

# Unlock the Secrets of R



G. Janacek

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.0.1	Why $\mathbb{R}$ . . . . .	3
1.0.2	Large Data Sets . . . . .	3
1.0.3	Other programs . . . . .	3
1.0.4	Before you start . . . . .	4
<b>2</b>	<b>Getting Up and Running</b>	<b>4</b>
2.0.5	Startup and shut down . . . . .	4
2.1	Manipulating data . . . . .	4
<b>3</b>	<b>Help</b>	<b>6</b>
<b>4</b>	<b>Vectors and Matrices</b>	<b>8</b>
4.0.1	Slices and elements . . . . .	11
4.1	Data and data input . . . . .	11
4.1.1	Missing values . . . . .	12
4.1.2	scan . . . . .	12
4.1.3	Data tables - the dataframe . . . . .	13
4.2	Reading data files . . . . .	14
4.2.1	Excel files . . . . .	16
4.3	Graphics . . . . .	16
4.4	Simple Stats . . . . .	19
<b>5</b>	<b>Regression</b>	<b>21</b>
5.0.1	Sales . . . . .	21
5.0.2	Babies . . . . .	24
5.0.3	Fixed effects ANOVA . . . . .	29
<b>6</b>	<b>Introduction</b>	<b>32</b>
6.1	Principal Components . . . . .	35
6.1.1	Reducing Dimension . . . . .	36
6.1.2	Summary . . . . .	38
6.1.3	Swiss Bank Notes . . . . .	38
<b>7</b>	<b>MDS</b>	<b>41</b>
<b>8</b>	<b>GLIMS</b>	<b>42</b>
8.0.4	Toxicity of the Tobacco budworm . . . . .	42
<b>9</b>	<b>Reshapeing data</b>	<b>46</b>
9.1	An Ecological example . . . . .	48
<b>10</b>	<b>Exercises</b>	<b>55</b>
<b>11</b>	<b>Fitting Linear Models: lm R Documentation</b>	<b>57</b>
11.1	Description . . . . .	57
11.1.1	Usage . . . . .	57
11.1.2	Arguments . . . . .	57
11.1.3	Details . . . . .	58

11.1.4	Value . . . . .	58
11.1.5	Using time series . . . . .	59
11.1.6	Note . . . . .	59
11.1.7	Author(s) . . . . .	59
11.1.8	Examples . . . . .	59
<b>12</b>	<b>Fitting Generalized Linear Models: glm stats R Documentation</b>	<b>60</b>
12.0.9	Description . . . . .	60
12.0.10	Usage . . . . .	60
12.0.11	Arguments . . . . .	60
12.0.12	Details . . . . .	61
12.0.13	Value . . . . .	62
12.0.14	Author(s) . . . . .	63
12.0.15	Examples . . . . .	63
<b>13</b>	<b>Random Number generators</b>	<b>64</b>
<b>14</b>	<b>Publications related to R - from CRAN</b>	<b>65</b>

## 1 Introduction

There are three main kinds of statistics program available at UEA. They are

- SPSS - a program supported by the ITSC. It was designed to support the kinds of data analysis that you might need in the social sciences. It is fine for routine analysis.
- Stata - a complete, integrated statistical package which is popular amongst medics.
- $\mathbb{R}$  which is (almost) a free clone of Splus, a program which implements the S language, see Becker and Chambers (1981).

In what follows we aim to get you up and running with  $\mathbb{R}$ . *We will not be teaching you statistics or the details of Lisp*, our aim is to get you some way along the learning curve so that you can start to use  $\mathbb{R}$  for your own analysis.

### 1.0.1 Why $\mathbb{R}$

$\mathbb{R}$  is used by statistics departments and researchers around the world because it is an open source implementation of Splus. It allows the user to be flexible in their analysis and is much more up to date than SPSS. It has extensive and powerful graphics abilities, that are tightly linked with its analytic abilities.  $\mathbb{R}$  is also developing rapidly and new features and abilities appear every few months. It is available on most machines at UEA and you can download your own copy to run on a personal machine.

**$\mathbb{R}$  is free and I would urge you to get your own copy.**

Binary versions and the source code (if you are very brave) for  $\mathbb{R}$  are available from the CRAN website. You will also find comprehensive documentation, Web Pages and Email Lists. Go to

<http://cran.r-project.org>

Details of the  $\mathbb{R}$ -help list, and of other lists that serve the  $\mathbb{R}$  community, are available from the web site for the  $\mathbb{R}$  project at

<http://www.R-project.org/>

Sadly there is no support for  $\mathbb{R}$  at UEA, which is why I am writing these notes. There is a *wiki* at

<http://cmpwiki/wiki/Pearson>

but I have only just set it up so the content is thin. I have asked ITSC to set up a mailing list, probably called Capital.R. You will be able to subscribe to the list and exchange problems and ideas with other subscribers.

### 1.0.2 Large Data Sets

One important point you should bear in mind is that  $\mathbb{R}$  was not designed for very large data sets. The consensus is that on the same modern system SAS is usually better able to handle large, dumb calculations than S-PLUS, which is (generally) better than  $\mathbb{R}$ .  $\mathbb{R}$  has a hard limit of 2 GB total memory, as I understand, and its data model requires holding an entire set in memory. This is very fast until it isn't. This limit applies even on 64 bit systems. Horses for courses!

### 1.0.3 Other programs

There are other programs which will link to R and use R as a compute engine or a front end. These include Ggobi, Mondrian and Rbugs.

### 1.0.4 Before you start

$\mathbb{R}$  using the computer file system. It creates a workspace to do the computations that you require and when you quit  $\mathbb{R}$  this may be saved. It makes sense to create a working directory for each project and do all your work within that directory.

## 2 Getting Up and Running

$\mathbb{R}$  is a functional language, that is there is a language core that uses standard forms of algebraic notation, allowing the calculations such as  $2 + 3$ , or  $3^{11}$ . If you type  $3+4-11$  then the answer is printed. However if you assign this to a variable  $w<-3+4-11$  nothing is printed. To see the value type

`w`

Beyond this, most computation is handled using functions. Rather than go on about structure and syntax we will start to use  $\mathbb{R}$ .

### 2.0.5 Startup and shut down

Start up  $\mathbb{R}$ , either by clicking on the ikon (PC and Mac) or typing R on a UNIX box, and you will get the welcome screen. What you will have loaded is the *base* package. You can load other packages that come with your installation or from elsewhere, see the header of the  $\mathbb{R}$  console window but for the moment we work with the base package.

The action of quitting from an  $\mathbb{R}$  session uses the function call `q()`. When you use  $\mathbb{R}$  you create a workspace in which all the objects you have created are stored. When you try and quit you will be asked if you wish to save your (workspace) session.

If you use a PC then you may be able to quit via the pull down menu on your screen.

## 2.1 Manipulating data

Most of the data sets in **data** are structured in some way so, for the moment, we will try something simpler to get us going.  $\mathbb{R}$  has lots of ways of simulating data from given distributions, for example

```
rnorm(n, mean=0, sd=1)
```

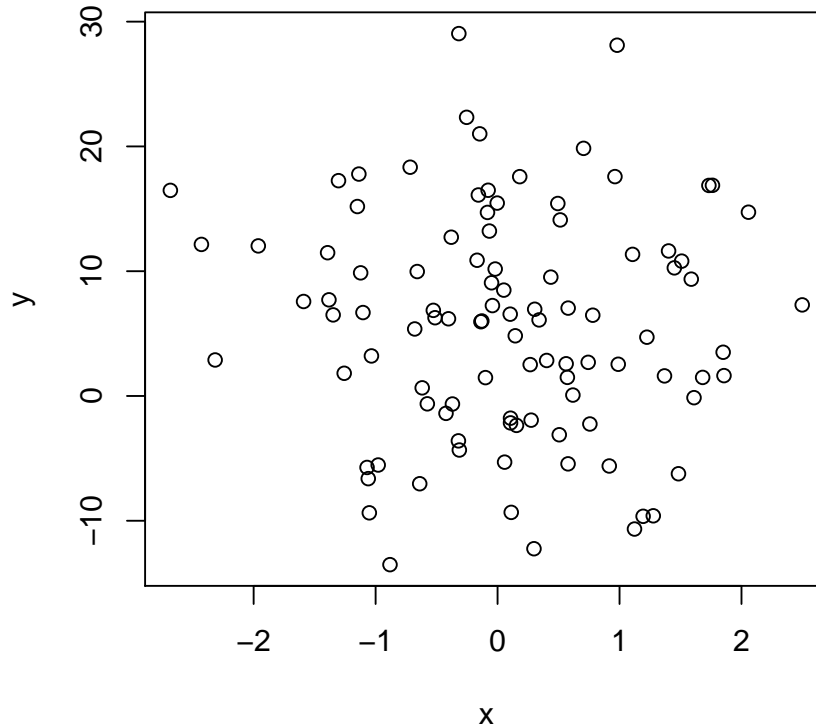
This command gives a vector of length `n` whose elements are random and have a Normal distribution mean zero and standard deviation 1. If you do not specify the `mean` and `sd` the values default to 0 and 1. So to create two `x` and `y` vectors of length 100

```
> x<-rnorm(100)
> y<-rnorm(100,5,9)
```

if you type `x` you can examine the numbers in `x`. If 100 is too many numbers try `x[20:50]` which gives a slice of the vector `x`. You can use the indices in many useful ways, e.g. `x[100:87]`

**Note we use a left pointing arrow as the assignment operator.** For versions of R after 1.4 you can, if you prefer, use `=`. I prefer to use `←` for compatability.

To check the random numbers we could plot `x` and `y`. Try `?plot` or `help(plot)` This will explain the plotting command. You could try `plot(x,y,pch="+")` or `plot(x,y,pch="a")`



The obvious data summaries are the mean, median, standard deviation etc. try some of these functions

```
hist(x) # computes and plots a histogram of x
mean(x) # computes the mean of the vector x
sd(x) # computes the standard deviation of the vector x
median(x) # computes the median of the vector x
summary(x) # summarizes the vector x
print(x) # prints the object x
cat() # Prints multiple objects, one after the other
length() # Number of elements in a vector or of a list
unique() # Gives the vector of distinct values
diff() # Replace a vector by the vector of first differences
#N. B. diff(x) has one less element than x
sort() # Sort elements into order, but omitting NAs
order() # x[order(x)] orders elements of x, with NAs last
cumsum(x) # computes the cumulative sum of the vector x
cumprod(x) # computes the cumulative product of the vector x
rev() # reverse the order of vector elements
# is - as you have probably guessed - a comment symbol. Everything to the right of # is ignored
```

### 3 Help

You are probably beginning to see a problem in using  $\mathbb{R}$ , *you have to know the name* of the function you would like to use. While there are some prototype GUI interfaces you will have to resign yourself to the UNIX command driven world. The appendices contain copies of some of the CRAN crib sheets but in addition the help system can be useful.

*On most current versions of  $\mathbb{R}$  on Windows machines there are pull down windows which help you access the help features.*

- The `apropos()` command is convenient when you are not sure that you know the name of a function, for example if you were after a stem and leaf function but were not sure if the name was `stem` or `stemandleaf`. For example

```
> apropos(stem)
[1] "stem"          "system"        "system.file"  "system.time"
```

- The `help` command will help us find help on the given function or data-set *once we know the name*. For example `help(stem)` or the abbreviated `?stem` will display the documentation on the `stem` function. For example `stem` gives

```
stem {graphics} R Documentation
Stem-and-Leaf Plots
Description
stem produces a stem-and-leaf plot of the values in x.
The parameter scale can be used to expand the scale of the plot.
A value of scale=2 will cause the plot to be roughly twice as long as the default.
Usage
stem(x, scale = 1, width = 80, atom = 1e-08)
Arguments
x a numeric vector.
scale This controls the plot length.
width The desired width of plot.
atom a tolerance.
References
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988)
The New S Language. Wadsworth & Brooks/Cole.
Examples
stem(islands)
stem(log10(islands))
```

- More help is available via a web browser, the command is `help.start()`. See figure 1.

You can follow the hyperlinks to the packages or use the search. In fact Google also works very well.

As a second example suppose we wished to compare the two vectors `x,y` we created. If we are going to perform some sort of test we might try

```
> apropos("test")
[1] ".valueClassTest"      "Box.test"             "PP.test"
```

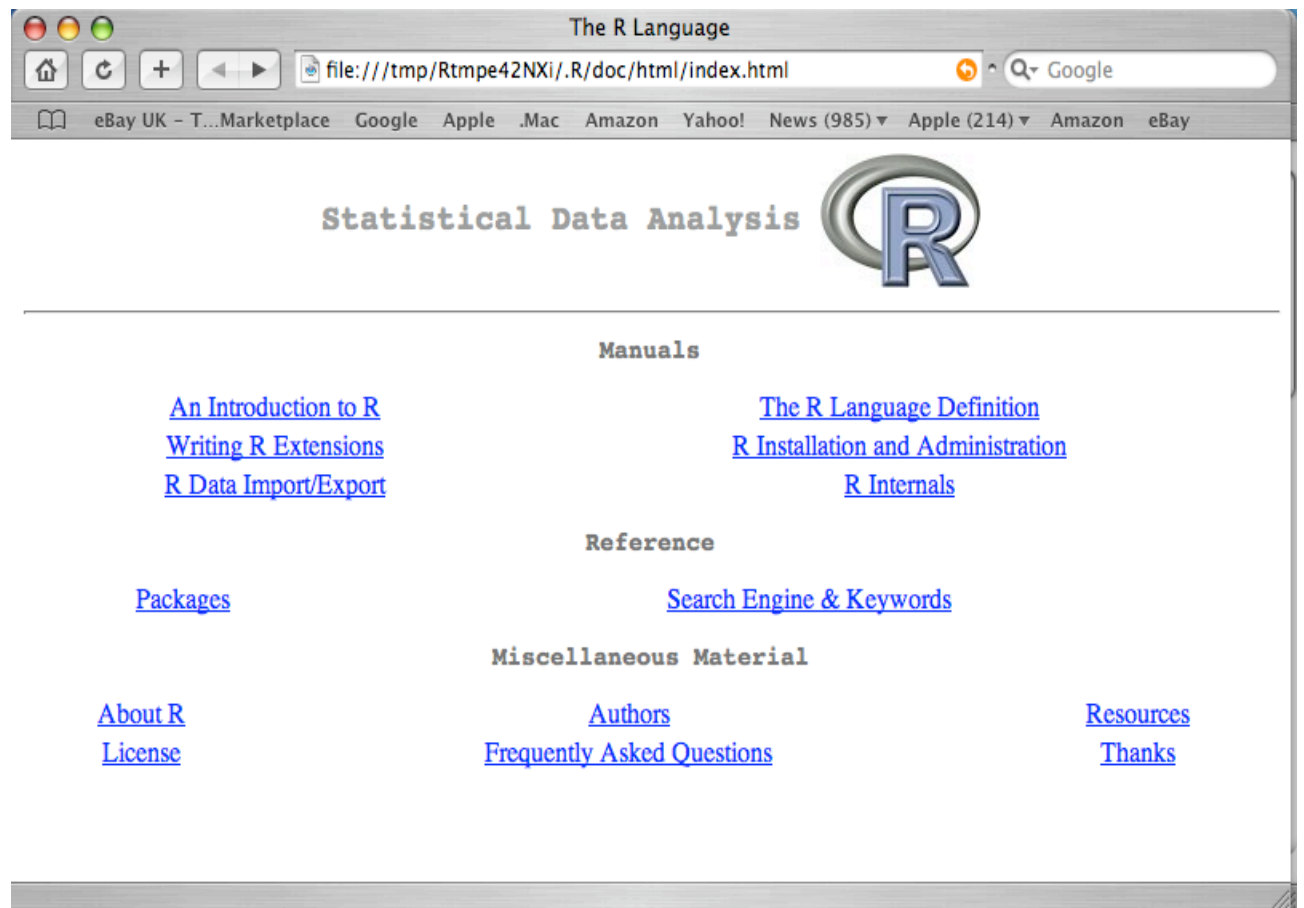


Figure 1: Help screen

```

[4] "ansari.test"          "bartlett.test"      "binom.test"
[7] "chisq.test"           "cor.test"           "file.test"
[10] "fisher.test"          "fligner.test"       "friedman.test"
[13] "kruskal.test"         "ks.test"            "mantelhaen.test"
[16] "mauchley.test"        "mauchly.test"       "mcnemar.test"
[19] "mood.test"            "oneway.test"        "pairwise.prop.test"
[22] "pairwise.t.test"      "pairwise.wilcox.test" "power.anova.test"
[25] "power.prop.test"      "power.t.test"       "prop.test"
[28] "prop.trend.test"      "quade.test"         "shapiro.test"
[31] "t.test"               "testPlatformEquivalence" "testVirtual"
[34] "var.test"             "wilcox.test"

