are no other variables of the same names in existence)*
> detach(claims) ← *reverses the effect of attach – variables now not available outside data frame*

♦ *Opening/editing an existing data frame*

To view the contents of a data frame (or other object), just type name of object at the prompt and enter it.

To <u>edit</u> the contents of a data frame (including changing names and types of variables):

Use menus: *Edit* → *Data editor* or

> fix(claims) or > edit.data.frame(claims) ← *opens the frame in an editor window*

Clicking on a variable name allows you to edit it.

Using simple functions – three examples

If you type the function outside ***R*** in a simple text editor you should store the resulting text file (say file func1.txt) in the directory which contains your ***R*** workspace (the ***R*** image – the file of the form *.RData) and then load it using the ***R*** command "source", e.g. funcA=source("func1.txt")

♦ function fA calculates $2i + i^2$ for i = 1,2, … n for a specified n; first type in the function line by line as follows (***R*** will supply a "+" at the start of each line – or type it externally as a text file and source it)

```
>fA=function(n){
      a=rep(0,n)
      for(i in 1:n){
      a[i]=2*i + i*i
      }
      a
   }
```

then issue

```
> b=fA(5)
> b
```

producing output [1] 3 8 15 24 35

- function fB calculates the means of *n* samples of 200 observations simulated from an exponential distribution with mean 10; the output is a vector of those means which lie between 9 and 11
- function fC calculates the means of *n* samples of 200 observations simulated from a Pareto distribution with parameters $\alpha = 3$ and $\lambda = 20$ (and so with mean 10); the output is a vector of those means which lie between 9 and 11

```
>  fB=function(n){
e=rep(0,n)
for (i in (1:n)){
c=rexp(200,0.1)
d=mean(c)
e[i]=d
f=sort(e)
g=f[f>9]
h=g[g<11]
}
h
}

> m1=fB(100)
> m1
```

```
>  fC=function(n){
b= -1/3
e=rep(0,n)
for (i in (1:n)){
a=runif(200)
c=20*(a^b – 1)
d=mean(c)
e[i]=d
f=sort(e)
g=f[f>9]
h=g[g<11]
}
h
}

> m2=fC(100)
> m2
```

Statistical modelling

If the model includes one or more qualitative factors, see **Contrasts for factors** below before fitting the model.

Linear models

> model1 = lm(y ~ x) ← *normal linear regression model of y on x*

> model2 = lm(y1~x1+x2, data = illus3) ← *normal linear regression model of y1 on x1 and x2, data held in data frame "illus3"*

- *Information from fitted models*

> summary(model3) ← *displays parameter estimates and st. errors, deviance, and correlation matrix*

>summary.aov(model5) ← *displays the analysis of variance for the fitted model*

> fitted(model4) > resid(model4) > coef(model4)

> f4 = fitted(model4) ← *vectors containing fitted values, residuals, coefficients in fitted model*
> r2 = resid(model2)
> c3 = coef(model3)

> plot(fitted(model3), resid(model3)) ← *plot of residuals against fitted values*
> abline(model3) ← *adds fitted line to current data plot*
> abline(h=0, lty=2) ← *adds a horizontal dashed line at y = 0 to current data plot*

> plot(model3) ← *supplies 4 plots associated with the fitted model "model3": click on the command window (and then return) each time to get each plot;*
1 residuals v fitted, 2 normal Q-Q plot,
3 scale-location plot, and 4 Cook's distance plot

Generalised linear models

Log linear models

> model2 = glm(n ~ rc + cc, family = poisson)
> model3 = glm(n ~ age, family = poisson)
> model4 = glm(n ~ attitude + age + gender, family = poisson)
> model5 = glm(n ~ attitude*age + gender, family = poisson))
> model6 = glm(n ~ attitude*age*gender, family = poisson)

Logistic regression models

> model7 = glm(propdead ~ dose + age, weights = groupsize, family = binomial)
> model8 = glm(propdead ~ dose*age, weights = groupsize, family = binomial)

Contrasts for factors

This refers to the parameterisation which contrasts the response at each level of a factor with that of the first level of the factor. There are several possible ways to set the parameterisation.

In ***R*** the default setting is the "treatment setting". In this case the parameter values given are the additions required for the second, third, … levels of the factor (the first level for each factor being the "base" or "reference" level). This is convenient and easy to understand.