```

I will pick a t test so will try  
t.test then after reading the help file we have

```
> t.test(x,y)
```

Welch Two Sample t-test

```

data:  x and y
t = -4.6575, df = 100.839, p-value = 9.79e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -7.052294 -2.839217
sample estimates:
 mean of x  mean of y
-0.04790487  4.89785017

```

## 4 Vectors and Matrices

Most items in  $\mathbb{R}$  are vectors or lists. A vector is a list of objects of the same type. We can create vectors

```

> p<-1:10 # gives a list
> p
[1] 1 2 3 4 5 6 7 8 9 10
> p[4:9]
[1] 4 5 6 7 8 9
> p[4:1]
[1] 4 3 2 1
> q=-10:-1
> q
[1] -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

```

or

```

> z=seq(1,2,by=0.2)
> z
[1] 1.0 1.2 1.4 1.6 1.8 2.0

```

We can bind vectors together to get matrices

```
m1<-cbind(p,q)
> m1
      p  q
[1,]  1 -10
[2,]  2  -9
[3,]  3  -8
[4,]  4  -7
[5,]  5  -6
[6,]  6  -5
[7,]  7  -4
[8,]  8  -3
[9,]  9  -2
[10,] 10  -1
m2<-rbind(p,q)
> m2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
p       1    2    3    4    5    6    7    8    9   10
q     -10   -9   -8   -7   -6   -5   -4   -3   -2   -1
```

These are proper matrices with dimensions, we can check the dimensions

```
> dim(m1)
[1] 10  2
```

All of matrix algebra is available

```
> m1*2
      p  q
[1,]  2 -20
[2,]  4 -18
[3,]  6 -16
[4,]  8 -14
[5,] 10 -12
[6,] 12 -10
[7,] 14  -8
[8,] 16  -6
[9,] 18  -4
[10,] 20  -2
```

thus

```

> m1-4
      p  q
[1,] -3 -14
[2,] -2 -13
[3,] -1 -12
[4,]  0 -11
[5,]  1 -10
[6,]  2  -9
[7,]  3  -8
[8,]  4  -7
[9,]  5  -6
[10,] 6  -5
> m2%*%m1      ##### notice the percentage signs for matrix multiplication
      p  q
p  385 -220
q -220  385
> m3<-m2%*%m1 ##### notice the percentage signs for matrix multiplication
> solve(m3) ##### solve gives the inverse in this case
#####- it can be used to solve systems of equations
      p  q
p 0.003856749 0.002203857
q 0.002203857 0.003856749
> m4<-solve(m3)
> m4%*%m3
      p  q
p 1.000000e+00 6.678685e-17
q -1.301043e-16 1.000000e+00

> eigen(m3)
$values
[1] 605 165

$vectors
      [,1] [,2]
[1,] 0.7071068 0.7071068
[2,] -0.7071068 0.7071068

```

You can coerce vectors to matrices, for example

```

> p<-1:24
> p2<-p
> dim(p)<-c(4,6)
> p
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  1    5    9   13   17   21
[2,]  2    6   10   14   18   22
[3,]  3    7   11   15   19   23
[4,]  4    8   12   16   20   24
> dim(p2)<-c(6,4)

```

```
> p2
      [,1] [,2] [,3] [,4]
[1,]    1    7   13   19
[2,]    2    8   14   20
[3,]    3    9   15   21
[4,]    4   10   16   22
[5,]    5   11   17   23
[6,]    6   12   18   24
```

A last thought, sometime you need to set up a matrix which you aim to use later. Useful commands are

```
> w<-1:20
> dim(w)=c(4,5)
> w
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

or

```
w2<-matrix(0,4,5)
```

gives a matrix of zeros with 4 rows and 5 columns

#### 4.0.1 Slices and elements

We can get individual members of a matrix in the obvious way so `w[2,4]` is 14. We can get columns using `w[,3]` and rows by `w[3,]`

```
> w[2,3]
[1] 10
> w[,3]
[1] 9 10 11 12
> w[,1]
[1] 1 2 3 4
> w[1,]
[1] 1 5 9 13 17
> w[4,]
[1] 4 8 12 16 20
```

### 4.1 Data and data input

Getting your own data into  $\mathbb{R}$  is reasonably easy. For a small amount of data try using `c`  
`mydata<-c(12,34,23,39,47)`

This function combines, or concatenates terms together. As an example, suppose we have the following count of the number of typos per page of these notes:

```
2 3 0 3 1 0 0 1
```

To enter this into an  $\mathbb{R}$  session we do so with

```
typos<-c(2,3,0,3,1,0,0,1)
typos
[1] 2 3 0 3 1 0 0 1
```

We can also concatenate vectors

```
> a<-c(1,2,3)
> b<-c(4,3,2,1,0)
> c(a,b)
[1] 1 2 3 4 3 2 1 0
> d<-c(a,b)
> d
[1] 1 2 3 4 3 2 1 0
```

There are several other options

```
> w<--8:2
> w
[1] -8 -7 -6 -5 -4 -3 -2 -1 0 1 2
> w<-1:3
> w
[1] 1 2 3
> t<-rep(w,each=3,length.out=12)
> t
[1] 1 1 1 2 2 2 3 3 3
```

#### 4.1.1 Missing values

If you are unfortunate enough to have a missing value you can put a NA in its place.  $\mathcal{R}$  will understand that it denotes a missing value.

#### 4.1.2 scan

if you have more data then you might prefer ( as I do ) to type in data from the keyboard, for example

```
mydata <-scan()
12
34
23 39 47
```

A blank line tells scan that input has ended. `?scan` will tell you how to read from a file.

I like to cut data and past into  $\mathcal{R}$  using scan. This can be a little tricky at times since cut and past from different applications may behave oddly - or not at all. If all else fails paste into a word processor and then to  $\mathcal{R}$  . This can be fairly convenient when entering in a few data points (10-40 say), but you might want to use a data file if you have more. You might, as mentioned above, cut and paste from a file to  $\mathcal{R}$  or you can use the `scan` options

If we have our numbers stored in a text file, then `scan` can be used to read them. You just need to tell scan to open the file and read from it . Here are two examples

- Suppose the file `ReadWithScan.txt` has contents 1 2 3 4 then the command

```
> x <- scan(file = "ReadWithScan.txt")
```

will read the contents into your R session, putting the contents of the file into `x`.

- If you had some formatting between the numbers you want to get rid of, say your file `ReadWithScan.txt` is  
1,2,3, 4  
then

```
> x<-scan(file = "ReadWithScan.txt",sep=",")
```

is what you need.

- If you are like me and forget where the file is

```
> x <- scan(file =file.choose())
```

will open a dialogue box to help you find your file.

### 4.1.3 Data tables - the dataframe

If you want to enter multivariate sets of data, you can do any of the above for each variable. However, it may be more convenient to read in tables of data at once.  $\mathbb{R}$  has many built in data sets, you can see what they are by typing `data()`. If we choose the `women` data set we load it using `data(women)` and look at it

```
> women
  height weight
1     58    115
2     59    117
3     60    120
4     61    123
5     62    126
6     63    129
7     64    132
8     65    135
9     66    139
10    67    142
11    68    146
12    69    150
13    70    154
14    71    159
15    72    164
```

This kind of structured data is called a **data.frame**. You can read files of this type into  $\mathbb{R}$  but as with the built in data frames they are not immediately available. To the height or weight data you have several possibilities

1. Use `women$height` or `women$weight`
2. Use `women[1]` or `women[2]`

3. Perhaps the simplest approach is to `attach` the file. If you type `attach(women)`  $\mathbb{R}$  will realize you want to use the names within the table and you can just refer to `height` or `weight`. The reverse is of course `detach(women)`

## 4.2 Reading data files

Suppose your data is in tabular form such as this file `ReadWithReadTable.txt`.

```
gender weeks weight
  m     40   2968
  m     38   2795
  m     40   3163
  m     35   2925
  m     36   2625
```

Notice the first row supplies column names, the second and following rows the data. The command `read.table` will read this in and store the results in a *data frame*. As we mentioned above a data frame is a special matrix where all the variables are stored as columns and each has the same length. Notice we need to specify that the headers are there in this case so we write `header=T` otherwise it is assumed that the columns do not have names, hence

```
> babies <-read.table(file="ReadWithReadTable.txt",header=T)
```

Because  $\mathbb{R}$  can hold many such dataframes the variable headings in each table are initially hidden from the main program as in the case above. You can refer to them in the same way.

- `> dim(babies)# in case I forget the dimensions`  
`[1] 24 3`  
`> babies[,2] # the second column`  
`[1] 40 38 40 35 36 37 41 40 37 38 40 38 40 36 40 38 42 39 40 37 36 38`  
`[23] 39 40`  
`> babies[2,] # the second row`  
`gender weeks weight`  
`2 m 38 2795`

- Alternatively

```
> babies[['gender']] # a factor, it prints the levels [1]
[1] m m m m m m m m m m m f f f f f f f f f f f
Levels: f m
> babies[['weight']] # a numeric vector
[1] 2968 2795 3163 2925 2625 2847 3292 3473 2628 3176 3421 2975 3317 2729 2935 2754 3210 28
```

Personally I would refer to the variables in a table using `$` as follows

```
> babies<-read.table(file="ReadWithReadTable.txt",header=T)
> babies$gender # a factor, it prints the levels [1]
> babies$gender
[1] m m m m m m m m m m m f f f f f f f f f f f
Levels: f m
```

```

> babies$weeks # a numeric vector
[1] 40 38 40 35 36 37 41 40 37 38 40 38 40 36 40 38 42 39 40 37 36 38 39 40
> x # default print out for a data.frame
  gender weeks weight
1      m    40  2968
2      m    38  2795
3      m    40  3163
4      m    35  2925
5      m    36  2625
6      m    37  2847
7      m    41  3292
8      m    40  3473
9      m    37  2628
10     m    38  3176
11     m    40  3421
12     m    38  2975
13     f    40  3317
.....
.....

```

`read.table` treats the variables as numeric or as factors. A *factor* is special class to R and has a special print method. You can think of a factor as a vector of *categories*. For example

```

(red,blue,blue,red)
or
(1,2,1,2,2,1)

```

The "levels" of the factor are displayed after the values are printed.

Do note that it is common to read data as a vector and then to use a factor vector to indicate which class, type etc of observation.

- The command `attach(babies)` makes the column headers available to  $\mathbb{R}$ . The reverse being `detach`. Be careful because already existing variable will not be overwritten.
- Some commands allow you to specify the data frame you are referring to.

Sometimes it is nice to keep the rows names in the file but not to use the vector of names as a variable. For example suppose your file was

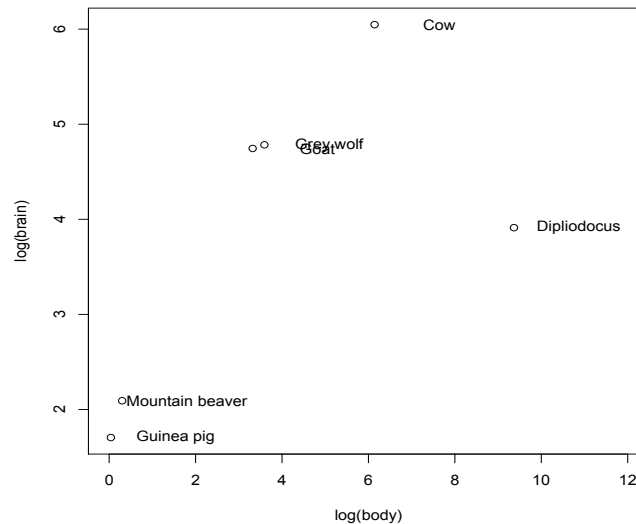
	body	brain
Mountain beaver	1.350	8.1
Cow	465.000	423.0
Grey wolf	36.330	119.5
Goat	27.660	115.0
Guinea pig	1.040	5.5
Dipliodocus	11700.000	50.0

Then

`read.table(file.choose(animals,header=TRUE,row.names=1))` gives you the data table but with a set of names in a system vector `row.names`. This can be useful, for example

```
> plot(log(body),log(brain),xlim=c(0,12) )
> text(log(body)+1.5,log(brain),row.names(animals) )
```

gives



#### 4.2.1 Excel files

If you have Excel files you can always save them as `csv` files. Then they can be read into R using `x<-read.csv(file="fred",header=T)`.

The  $\mathbb{R}$  documentation says ( I abbreviate ): The most common —R data import/export question seems to be ‘how do I read an Excel spreadsheet’. *The first piece of advice is to avoid doing so if possible!* If you have access to Excel, export the data you want from Excel in tab-delimited or comma-separated form, and use `read.delim` or `read.csv` to import it into  $\mathbb{R}$ . (You may need to use `read.delim2` or `read.csv2` in a continental European locale that uses comma as the decimal point.). Exporting a DIF file and reading it using `read.DIF` is another possibility.

Note that an Excel `.xls` is not just a spreadsheet: such files can contain many sheets, and the sheets can contain formulae, macros and so on. Not all readers can read other than the first sheet, and may be confused by other contents of the file. Windows users can use `odbcConnectExcel` in package `RODBC`. This can select rows and columns from any of the sheets in an Excel spreadsheet file. Also ( only for Windows) the package `xlsReadWrite` has a function `read.xls` to read `.xls` files . Perl users have contributed a module `OLE::SpreadSheet::ParseExcel` and a program `xls2csv.pl` to convert Excel spreadsheets to CSV files. Package `gdata` provides a basic wrapper in its `read.xls` function. The package `Foreign` has several useful functions for reading files from packages like STATA or SPSS- as we shall see.

### 4.3 Graphics

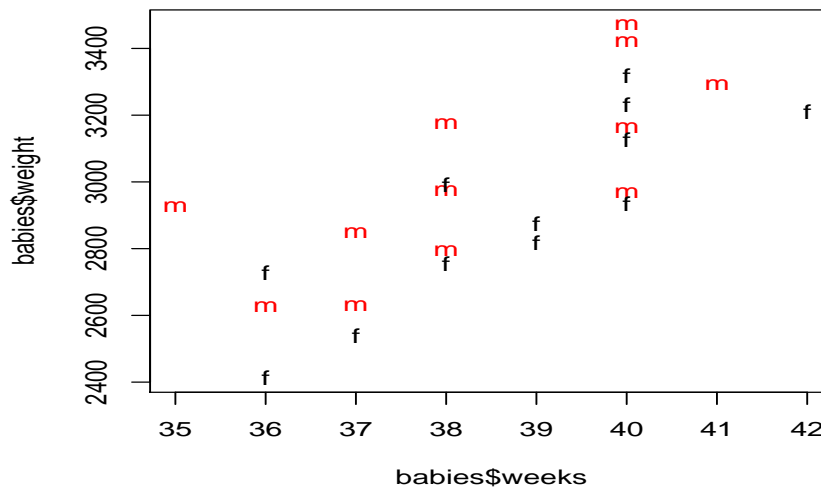
There are two graphic systems in R the traditional system and the grid system. For the moment we will look at the traditional one and return to the grid if we have time. As we have seen the `plot` command gives us basic plots. For the birth weight data we have

```

> babies<-read.table(file.choose(),header=T)
> names(babies)# I always forget what is in the tables so this is my check
[1] "gender" "weeks" "weight"
> plot(babies$weeks,babies$weight,pch=as.character(babies$gender),col=as.integer(babies$gender))

```

Here I have used the `pch` command to give different symbols for males and females. While I am at it I have use different colours. I have to use the `babies$gender` vector but `pch` expects characters. Hence we coerce `babies$gender` to characters. In the same way `col` expects integers so we coerce to integers.

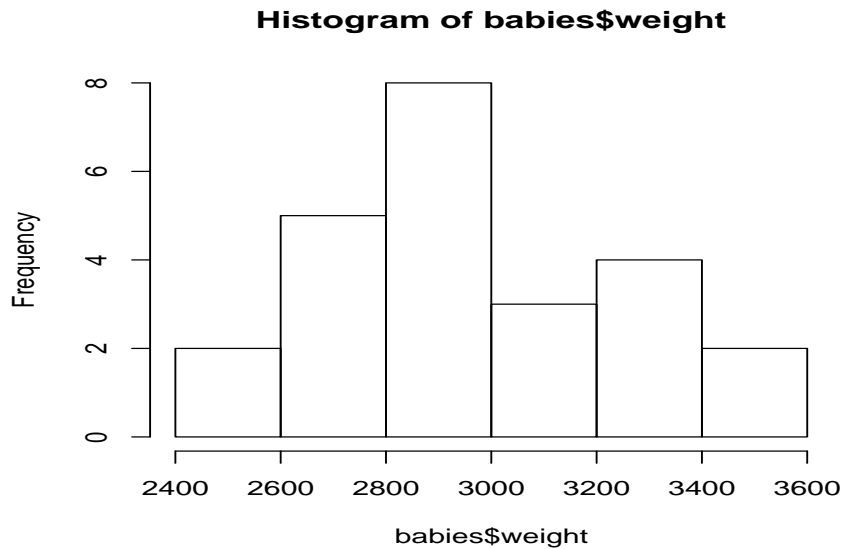


You can add line and points to an already existing graph using `lines` or `points`. We can look at histograms

```

> hist(babies$weight)

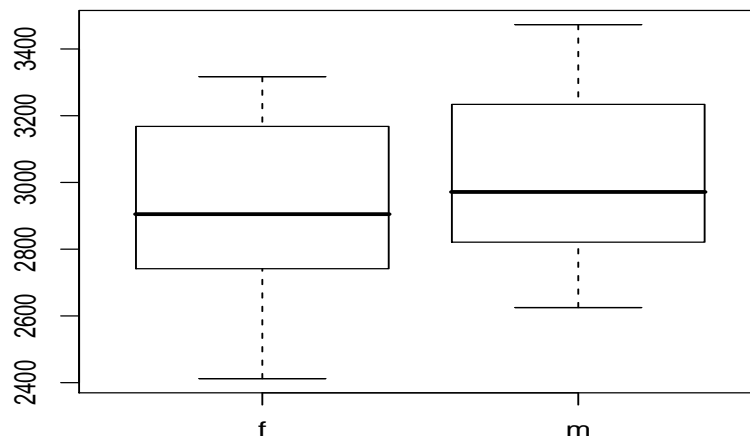
```



Or box-plots

```
> box-plot(babies$weight~babies$gender)
```

Notice here we use the model formula `babies$gender` which enables us to perform box-plots by gender



We can have multiple plots on the one page. The parameter `mfrow` can be used to configure the graphics sheet so that subsequent plots appear row by row, one after the other in a rectangular layout, on the one page. For a column by column layout, use `mfcol` instead. In the example below we present four different transformations of the primates data, in a two by two layout:

```

par(mfrow=c(2,2), pch=16)
  data(Animals)      # Needed if Animals (MASS package) is not already loaded
  attach(Animals)
  plot(body, brain)
  plot(sqrt(body), sqrt(brain))
  plot((body)^0.1, (brain)^0.1)
  plot(log(body),log(brain))
  detach(Animals)
par(mfrow=c(1,1),pch=1)      # Restore to 1 figure per page

```

#### 4.4 Simple Stats

We can do simple stats with few problems. If we take the babies data we might test to see if the means differ for the sexes. If we we (naively) to do a t test then

```
> t.test(babies$weight~babies$gender)
```

Welch Two Sample t-test

```

data: babies$weight by babies$gender
t = -0.9775, df = 21.996, p-value = 0.3390
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -351.7111  126.3778
sample estimates:
mean in group f mean in group m
      2911.333      3024.000

```

Of course is you look up the help you will see we can specify the data file to get

```
> t.test(weight~gender,data=babies)
```

Welch Two Sample t-test

```

data: weight by gender
t = -0.9775, df = 21.996, p-value = 0.3390
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -351.7111  126.3778
sample estimates:
mean in group f mean in group m
      2911.333      3024.000

```

Notice we write `weight ~ gender`. This is a model formula and expresses our wish to think of the data vector `weight` in terms of the *factor* `gender`. In this case  $\mathbb{R}$  knows that `gender` is a factor but it is up to you to ensure that it is clear that a vector is a factor. Eg

```

> f<-rep(1:3,each=3,length.out=12)
> f
[1] 1 1 1 2 2 2 3 3 3

```

```
> f<-factor(f)
> f
[1] 1 1 1 2 2 2 3 3 3
Levels: 1 2 3
```

A more cautious person might use the non-paired Wilcoxon (Mann-Whitney)

```
> wilcox.test(babies$weight~babies$gender)
```

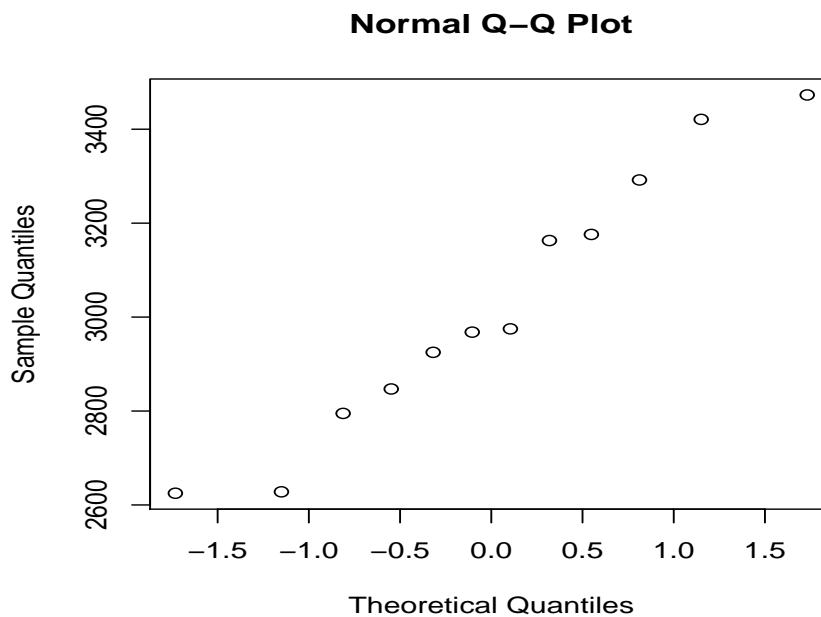
Wilcoxon rank sum test

```
data: babies$weight by babies$gender
W = 58, p-value = 0.4428
alternative hypothesis: true location shift is not equal to 0
```

```
>
```

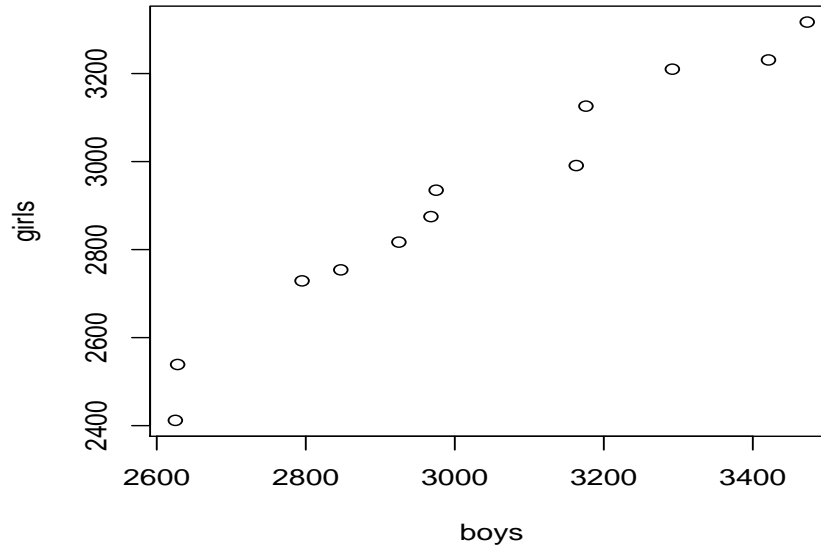
A really cautious person would check for normality. The boys are the first 12 observations, personally I would do a probability plot

```
> boys<-babies$weight[1:12] # makes boys vector
> qqnorm(boys) #probability plot
```



To compare the boys and girls I would use a Q-Q plot

```
> girls<-babies$weight[13:24] make a girls vector
> qqplot(boys,girls)#q-q plot
```



## 5 Regression

We now look at some more complex statistics. We begin with regression. To start with we can use the babies data. If you were doing regression you would probably start by using the help system to find out the function and the parameters. The help output is included in these notes, see section 11

### 5.0.1 Sales

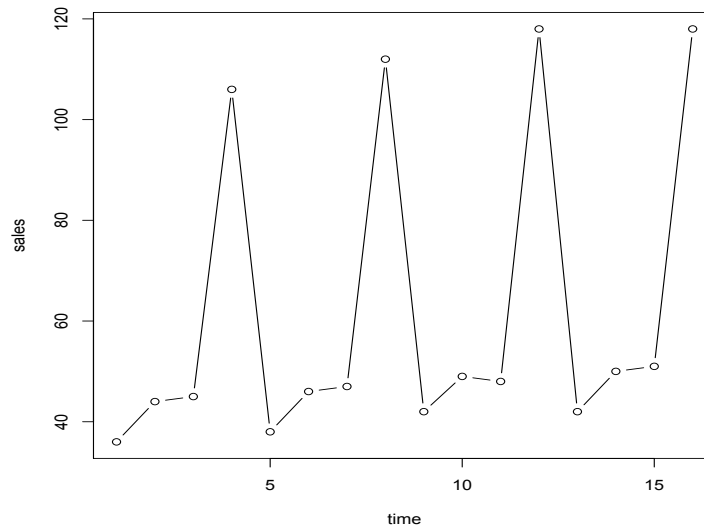
We take a small sales data set, these are sales in Canadian dollars by quarter in the file `gems.csv`

```
> gems<-read.csv(file.choose(),header=TRUE)# read the data file
> gems # look at the data
  sales time
1    36    1
2    44    2
3    45    3
4   106    4
5    38    5
6    46    6
7    47    7
8   112    8
9    42    9
10   49   10
11   48   11
12  118   12
13   42   13
```

```

14    50    14
15    51    15
16   118    16
> attach(gems) # make it available
> plot(time,sales,type="b")# plot the data - lines and points

```



We clearly need to take the seasonal into effect so we have some dummies which I shall call quarter

```

> quarter=factor(rep(1:4,each=1,length.out=16))
> cbind(sales,time,quarter)
   sales time quarter
[1,]   36    1      1
[2,]   44    2      2
[3,]   45    3      3
[4,]  106    4      4
[5,]   38    5      1
[6,]   46    6      2
[7,]   47    7      3
[8,]  112    8      4
[9,]   42    9      1
[10,]   49   10      2
[11,]   48   11      3
[12,]  118   12      4
[13,]   42   13      1
[14,]   50   14      2
[15,]   51   15      3
[16,]  118   16      4
> r1=lm(sales~time+quarter) # this is the regression command

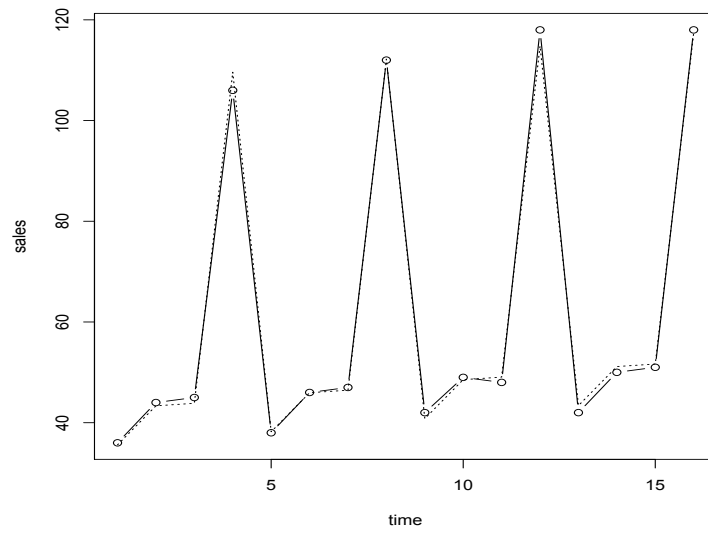
```

We now have a regression object that we can examine

```
> summary(r1)
Call:
lm(formula = sales ~ time + quarter)
Residuals:
    Min       1Q   Median       3Q      Max
-3.6000 -0.7500  0.2250  0.6125  3.2000
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 34.95000    1.08799   32.123 3.17e-12 ***
time         0.65000    0.09579    6.786 3.01e-05 ***
quarter2     7.10000    1.21547    5.841 0.000112 ***
quarter3     6.95000    1.22674    5.665 0.000145 ***
quarter4    72.05000    1.24530   57.858 5.08e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.714 on 11 degrees of freedom
Multiple R-squared:  0.9978, Adjusted R-squared:  0.997
F-statistic: 1230 on 4 and 11 DF,  p-value: 1.690e-14

> anova(r1) # for an ANOVA table
Analysis of Variance Table

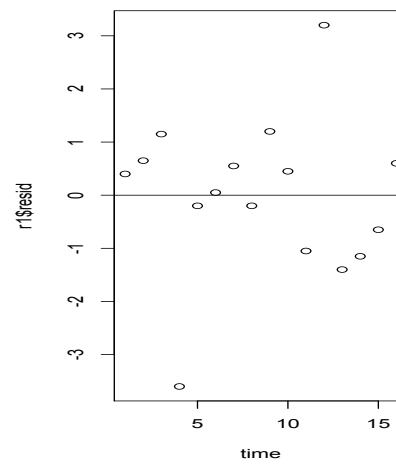
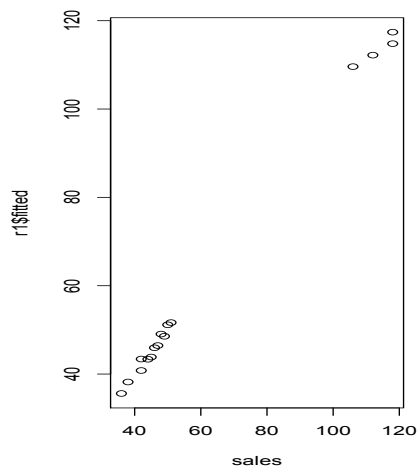
Response: sales
      Df Sum Sq Mean Sq F value    Pr(>F)
time    1  1254.1  1254.1   427.11 3.759e-10 ***
quarter 3 13197.6  4399.2  1498.17 1.208e-14 ***
Residuals 11    32.3     2.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> lines(time,r1$fitted,lty=3)# diagnostic plot
```



```

> par(mfcol=c(1,2))
> plot(sales,r1$fitted)
> plot(time,r1$resid)
> abline(h=0)
> par(mfrow=c(1,1))

```



### 5.0.2 Babies

The data below gives the weight of (Australian) babies at birth by gender

male		female	
Age	weight	age	weight
40	2968	40	3317
38	2795	36	2729
40	3163	40	2935
35	2925	38	2754
36	2625	42	3210
37	2847	39	2817
41	3292	40	3126
40	3473	37	2539
37	2628	36	2412
38	3176	38	2991
40	3421	39	2875
38	2975	40	3231

We start by attempting to fit a model which relates time of gestation to weight. The command we want is, as before, `lm`. Since it gets irritating to refer to `babies$weight` etc we `attach` the data frame

```
attach(babies)
```

so we try

```
r1<-lm( weight~weeks)
```

By this we mean we aim to explain the variable `weight` by `time`. In fact if we do this nothing appears to happen. In fact `R` does the computations and puts the output in the object `r1`. To get results we must summarize `r1`, thus

```
> attach(babies)
> r1<-lm(weight~weeks)
> summary(r1)
```

Call:

```
lm(formula = weight ~ weeks)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-262.032 -158.292   8.355   88.147  366.496
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1485.0      852.6   -1.742   0.0955 .
weeks           115.5       22.1    5.228 3.04e-05 ***
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 192.6 on 22 degrees of freedom

Multiple R-Squared: 0.554, Adjusted R-squared: 0.5338

F-statistic: 27.33 on 1 and 22 DF, p-value: 3.04e-05

The object `r1` has lots of information thus

```
> names(r1)
[1] "coefficients" "residuals" "effects" "rank"
[5] "fitted.values" "assign" "qr" "df.residual"
[9] "xlevels" "call" "terms" "model"
>
```

and we can plot it - try `plot(r1)`.

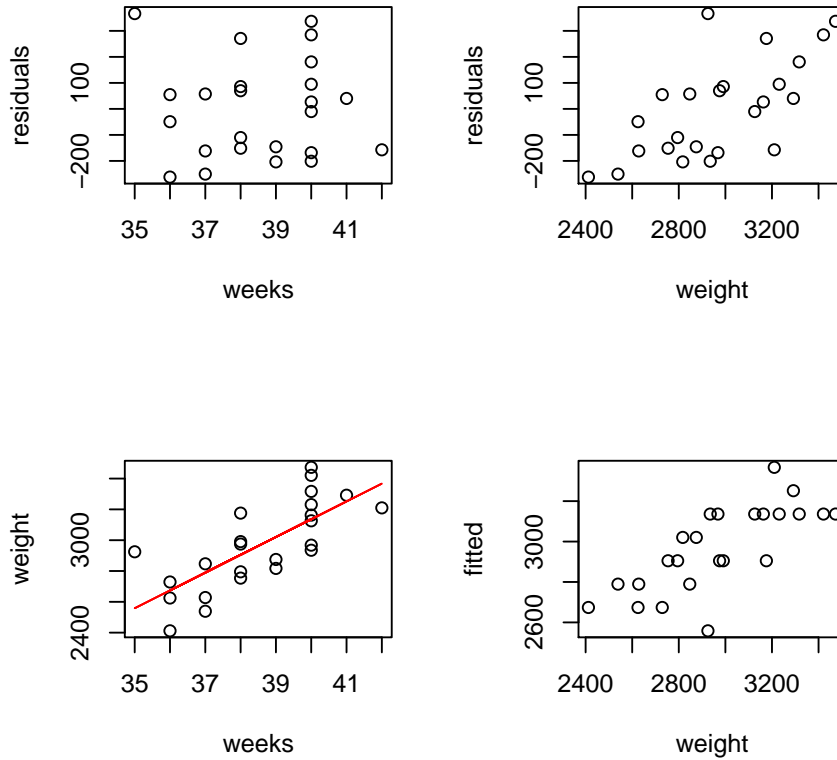
If you like/understand the ANOVA we have

```
> anova(r1)
Analysis of Variance Table

Response: weight
      Df Sum Sq Mean Sq F value Pr(>F)
weeks   1 1013799 1013799  27.330 3.04e-05 ***
Residuals 22  816074   37094
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
>
```

I like to look at the residuals, plot the line (which we can do here ), plot the residuals against the fitted,

```
> par(mfrow=c(2,2)) #tell R to expect 4 plots
> plot(weights,r1$resid,ylab="residuals") # plot the residuals and label y axis residuals
> plot(weeks,weight) # plot the data
> lines(weeks,r1$fitted,col=2)# over plot the regression as a line
> plot(weight,r1$fitted,ylab="fitted") # plot the actual against the fitted
> par(mfrow=c(1,1))# back to single plot
```



Of course it is probable that the gender also has some input so we can try adding gender (a factor) to the model formula

```
> r2<-lm(weight~weeks +gender)
> summary(r2)
```

Call:

```
lm(formula = weight ~ weeks + gender)
```

Residuals:

Min	1Q	Median	3Q	Max
-257.49	-125.28	-58.44	169.00	303.98

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-1773.32	794.59	-2.232	0.0367 *
weeks	120.89	20.46	5.908	7.28e-06 ***
genderm	163.04	72.81	2.239	0.0361 *

---

Signif. codes: 0 \*\*\* 0.001 \*\* 0.01 \* 0.05 . 0.1 1

Residual standard error: 177.1 on 21 degrees of freedom  
 Multiple R-Squared: 0.64, Adjusted R-squared: 0.6057  
 F-statistic: 18.67 on 2 and 21 DF, p-value: 2.194e-05

```
> anova(r2)
Analysis of Variance Table
```

```
Response: weight
      Df Sum Sq Mean Sq F value    Pr(>F)
weeks  1 1013799 1013799 32.3174 1.213e-05 ***
gender  1  157304  157304   5.0145  0.03609 *
Residuals 21  658771   31370
---

```

```
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
```

Here we see the gender has a significant effect, both from the anova and from the coefficient estimates. Our model thus has a slope and different intercepts depending on the gender.

Of course it is possible that the genders have differing slopes - that is we have an interaction between weeks and gender. To model this we use `weeks *gender` a shorthand for `weeks +gender+weeks:gender` where `weeks:gender` is the interaction term.

```
> r3<-lm(weight~weeks *gender)
>
> summary(r3)
```

```
Call:
lm(formula = weight ~ weeks * gender)
```

```
Residuals:
      Min       1Q   Median       3Q      Max
-246.69 -138.11  -39.13  176.57  274.28
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2141.67    1163.60  -1.841 0.080574 .
weeks         130.40     30.00   4.347 0.000313 ***
genderm       872.99    1611.33   0.542 0.593952
weeks:genderm -18.42     41.76  -0.441 0.663893
---

```

```
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
```

Residual standard error: 180.6 on 20 degrees of freedom  
 Multiple R-Squared: 0.6435, Adjusted R-squared: 0.59  
 F-statistic: 12.03 on 3 and 20 DF, p-value: 0.0001010

```
> anova(r3)
Analysis of Variance Table
```

```

Response: weight
          Df Sum Sq Mean Sq F value    Pr(>F)
weeks     1 1013799 1013799 31.0779 1.862e-05 ***
gender    1  157304  157304   4.8221  0.04006  *
weeks:gender 1    6346    6346   0.1945  0.66389
Residuals 20  652425   32621
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1  1
>

```

For influence measure try

- `summary.lm` for summary and related methods.
- `influence.measures`.
- `hat` for the hat matrix diagonals
- `dfbetas`, `dffits`, `covratio`, `cooks.distance`, `lm`.

### 5.0.3 Fixed effects ANOVA

The table 1 gives the output of thermocouples made of 3 different materials at 3 different temperatures. Possible models are

Temperature	material					
	M1		M2		M3	
T1	130	155	34	40	20	70
	74	80	80	75	82	58
T2	150	188	136	122	25	70
	159	126	108	115	58	45
T3	138	110	174	120	96	104
	168	160	150	139	82	60

Table 1: Output voltages

$$y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk} \quad (1)$$

that is just the main effects and

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk} \quad (2)$$

where  $(\alpha\beta)_{ij}$  denotes the interaction between the main effects of material and temperature. I use a corner-point constraint

$$(\alpha\beta)_{1j} = 0 \quad i = 1, 2, 3 \text{ and } (\alpha\beta)_{i1} = 0 \quad j = 1, 2, 3$$

The table of means, see table 2 is and these are plotted in figure 2. We see that the response over temperature is different for the different materials - hence the interaction. How do we proceed?

How do we proceed? Type the values into  $R$ , call the values  $y$ . I also create two vectors `temp` and `mat`. You could just type them in or use the command `rep`

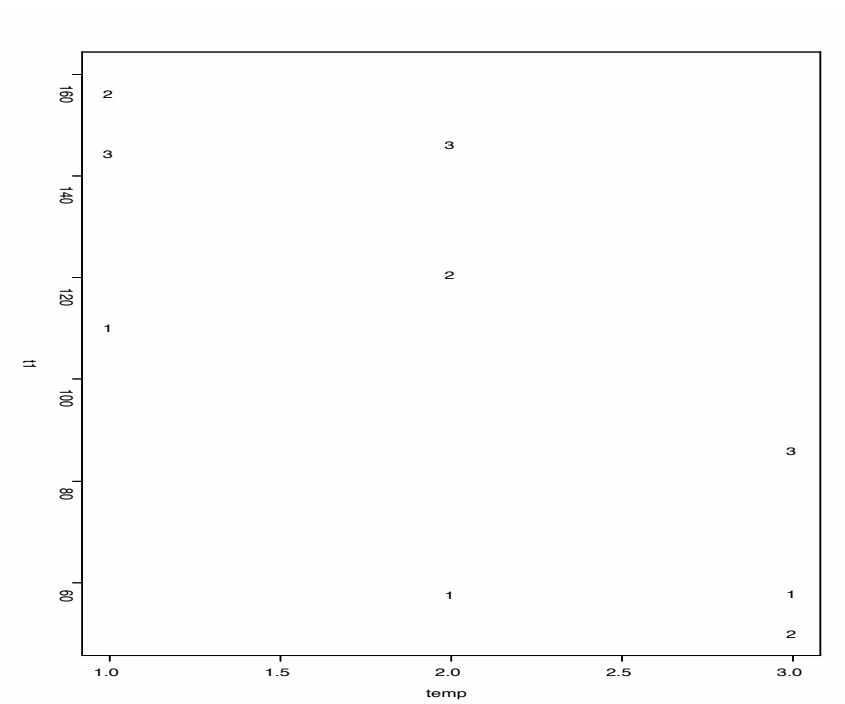


Figure 2: interaction example

	M1	M2	M3
T1	109.75	155.75	144.00
T2	57.25	120.25	145.75
T3	57.50	49.50	85.50

Table 2: Means of voltages

```
> temp<-rep( 1:3 ,each=12,length.out=36)
> temp
[1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
> temp<-factor(temp)
> mat<-rep( 1:3 ,each=2,length.out=36)
> mat
[1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
> mat<-factor(mat)
```

We must not forget to tell R that we are using factors. With the variables as shown use `lm` to get

```
> y
[1] 130 155 34 40 20 70 74 80 80 75 82 58 150 188 136 122 25 70
[19] 159 126 108 115 58 45 138 110 174 120 96 104 168 160 150 139 82 60
> temp
[1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
Levels: 1 2 3
> mat
[1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
Levels: 1 2 3
> v1<-lm(y~mat*temp)
> summary(v1)
lm(formula = y ~ mat * temp)
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   109.75      12.41    8.845 1.85e-09 ***
mat2           -52.50      17.55   -2.992  0.00586 **
mat3           -52.25      17.55   -2.978  0.00607 **
temp2           46.00      17.55    2.621  0.01421 *
temp3           34.25      17.55    1.952  0.06141 .
mat2:temp2      17.00      24.82    0.685  0.49917
mat3:temp2     -54.00      24.82   -2.176  0.03848 *
mat2:temp3      54.25      24.82    2.186  0.03766 *
mat3:temp3     -6.25      24.82   -0.252  0.80307
Residual standard error: 24.82 on 27 degrees of freedom
Multiple R-Squared: 0.7706, Adjusted R-squared: 0.7026
F-statistic: 11.34 on 8 and 27 DF, p-value: 7.018e-07

> anova(v1)
Analysis of Variance Table
```

```

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
mat         2  31833   15916  25.8433 5.355e-07 ***
temp        2  15734    7867  12.7736 0.0001247 ***
mat:temp    4   8296    2074   3.3676 0.0232758 *
Residuals 27  16629     616
---

```

Notice we have several observations per cell in the data table. We can thus estimate the interaction terms. If there was only one observation per cell then the interaction terms would be aliased. They would be inseparable from the main effect terms!

An alternative approach is to use the command `aov`. You might also consider looking at the package `car`. There is also `reml.R`, it is up to you!

## 6 Introduction

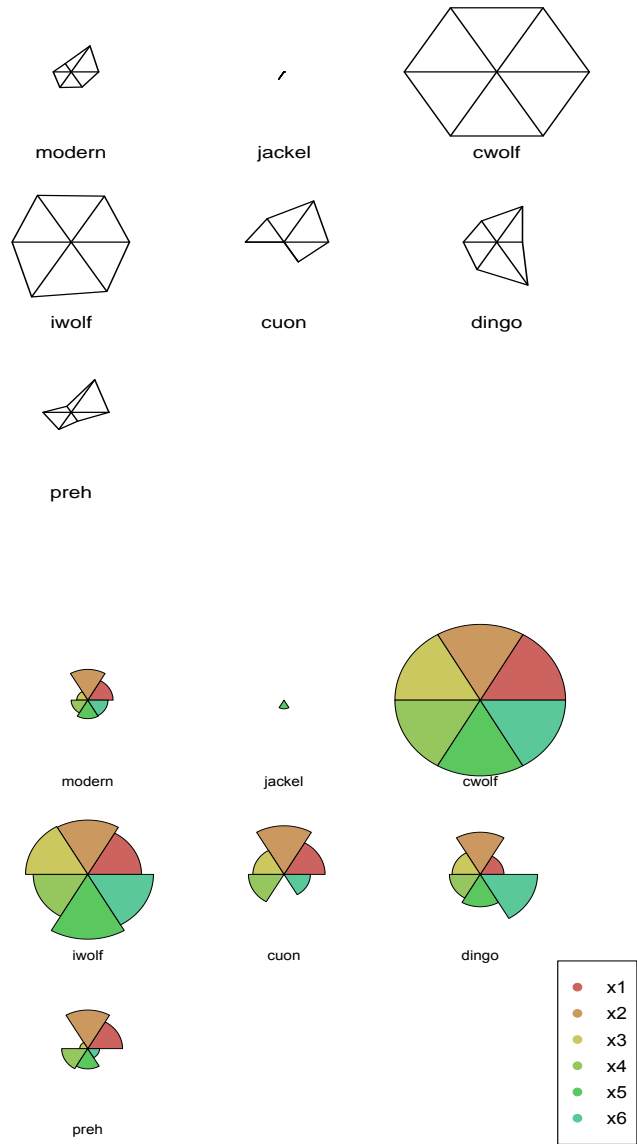
In many situations we have complex data sets with several observations on each subject. For example  $x_1, x_2, \dots, x_6$  are measurements made at different sites on the teeth of the following animals.

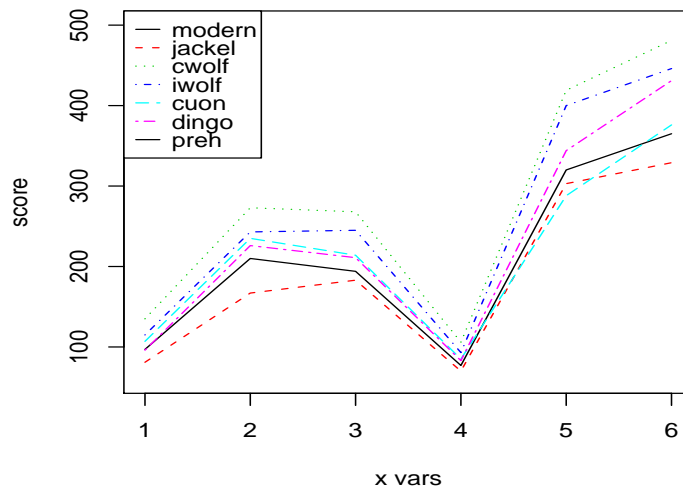
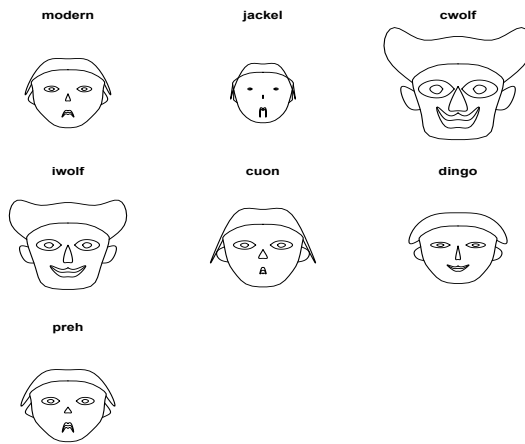
	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
modern	97	210	194	77	320	365
jackel	81	167	183	70	303	329
cwolf	135	273	268	106	419	481
iwolf	115	243	245	93	400	446
cuon	107	235	214	85	288	376
dingo	96	226	211	83	344	431
preh	103	221	191	81	323	350

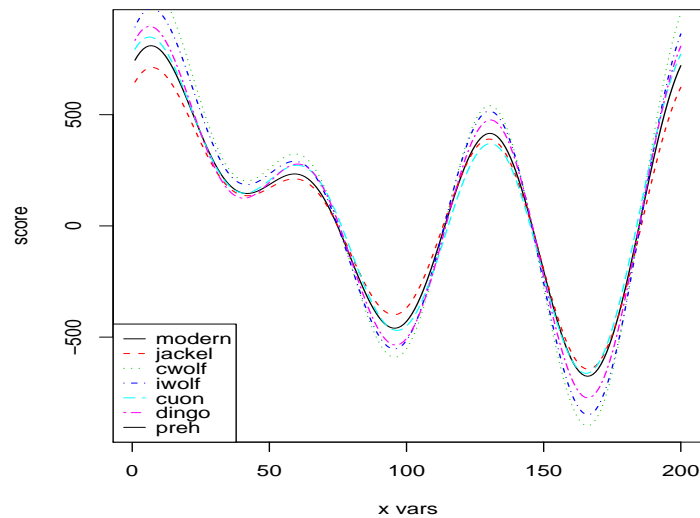
One of the real problems in this case is the difficulty in drawing pictures and seeing what is going on. There are many specialized graphical methods that can be used in these situations but in my view they

1. Are difficult to draw, they require specialized software.
2. Are difficult to interpret. especially by non-technical users.

We can illustrate some ideas on the dog data (transposed)







## 6.1 Principal Components

To determine the interrelationships between the variables we can of course work out the correlations ( or covariances) between the  $\mathbf{x}_j$ . If  $\text{cov}(\mathbf{x}_i, \mathbf{x}_j) = c_{ij}$  then we can write these in a matrix  $\mathbf{C}$  whose  $ij$ th element is  $c_{ij}$ .

For the dogs data this gives a covariance matrix

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$x_1$	288.143	525.167	484.690	193.333	652.405	773.976
$x_2$	525.167	1055.667	889.500	359.333	1145.000	1558.333
$x_3$	484.690	889.500	961.143	350.833	1338.762	1630.524
$x_4$	193.333	359.333	350.833	135.667	485.667	592.000
$x_5$	652.405	1145.000	1338.762	485.667	2429.619	2460.238
$x_6$	773.976	1558.333	1630.524	592.000	2460.238	3151.810

or the correlation matrix

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$x_1$	1.000	0.952	0.921	0.978	0.780	0.812
$x_2$	0.952	1.000	0.883	0.950	0.715	0.854
$x_3$	0.921	0.883	1.000	0.972	0.876	0.937
$x_4$	0.978	0.950	0.972	1.000	0.846	0.905
$x_5$	0.780	0.715	0.876	0.846	1.000	0.889
$x_6$	0.812	0.854	0.937	0.905	0.889	1.0001

While these correlation are useful they are still a complex structure and quite difficult to decipher. It would be very much simpler if the variables were **uncorrelated**. In this case we would have a diagonal matrix and from an intuitive view uncorrelated variables seem rather better. Converting to uncorrelated variables is a simple problem in linear algebra. The new variables PC1, PC2, etc are linear combinations of the original variables, here  $x_1, x_2, x_3, x_4, x_5, x_6$ .

We do not need a very acute grasp of algebra as this is ( has to be ) done by software. What we need to focus on is that instead of our original measurement variables ( $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_p$ ) we have a new set ( $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_p$ ) called the principal components which are uncorrelated and which are made up of our original variables.

For example using R

```
> d1<-prcomp(dogs, scale = TRUE)
> summary(d1)
Importance of components:
              PC1      PC2      PC3      PC4      PC5      PC6
Standard deviation  2.330 0.6077 0.3587 0.2545 0.07821 0.04628
Proportion of Variance 0.905 0.0615 0.0215 0.0108 0.00102 0.00036
Cumulative Proportion 0.905 0.9664 0.9878 0.9986 0.99964 1.00000
end{verbatim}
```

We have a bit more detail

```
> d1
Standard deviations:
[1] 2.33002707 0.60767458 0.35872870 0.25448045 0.07821380 0.04627633
```

Rotation:

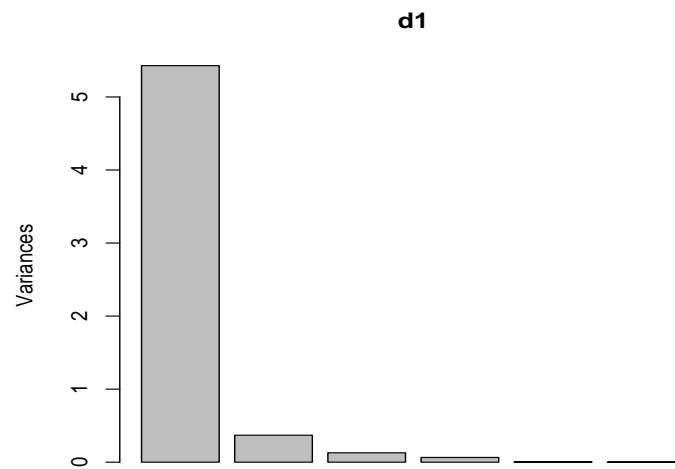
```
      PC1      PC2      PC3      PC4      PC5      PC6
x1 0.4099426 0.40138614 -0.45937507 -0.005510479 0.009871866 0.6779992
x2 0.4033020 0.48774128 0.29350469 -0.511169325 -0.376186947 -0.3324158
x3 0.4205855 -0.08709575 0.02680772 0.737388619 -0.491604714 -0.1714245
x4 0.4253562 0.16567935 -0.12311823 0.170218718 0.739406740 -0.4480710
x5 0.3831615 -0.67111237 -0.44840921 -0.404660012 -0.136079802 -0.1394891
x6 0.4057854 -0.33995660 0.69705234 -0.047004708 0.226871533 0.4245063
>
```

Here we have 4 new variables PC1,PC2, etc which are constructed from our original data,  
 $PC1 = 0.4099426*x1 + 0.4033020*x2 + 0.4205855*x3 + 0.4253562*x4 + 0.3831615*x5 + 0.4057854*x6$

**Notice we have used the correlation matrix as the basis of our transformation.**  
 We could equally well have used the covariance matrix giving

### 6.1.1 Reducing Dimension

If we plot the variances of the new components we get

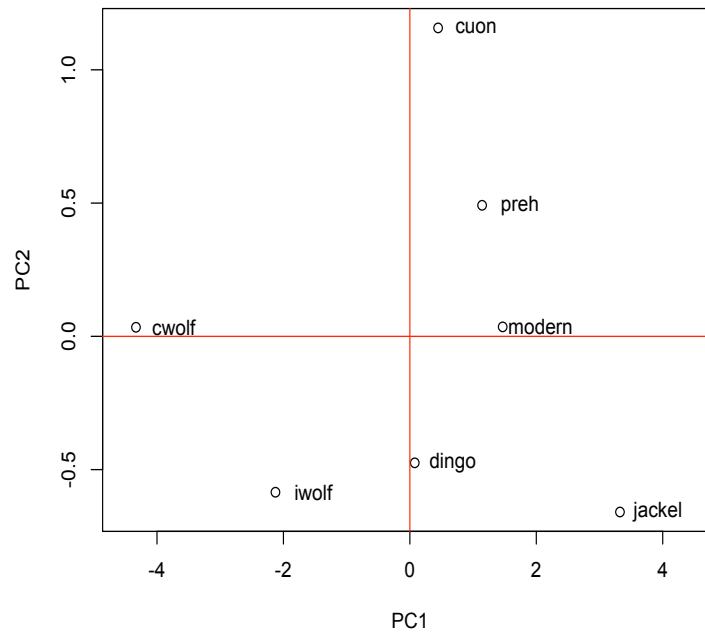


We see that in terms of the variation we have

k	1	2	3	4	5	6
Varn due to kth PC	0.905	0.0615	0.0215	0.0108	0.00102	0.00036
Cumulative variance	0.905	0.9664	0.9878	0.9986	0.99964	1.00000

It looks as though the first one or two principal components explain nearly all the variability. We could think in terms of just these two and reduce the dimensionality of our problem from 6 to 2 .

If we plot the values in the PC scale we have



### 6.1.2 Summary

- We use eigenvalue analysis to find the principal components.
- These new components are uncorrelated.
- We can use our eigen analysis on either the covariance or the correlation matrix.
- Plotting the variance or standard deviation of the principal components against order, a *scree plot* may help us reduce the dimension of the problem.
- Jolliffe suggests ignoring eigenvalues whose values are less than 1 when doing a correlation based extraction

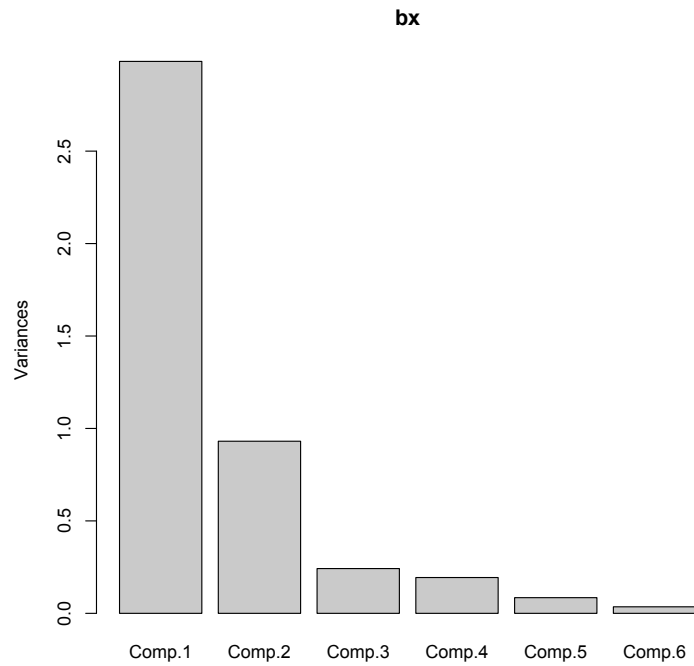
### 6.1.3 Swiss Bank Notes

Six variables measured on 100 genuine and 100 forged old swiss 1000-franc bills. The observations 1-100 are genuine, the other 100 observations are forged banknotes. The data is used in Applied Multivariate Statistical Analysis (2003) by Wolfgang Hrdle and Lopold Simar.

source: [1988] "Multivariate Statistics, A practical Approach", Cambridge University Press. Flury and Riedwyl

- Length of the bill
- Height of the bill, measured on the left
- Height of the bill, measured on the right

- Distance of inner frame to the lower border
- Distance of inner frame to the upper border
- Length of the diagonal



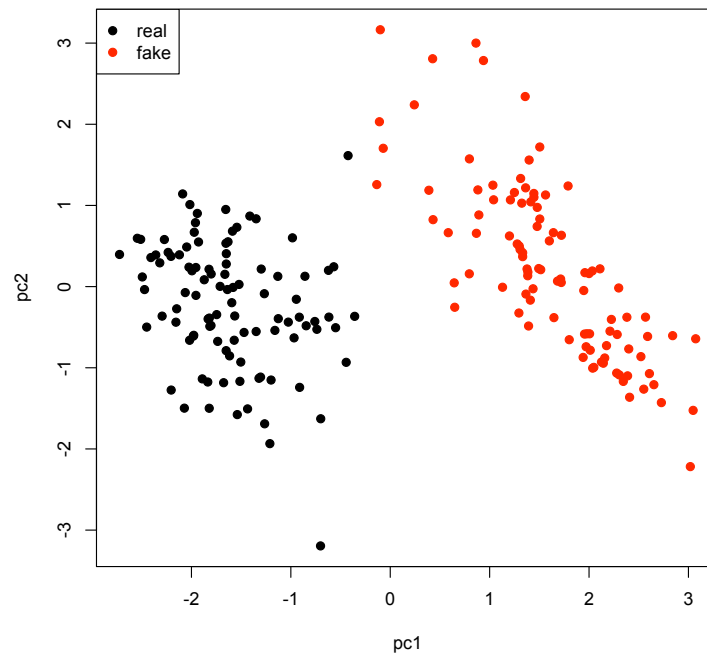
```
summary(bx)
```

```
Importance of components:
```

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Standard deviation	1.7162629	1.1305237	0.9322192	0.67064796	0.51834053
Proportion of Variance	0.4909264	0.2130140	0.1448388	0.07496145	0.04477948
Cumulative Proportion	0.4909264	0.7039403	0.8487791	0.92374054	0.96852002

	Comp.6
Standard deviation	0.43460313
Proportion of Variance	0.03147998
Cumulative Proportion	1.00000000



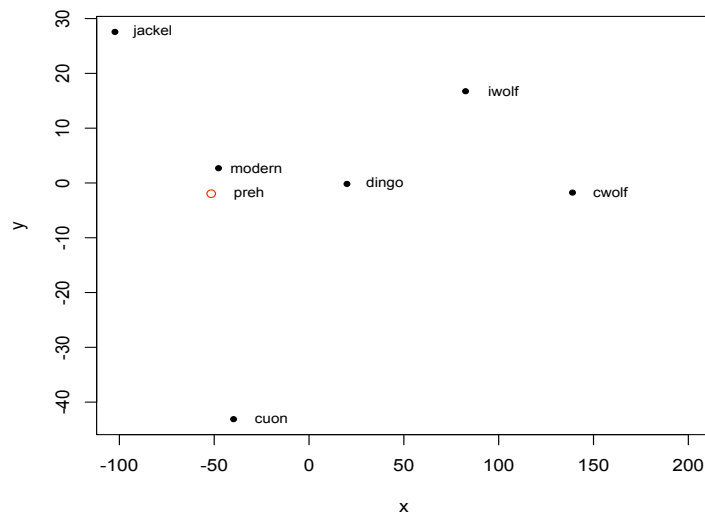
## 7 MDS

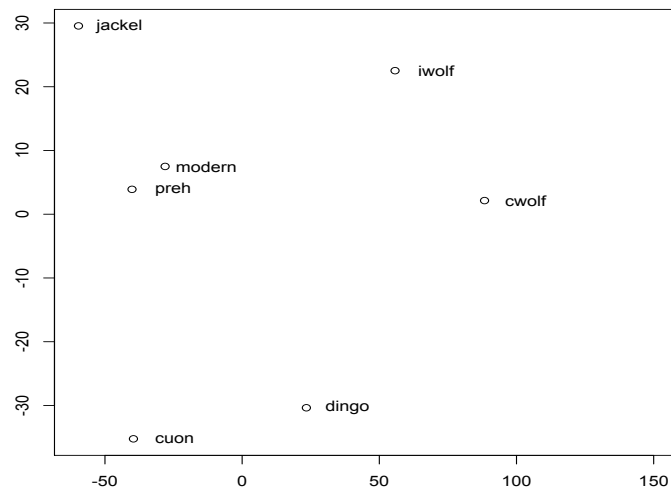
Suppose we take a matrix of distances between major cities. We construct a map to reproduce the distances we are given in the distance matrix. As a result of the MDS analysis, we would get a two-dimensional representation of the locations of the cities, that is, we would basically obtain a two-dimensional map.

In general then, MDS attempts to arrange "objects" (major cities in this example) in a space with a given number of dimensions (2 in our example) so as to reproduce the observed distances. As a result, we can "explain" the distances in terms of underlying dimensions; in our example, we could explain the distances in terms of the two geographical dimensions: north/south and east/west.

Clearly with just distance information the actual orientation of axes in the final solution is arbitrary. In our example, we could rotate the map in any way we want, the distances between cities remain the same. Thus, the final orientation of axes in the plane or space is mostly the result of a subjective decision by the researcher, who will choose an orientation that can be most easily explained.

MDS is not so much an exact procedure as rather a way to "rearrange" objects in an efficient manner, so as to arrive at a configuration that best approximates the observed distances. It actually moves objects around in the space defined by the requested number of dimensions, and checks how well the distances between objects can be reproduced by the new configuration. In more technical terms, it uses a function minimization algorithm that evaluates different configurations with the goal of maximizing the goodness-of-fit (or minimizing "lack of fit").





## 8 GLIMS

### 8.0.4 Toxicity of the Tobacco budworm

Collett reports an experiment in which batches of 20 moths were exposed for 3 days to a pyrethroid. The number in each batch which were killed or knocked down are recorded in table 8.0.4 below. If we plot the numbers killed by dose we see in figure 3 that the relation

		dose					
		1	2	4	8	16	32
Sex	male	1	4	9	13	18	20
	female	0	2	6	10	12	16

Table 3: Numbers of moths killed

between the predictor variable (dose) and the response (deaths) is not linear. However since the response is binomial, deaths per 20, there is no real reason to suppose that it should be. Since we have binomial error I will try fitting a generalized linear model. The canonical link is the logistic. So if  $\pi$  is the probability of death then perhaps

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 \text{sex} + \beta_2 \text{dose}$$

One possibility is to use the logit link ( which is the canonical one for the binomial) and if the model is not satisfactory to try variants.

If you are not sure then you can try the empirical variant Suppose  $y_j$  is the binomial response from  $n$ , we can plot the empirical logistic

$$z_j = \log\left(\frac{y_j + 0.5}{n - y_j + 0.5}\right)$$

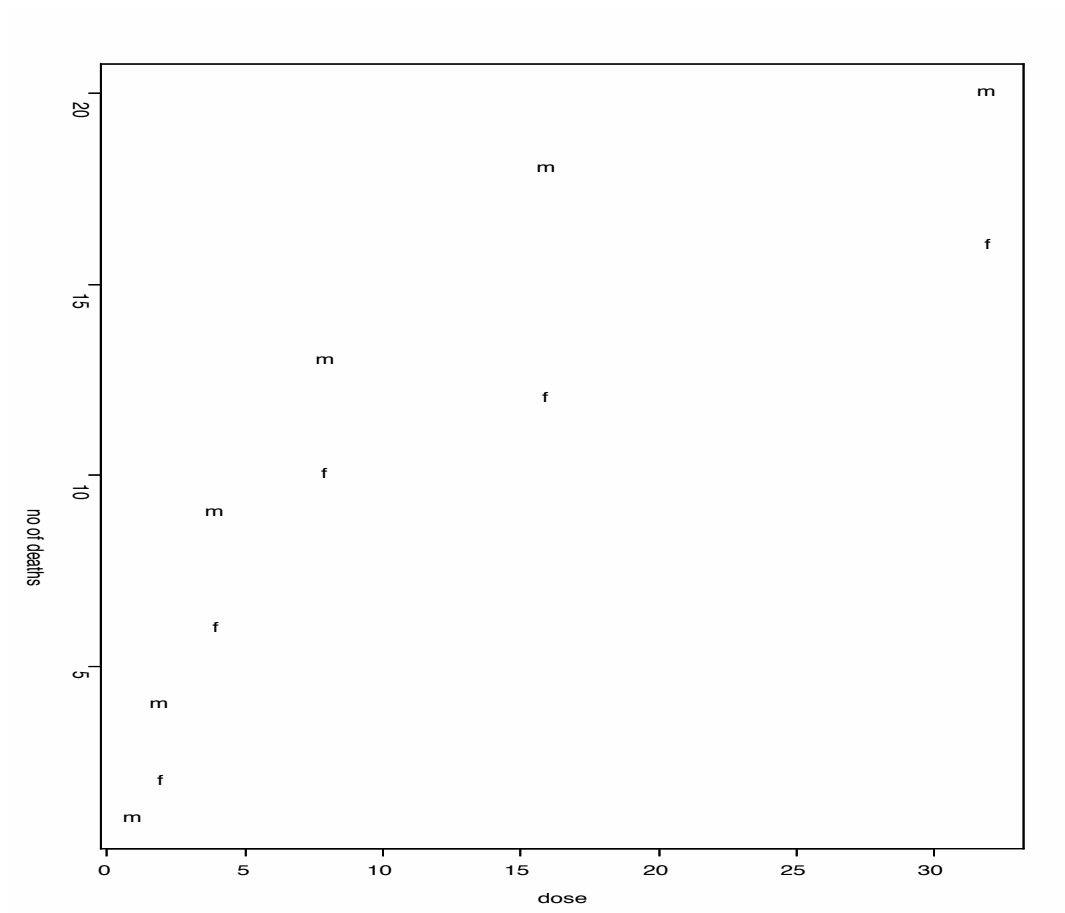


Figure 3: Moths deaths by dose

against the explanatory variable dose.

```
> moth<-read.table(file.choose(),header=T) # read table
> moth
  dose no sex
1     1  1  1
2     2  4  1
3     4  9  1
4     8 13  1
5    16 18  1
6    32 20  1
7     1  0  2
8     2  2  2
9     4  6  2
10    8 10  2
11   16 12  2
12   32 16  2

> p<-moth$no/20 # estimate probability
> logit<- log(p/(1-p)) # estimate logits
> plot(moth$dose,logit) #plot logits against dose and log dose
> par(mfrow=c(2,1))
> plot(moth$dose,logit)
> plot(log(moth$dose),logit)
> y<-cbind(moth$no,20-moth$no) #se up input MATRIX
> y
      [,1] [,2]
[1,]    1  19
[2,]    4  16
[3,]    9  11
[4,]   13   7
[5,]   18   2
[6,]   20   0
[7,]    0  20
[8,]    2  18
[9,]    6  14
[10,]  10  10
[11,]  12   8
[12,]  16   4
> g1<-glm(y~log(dose)+sex,family=binomial,data=moth) # glm
> summary(g1)
```

Call:

```
glm(formula = y ~ log(dose) + sex, family = binomial, data = moth)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.10540	-0.65343	-0.02225	0.48471	1.42944

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.2717      0.5753  -2.211  0.02707 *
log(dose)    1.5353      0.1891   8.119  4.7e-16 ***
sex          -1.1007     0.3558  -3.093  0.00198 **
---

```

```
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
```

```
(Dispersion parameter for binomial family taken to be 1)
```

```

Null deviance: 124.876 on 11 degrees of freedom
Residual deviance: 6.757 on 9 degrees of freedom
AIC: 42.867

```

```
Number of Fisher Scoring iterations: 4
```

```

> anova(g1,test="Chisq")
Analysis of Deviance Table

```

```
Model: binomial, link: logit
```

```
Response: y
```

```
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	P(> Chi )
NULL			11	124.876	
log(dose)	1	107.892	10	16.984	2.839e-25
sex	1	10.227	9	6.757	0.001

```
> names(g1)
```

```

 [1] "coefficients"      "residuals"          "fitted.values"      "effects"             "R"
 [6] "rank"              "qr"                 "family"             "linear.predictors"  "deviance"
[11] "aic"              "null.deviance"     "iter"              "weights"            "prior.weights"
[16] "df.residual"      "df.null"           "y"                 "converged"         "boundary"
[21] "model"            "call"              "formula"           "terms"              "data"
[26] "offset"          "control"           "method"            "contrasts"         "xlevels"

```

```
> g1$fitted
```

```

      1      2      3      4      5      6      7      8
0.08530076 0.21278854 0.43930479 0.69428515 0.86812073 0.95020002 0.03008577 0.08249341 0.2067337
      11      12
0.68647712 0.86388206

```

```
> plot(moth$dose,p)
```

```
> points(moth$dose,g1$fitted,pch="+")
```

```
> title(main="moth fit")
```

## 9 Reshapeing data

We often wish to reshape or edit data in order to perform some favorite analysis. In this section we look at some of the possibilities. `edit()` is a useful if somewhat limited editor for data already in R. This can be changed see `options()` and `options(editor="my_favorite_editor")`.

Sometimes spreadsheet data is in a compact format that gives the covariates for each subject followed by all the observations on that subject. R's modelling functions need observations in a single column. Consider the following sample of data from repeated MRI brain measurements

```
Status  Age   V1    V2    V3    V4
   P 23646 45190 50333 55166 56271
   CC 26174 35535 38227 37911 41184
   CC 27723 25691 25712 26144 26398
   CC 27193 30949 29693 29754 30772
   CC 24370 50542 51966 54341 54273
   CC 28359 58591 58803 59435 61292
   CC 25136 45801 45389 47197 47126
```

There are two covariates and up to four measurements on each subject. The data were exported from Excel as a file `mr.csv`.

We can use `stack` to help manipulate these data to give a single response.

```
zz <- read.csv("mr.csv", strip.white = TRUE)
zzz <- cbind(zz[gl(nrow(zz), 1, 4*nrow(zz)), 1:2], stack(zz[, 3:6]))
with result
```

```
      Status  Age values ind
X1      P 23646 45190 V1
X2      CC 26174 35535 V1
X3      CC 27723 25691 V1
X4      CC 27193 30949 V1
X5      CC 24370 50542 V1
X6      CC 28359 58591 V1
X7      CC 25136 45801 V1
X11     P 23646 50333 V2
...

```

Function `unstack` goes in the opposite direction, and may be useful for exporting data. Another way to do this is to use the function `reshape`, by

```
> reshape(zz, idvar="id",timevar="var",
  varying=list(c("V1","V2","V3","V4")),direction="long")
  Status  Age var  V1 id
1.1     P 23646  1 45190 1
2.1     CC 26174  1 35535 2
3.1     CC 27723  1 25691 3
4.1     CC 27193  1 30949 4
5.1     CC 24370  1 50542 5
6.1     CC 28359  1 58591 6
7.1     CC 25136  1 45801 7
1.2     P 23646  2 50333 1
```

```
2.2      CC 26174   2 38227   2
...

```

The reshape has a more complicated syntax than stack but can be used for data where the ‘long’ form has more than the one column in this example. With `direction="wide"`, reshape can also perform the opposite transformation.

Displaying higher-dimensional contingency tables in array form typically is rather inconvenient. In categorical data analysis, such information is often represented in the form of bordered two-dimensional arrays with leading rows and columns specifying the combination of factor levels corresponding to the cell counts. These rows and columns are typically ragged in the sense that labels are only displayed when they change, with the obvious convention that rows are read from top to bottom and columns are read from left to right. In R, such flat contingency tables can be created using `fTable`, which creates objects of class "fTable" with an appropriate print method.

As a simple example, consider the R standard data set `UCBAdmissions` which is a 3-dimensional contingency table resulting from classifying applicants to graduate school at UC Berkeley for the six largest departments in 1973 classified by admission and sex.

```
> data(UCBAdmissions)
> fTable(UCBAdmissions)
      Dept  A  B  C  D  E  F
Admit  Gender
Admitted Male    512 353 120 138 53 22
        Female    89 17 202 131 94 24
Rejected Male    313 207 205 279 138 351
        Female    19 8 391 244 299 317

```

The printed representation is clearly more useful than displaying the data as a 3-dimensional array.

There is also a function `read.fTable` for reading in flat-like contingency tables from files. This has additional arguments for dealing with variants on how exactly the information on row and column variables names and levels is represented. The help page for `read.fTable` has some useful examples. The flat tables can be converted to standard contingency tables in array form using `as.table`.

Note that flat tables are characterized by their ragged display of row (and maybe also column) labels. If the full grid of levels of the row variables is given, one should instead use `read.table` to read in the data, and create the contingency table from this using `xtabs`.

You should not lose sight of the fact that you can use R indexing to manipulate data. So you can look at the babies table and change the 3rd reading of column 1 to 6000 as follows

```
babies[3,1]<-6000
```

We know that the boys are the first 12 rows and can pick them out boys and girls as

```
boys<-babies[1:12,]
girls<-babies[13:24,]
```

We can get summaries which avoid looping using `tapply` and `lapply`.

```
> tapply(babies$weight,babies$gender,mean)
      f      m
2911.333 3024.000
> tapply(babies$weight,babies$gender,sd)
      f      m
280.4423 284.2189
```

We can also use the ifelse construct

```
> y
[1]  6 14 -3 -8 14 13  0 -12 -17 -15 14 -3 11 15 -6 -10
[17] 24 21  0 -12 -25 -19 26  5  5 28 -15  9  6 21  0 -15
[33] -22 -16 20  5  9 13 -12  2 35 12 12 -33 -18 -19 22  2
[49]  0 40
> z<-ifelse(y>0,1,0)# z is 1 if the value of y exceeds zero otherwise it is 0
> z
[1] 1 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0
[34] 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1
>
```

## 9.1 An Ecological example

We steal this example from

<http://ecology.msu.montana.edu/labdsv/R/>

a site which introduces  $\mathbb{R}$  to ecologists. site `j<-read.table('brycesit.s',header=TRUE)` Bryce Canyon National Park is a 14000 ha park in southern Utah with vegetation representative of a transition from the cordilleran flora of the central Rocky Mountains to the Colorado Plateau. The Park vegetation ranges from small areas of badlands and sagebrush steppe at low elevations through an extensive pinyon-juniper woodland, to ponderosa pine savanna at mid-elevations, reaching closed mixed conifer forest at high elevations.

The idea is we to model the distribution of *Berberis* (*Mahonia*) repens, a common shrub in mesic portions of Bryce Canyon. Preliminary graphical analysis (shown below, *Berberis* locations in red) suggests that the distribution of *Berberis repens* is related to elevation and possibly aspect value. He gives two data files

1. `bryceveg.s` which contains all the vegetation data
2. `brycesit.s` which has the environmental or site data for the same plots

We read these into data tables

```
> site <- read.table(file.choose(),header=TRUE)
> veg <- read.table(file.choose(),header=TRUE)
# being lazy I have combined them to give d
> d<-cbind(site,veg)
> rm(site) # removes the site table
> rm(veg) # removes the site table
```

The web page authour suggests that the distribution of *Berberis repens* is related to elevation `elev` and possibly aspect value `av`. To investigate I plot the `av` values against the `elev` values and at each point I draw a bubble whose size is proportional to the variable of interest `berrep`.

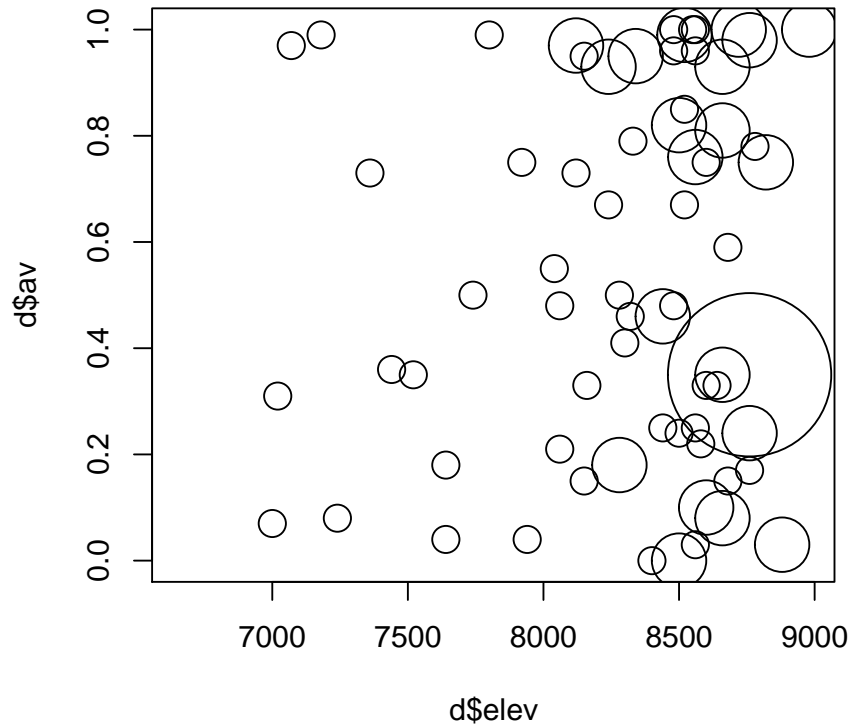


Figure 4: bub

```
plot(d$elev,d$av,cex=(4*d$berrep))# 4 seems a good multiplier - but you might prefer another
```

the plot, in figure ?? looks interesting You might prefer a filled set of bubbles as in figure 5

```
plot(d$elev,d$av,cex=(3*d$berrep),pch=19,col=2)
```

To model this we start by just looking at the presence or otherwise of the plant.

```
> y<-d$berrep>0 # indicate if bberis exists
> y
 [1] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE
 [12] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
 [23] FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
 [34] TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE
 [45] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
 [56] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

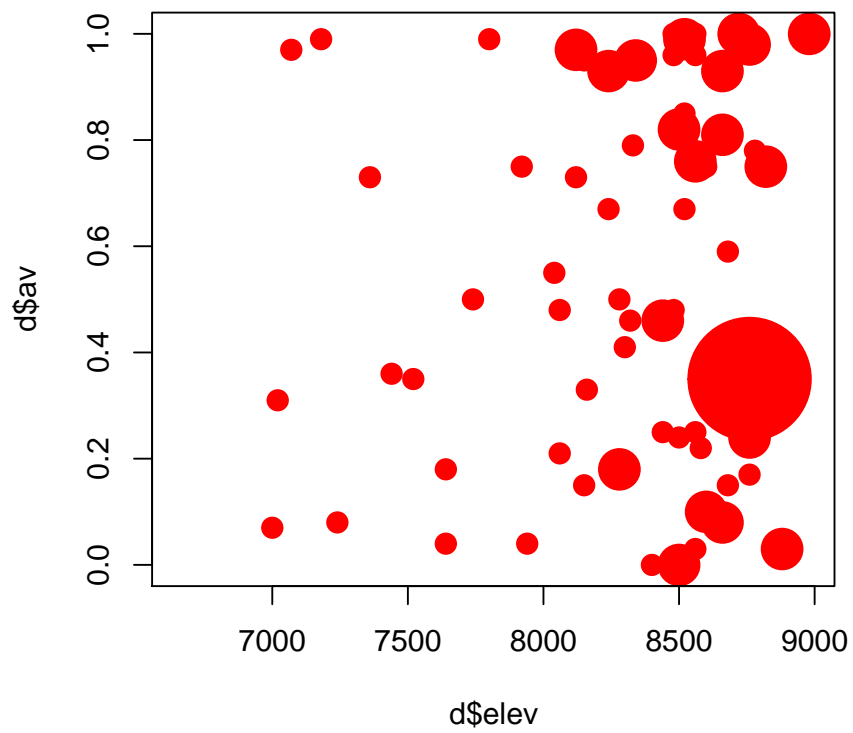


Figure 5: Filled bubbles using pch=21

```
[67] TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE FALSE TRUE TRUE
[78] TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[155] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
> bere1.glm <- glm(y ~ elev, family = binomial, data=d) # Fit model
summary(bere1.glm)
```

Call:

```
glm(formula = y ~ elev, family = binomial, data = d)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.1110	-0.7981	-0.2707	0.6862	2.4500

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-2.265e+01	3.593e+00	-6.302	2.93e-10 ***
elev	2.814e-03	4.484e-04	6.275	3.49e-10 ***

Signif. codes: 0 \*\*\* 0.001 \*\* 0.01 \* 0.05 . 0.1 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 218.19 on 159 degrees of freedom  
 Residual deviance: 151.95 on 158 degrees of freedom  
 AIC: 155.95

Number of Fisher Scoring iterations: 5

```
> anova(bere1.glm, test="Chisq")
Analysis of Deviance Table
```

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

	Df	Deviance	Resid. Df	Resid. Dev	P(> Chi )
NULL			159	218.193	
elev	1	66.247	158	151.947	3.979e-16

```
> plot(elev, bere1.glm$resid)
> abline(h=0, col=3)
```

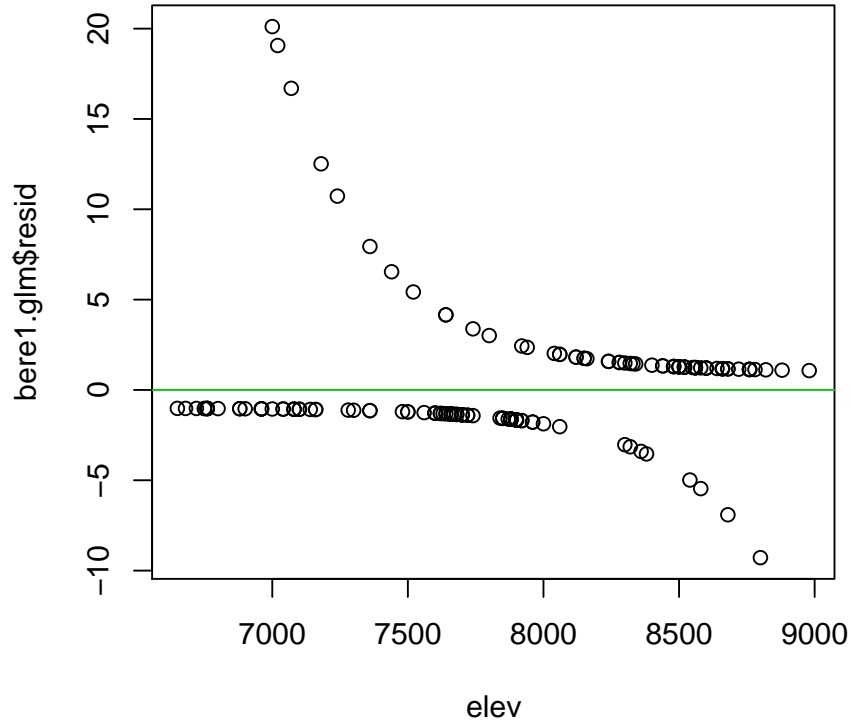


Figure 6: berberis loglinear residuals

We have plotted the residuals in 6. Now I am a little reluctant to complicate this model but we could try

```
bere2.glm<-glm(y~elev+I(elev^2)+av+I(av^2),family=binomial)
```

I think it is pushing your luck but

```
> summary(bere2.glm)
```

Call:

```
glm(formula = y ~ elev + I(elev^2) + av + I(av^2), family = binomial,
     data = d)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.3801	-0.7023	-0.4327	0.5832	2.3540

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.831e+01  4.016e+01   1.950  0.0512 .
elev         -2.324e-02  1.039e-02  -2.236  0.0254 *
I(elev^2)    1.665e-06  6.695e-07   2.487  0.0129 *
av           4.385e+00  2.610e+00   1.680  0.0929 .
I(av^2)     -4.447e+00  2.471e+00  -1.799  0.0719 .
---

```

```
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
```

(Dispersion parameter for binomial family taken to be 1)

```

Null deviance: 218.19  on 159  degrees of freedom
Residual deviance: 143.19  on 155  degrees of freedom
AIC: 153.19

```

Number of Fisher Scoring iterations: 5

```
> anova(bere2.glm,test="Chisq")
Analysis of Deviance Table
```

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

```

              Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                159    218.193
elev              1    66.247      158    151.947 3.979e-16
I(elev^2)         1     5.320      157    146.627  0.021
av                1     0.090      156    146.537  0.765
I(av^2)           1     3.349      155    143.188  0.067
>

```

```
> avc<-ifelse(d$av>.5,1,0)+1
> plot(elev,bere2.glm$resid,col=avc}
```

I shall leave you to pick the bones out of this.

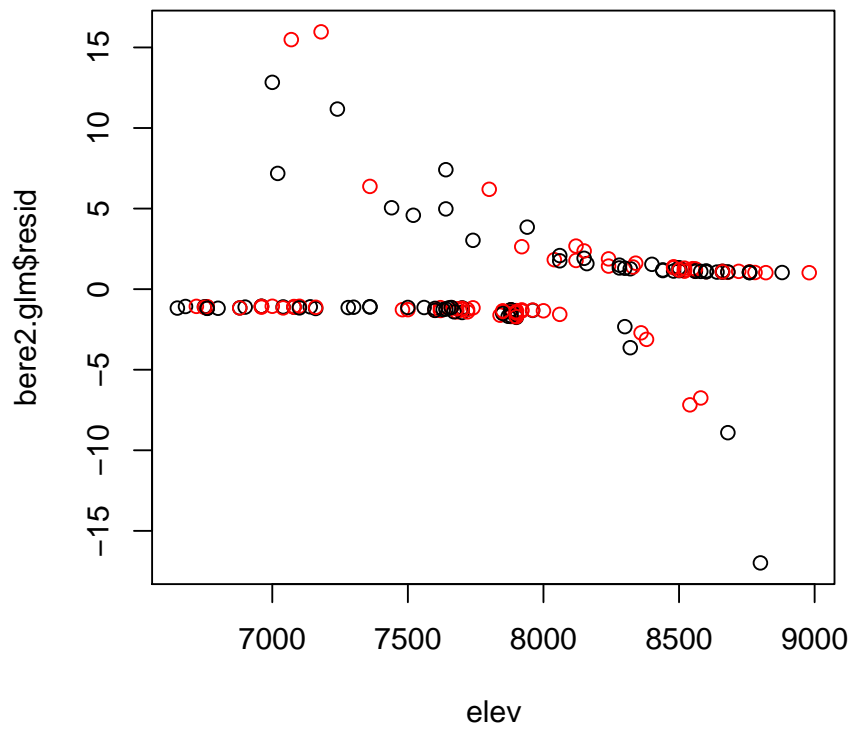


Figure 7: berberis loglinear residuals

## 10 Exercises

1. Generate a sample of size 10 from a normal distribution and find the order statistics.  
Commands : `rnorm`, `sort`
2. The minimum from a sample of size  $k$  from a standard normal distribution can be obtained by `min(rnorm(12))`. If you define `t` to be a vector of 500 then you can generate 500 minimums from a normal distribution by

```
t<-seq(500)
for (i in (1:500)){t[i]<-min(rnorm(12))}
```

Does the distribution of the minimum look normal? You might find `plot(density(t))` and `density` useful.

Repeat the exercise for the maximum of a uniform distribution.

Commands `runif`, `max`

3. Generate the order statistics from a random sample of size  $N$  taken from a uniform distribution. Plot  $j/N$  vs  $x_j$ . Now generate the order statistics from a random sample of size  $N$  from a normal distribution. Plot  $j/N$  vs  $x_j$ .

Now let  $y_j = \Phi^{-1}(x_j)$  where  $\Phi(x)$  is the cumulative distribution function of a standard normal variate.

Plot  $j/N$  vs  $y_j$ .

Repeat the above when  $y_j = \Phi^{-1}(u_j)$  where  $u_j$  are order statistics from a uniform distribution which the same mean and variance as the original  $X$  distribution.

Commands `rnorm`, `pnorm`, `qnorm`

4. Which has the smaller variance for an exponential distribution. The sample mean or the sample median?  
Commands `rexp`
5. The table below gives the total catch of anchovies off Chile and the price ( per tonne) of the catch in 1965 US dollars.

Year	1965	1966	1967	1968	1969	1970	1971
Price	190	160	134	129	172	197	167
Catch	7.23	8.53	9.82	10.96	8.96	12.27	10.28
Year	1972	1973	1974	1975	1976	1977	1978
Price	239	542	372	245	376	454	410
Catch	4.45	1.78	4	3.3	4.3	.08	.5

Compute the correlations between the variables - try `?cor`. Can one say that the correlation between price and catch is non-significant?

6. The command

```
x<-seq(30)
```

generates a vector containing the numbers 1 to 30. Suppose we now try

```
y<-2+11*x+rnorm(30,0,34)
```

What do you expect to be the relationship between (x) and (y)?

The command

```
r1<-lm(y~x)
```

produces a regression the results of which are held in `r1`. Try `summary(r1)` and compare the results with what you expect. Try

```
plot(x,y)
lines(x,r1$fitted)
```

Examine the residuals `r1$resid`

7. The data at <http://www.uea.ac.uk/~gj/nunion/wool.dat> gives the number of cycles to failure of wool yarn. Find a linear regression model for this data. How can you examine the fit of your model?

## 11 Fitting Linear Models: lm R Documentation

### 11.1 Description

lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although aov may provide a more convenient interface for these).

#### 11.1.1 Usage

```
lm(formula, data, subset, weights, na.action,  
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,  
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

#### 11.1.2 Arguments

- **formula** - a symbolic description of the model to be fit. The details of model specification are given below.
- **data** - an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
- **subset** - an optional vector specifying a subset of observations to be used in the fitting process.
- **weights** - an optional vector of weights to be used in the fitting process. If specified, weighted least squares is used with weights weights (that is, minimizing  $\sum(w * e^2)$ ); otherwise ordinary least squares is used.
- **na.action** - a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The factory-fresh default is **na.omit**. Another possible value is NULL, no action.
- **method** - the method to be used; for fitting, currently only method = "qr" is supported; method = "model.frame" returns the model frame (the same as with model = TRUE, see below). model, x, y, qr logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
- **singular** - .ok logical. If FALSE (the default in S but not in R) a singular fit is an error. contrasts an optional list. See the contrasts.arg of model.matrix.default.
- **offset** - this can be used to specify an a priori known component to be included in the linear predictor during fitting. An offset term can be included in the formula instead or as well, and if both are specified their sum is used.
- ... additional arguments to be passed to the low level regression fitting functions (see below).

### 11.1.3 Details

Models for `lm` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form

`first + second`

indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form

`first:second`

indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification

`first*second`

indicates the cross of `first` and `second`. This is the same as

`first + second + first:second`.

If `response` is a matrix a linear model is fitted to each column of the matrix. See `model.matrix` for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

A formula has an implied intercept term. To remove this use either  $y \sim x - 1$  or  $y \sim 0 + x$ . See `formula` for more details of allowed formulae.

`lm` calls the lower level functions `lm.fit`, etc, see below, for the actual numerical computations. For programming only, you may consider doing likewise.

All of `weights`, `subset` and `offset` are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.

### 11.1.4 Value

`lm` returns an object of class "lm" or for multiple responses of class `c("mlm", "lm")`. The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`. An object of class "lm" is a list containing at least the following components:

- `coefficients` a named vector of coefficients
- `residuals` the residuals, that is response minus fitted values.
- `fitted.values` the fitted mean values.
- `rank` the numeric rank of the fitted linear model.
- `weights` (only for weighted fits) the specified weights.
- `df.residual` the residual degrees of freedom.
- `call` the matched call.
- `terms` the terms object used.
- `contrasts` (only where relevant) the contrasts used.
- `xlevels` (only where relevant) a record of the levels of the factors used in fitting.
- `y` if requested, the response used.

- `x` if requested, the model matrix used.
- `model` if requested (the default), the model frame used.

In addition, non-null fits will have components `assign`, `effects` and (unless not requested) `qr` relating to the linear fit, for use by extractor functions such as `summary` and `effects`.

### 11.1.5 Using time series

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting NAs would invalidate the time series attributes, and if NAs are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or differenced regressor would be ignored. It is good practice to prepare a data argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and fitted values are time series.

### 11.1.6 Note

Offsets specified by `offset` will not be included in predictions by `predict.lm`, whereas those specified by an offset term in the formula will be.

### 11.1.7 Author(s)

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson and Rogers (1973).

### 11.1.8 Examples

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
anova(lm.D9 <- lm(weight ~ group))
summary(lm.D90 <- lm(weight ~ group - 1))# omitting intercept
summary(resid(lm.D9) - resid(lm.D90)) #- residuals almost identical

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

## model frame :
stopifnot(identical(lm(weight ~ group, method = "model.frame"),
                    model.frame(lm.D9)))
```

## 12 Fitting Generalized Linear Models: glm stats R Documentation

### 12.0.9 Description

`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

### 12.0.10 Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart,
    offset, control = glm.control(...), model = TRUE,
    method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)
```

```
glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = glm.control(), intercept = TRUE)
```

```
## S3 method for class 'glm':
weights(object, type = c("prior", "working"), ...)
```

### 12.0.11 Arguments

- **formula** - a symbolic description of the model to be fit. The details of model specification are given below.
- **family** - a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)
- **data** - an optional data frame containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `glm` is called.
- **weights** - an optional vector of weights to be used in the fitting process.
- **subse** -t an optional vector specifying a subset of observations to be used in the fitting process.
- **na.action** - a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of options, and is `na.fail` if that is unset. The factory-fresh default is `na.omit`.
- **start** -starting values for the parameters in the linear predictor.
- **etastart** - starting values for the linear predictor.
- **mustart** - starting values for the vector of means.

- `offset` - this can be used to specify an a priori known component to be included in the linear predictor during fitting.
- `control` - a list of parameters for controlling the fitting process. See the documentation for `glm.control` for details.
- `model` - a logical value indicating whether model frame should be included as a component of the returned value.
- `method` - the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). The only current alternative is "model.frame" which returns the model frame and does no fitting.
- `x`, `y` - For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension  $n * p$ , and `y` is a vector of observations of length `n`.
- `contrasts` -an optional list. See the `contrasts.arg` of `model.matrix.default`.
- `object` - an object inheriting from class "glm".
- `type` - character, partial matching allowed. Type of weights to extract from the fitted model object.
- `intercept` - logical. Should an intercept be included?
- `...` - further arguments passed to or from other methods.

### 12.0.12 Details

A typical predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for response. For binomial models the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form

`first + second`

indicates all the terms in `first` together with all the terms in `second` with duplicates removed. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

A specification of the form

`first:second`

indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification

`first*second`

indicates the cross of `first` and `second`. This is the same as

`first + second + first:second`.

`glm.fit` and `glm.fit.null` are the workhorse functions: the former calls the latter for a null model (with no intercept).

If more than one of `etastart`, `start` and `mustart` is specified, the first in the list will be used.

All of `weights`, `subset`, `offset`, `etastart` and `mustart` are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.

### 12.0.13 Value

`glm` returns an object of class inheriting from "glm" which inherits from the class "lm". See later in this section. The function `summary` (i.e., `summary.glm`) can be used to obtain or print a summary of the results and the function `anova` (i.e., `anova.glm`) to produce an analysis of variance table. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` can be used to extract various useful features of the value returned by `glm`. `weights` extracts a vector of weights, one for each case in the fit (after subsetting and `na.action`). An object of class "glm" is a list containing at least the following components:

- `coefficients` - a named vector of coefficients
- `residuals` - the working residuals, that is the residuals in the final iteration of the IWLS fit.
- `fitted.values` - the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
- `rank` - the numeric rank of the fitted linear model.
- `family` - the family object used.
- `linear.predictors` - the linear fit on link scale.
- `deviance` - up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
- `aic` - Akaike's An Information Criterion, minus twice the maximized log-likelihood plus twice the number of coefficients (so assuming that the dispersion is known).
- `null.deviance` - The deviance for the null model, comparable with deviance. The null model will include the offset, and an intercept if there is one in the model
- `iter` - the number of iterations of IWLS used.
- `weights` - the working weights, that is the weights in the final iteration of the IWLS fit.
- `prior.weights` - the case weights initially supplied.
- `df.residual` - the residual degrees of freedom.
- `df.null` - the residual degrees of freedom for the null model.
- `y` - the y vector used. (It is a vector even for a binomial model.)
- `converged` - logical. Was the IWLS algorithm judged to have converged?
- `boundary` - logical. Is the fitted value on the boundary of the attainable values?
- `call` - the matched call.
- `formula` - the formula supplied.
- `terms` - the terms object used.

- `data` - the data argument.
- `offset` - the offset vector used.
- `control` - the value of the control argument used.
- `method` - the name of the fitter function used, in R always "glm.fit". contrasts (where relevant) the contrasts used.
- `xlevels` - (where relevant) a record of the levels of the factors used in fitting.

In addition, non-empty fits will have components `qr`, `R` and effects relating to the final weighted linear fit. Objects of class "glm" are normally of class `c("glm", "lm")`, that is inherit from class "lm", and well-designed methods for class "lm" will be applied to the weighted linear model at the final iteration of IWLS. However, care is needed, as extractor functions for class "glm" such as `residuals` and `weights` do not just pick out the component of the fit with the same name. If a binomial glm model is specified by giving a two-column response, the weights returned by `prior.weights` are the total numbers of cases (factored by the supplied case weights) and the component `y` of the result is the proportion of successes.

#### 12.0.14 Author(s)

The original R implementation of `glm` was written by Simon Davies working for Ross Ihaka at the University of Auckland, but has since been extensively re-written by members of the R Core team.

The design was inspired by the S function of the same name described in Hastie and Pregibon (1992).

#### 12.0.15 Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family=poisson())
anova(glm.D93)
summary(glm.D93)

## an example with offsets from Venables & Ripley (2002, p.189)

## Not run:
## Need the anorexia data from a recent version of the package 'MASS':
library(MASS)
## End(Not run)
anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
               family = gaussian, data = anorexia)
summary(anorex.1)

# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
```

```
u = c(5,10,15,20,30,40,60,80,100),
lot1 = c(118,58,42,35,27,25,21,19,18),
lot2 = c(69,35,26,21,18,16,13,12,12))
summary(glm(lot1 ~ log(u), data=clotting, family=Gamma))
summary(glm(lot2 ~ log(u), data=clotting, family=Gamma))

## Not run:
## for an example of the use of a terms object as a formula
demo(glm.vr)
## End(Not run)
[Package stats version 2.0.1 Index]
```

## 13 Random Number generators

1. `runif(n, min=0, max=1)`  
Gives a vector of length `n` whose elements are random and have a uniform distribution between `min` and `max`.
2. `rnorm(n, mean=0, sd=1)`  
Gives a vector of length `n` whose elements are random and have a Normal distribution mean zero and standard deviation 1.
3. `rgamma(n, shape, scale=1)`  
Gives a vector of length `n` whose elements are random and have a Gamma distribution with `scale = 1`.
4. `rbinom(n, size, prob)`  
Gives a vector of length `n` whose elements are random and have a Binomial distribution with probability `prob` and size `size`.
5. `rpois(n, lambda)`  
Gives a vector of length `n` whose elements are random and have a Poisson distribution mean `1/lambda`.
6. `rcauchy(n, location = 0, scale = 1)`  
Gives a vector of length `n` whose elements are random and have a Cauchy distribution parameters `location` and `scale`

## 14 Publications related to R - from CRAN

This page gives a partially annotated list of books and other publications that are related to S or R and may be useful - not this is taken from CRAN.

1. RichardA. Becker, JohnM. Chambers, and AllanR. Wilks. *The New S Language*. Chapman and Hall, London, 1988.  
This book is often called the Blue Book, and introduced what is now known as S version 2.
2. JohnM. Chambers and TrevorJ. Hastie. *Statistical Models in S*. Chapman and Hall, London, 1992.  
This is also called the White Book, and introduced S version 3, which added structures to facilitate statistical modeling in S.
3. JohnM. Chambers. *Programming with Data*. Springer, New York, 1998. ISBN 0-387-98503-4.  
This Green Book describes version 4 of S, a major revision of S designed by John Chambers to improve its usefulness at every stage of the programming process.
4. WilliamN. Venables and BrianD. Ripley. *Modern Applied Statistics with S*. Fourth Edition. Springer, New York, 2002. ISBN 0-387-95457-0.  
A highly recommended book on how to do statistical data analysis using R or S-Plus. In the first chapters it gives an introduction to the S language. Then it covers a wide range of statistical methodology, including linear and generalized linear models, non-linear and smooth regression, tree-based methods, random and mixed effects, exploratory multivariate analysis, classification, survival analysis, time series analysis, spatial statistics, and optimization. The 'on-line complements' available at the books homepage provide updates of the book, as well as further details of technical material.
5. WilliamN. Venables and BrianD. Ripley. *S Programming*. Springer, New York, 2000. ISBN 0-387-98966-8.  
This provides an in-depth guide to writing software in the S language which forms the basis of both the commercial S-Plus and the Open Source R data analysis software systems.
6. Deborah Nolan and Terry Speed. *Stat Labs: Mathematical Statistics Through Applications*. Springer Texts in Statistics. Springer, 2000. ISBN 0-387-98974-9.  
Integrates theory of statistics with the practice of statistics through a collection of case studies (labs), and uses R to analyze the data.
7. JoseC. Pinheiro and DouglasM. Bates. *Mixed-Effects Models in S and S-Plus*. Springer, 2000. ISBN 0-387-98957-0.  
A comprehensive guide to the use of the 'nlme' package for linear and nonlinear mixed-effects models.
8. FrankE. Harrell. *Regression Modeling Strategies, with Applications to Linear Models, Survival Analysis and Logistic Regression*. Springer, 2001. ISBN 0-387-95232-2.  
There are many books that are excellent sources of knowledge about individual statistical tools (survival models, general linear models, etc.), but the art of data analysis is about choosing and using multiple tools. In the words of Chatfield ...students typically know the technical details of regression for example, but not necessarily when and how to apply it. This argues the need for a better balance in the literature and in statistical teaching between

techniques and problem solving strategies. Whether analyzing risk factors, adjusting for biases in observational studies, or developing predictive models, there are common problems that few regression texts address. For example, there are missing data in the majority of datasets one is likely to encounter (other than those used in textbooks!) but most regression texts do not include methods for dealing with such data effectively, and texts on missing data do not cover regression modeling.

9. Manuel Castejón Limas, Joaquín Ordieres Mer, Fco. Javier de Cos Juez, and Fco. Javier Martínez de Pisuñascibar. *Control de Calidad. Metodología para el análisis previo a la modelización de datos en procesos industriales. Fundamentos teóricos y aplicaciones con R*. Servicio de Publicaciones de la Universidad de La Rioja, 2001. ISBN 84-95301-48-2.  
This book, written in Spanish, is oriented to researchers interested in applying multivariate analysis techniques to real processes. It combines the theoretical basis with applied examples coded in R.
10. John Fox. *An R and S-Plus Companion to Applied Regression*. Sage Publications, Thousand Oaks, CA, USA, 2002. ISBN 0-761-92279-2.  
A companion book to a text or course on applied regression (such as *Applied Regression, Linear Models, and Related Methods* by the same author). It introduces S, and concentrates on how to use linear and generalized-linear models in S while assuming familiarity with the statistical methodology.
11. Peter Dalgaard. *Introductory Statistics with R*. Springer, 2002. ISBN 0-387-95475-9.
12. John Maindonald and John Braun. *Data Analysis and Graphics Using R*. Cambridge University Press, Cambridge, 2003. ISBN 0-521-81336-0.
13. Giovanni Parmigiani, Elizabeth S. Garrett, Rafael A. Irizarry, and Scott L. Zeger. *The Analysis of Gene Expression Data*. Springer, New York, 2003. ISBN 0-387-95577-1
14. Sylvie Huet, Annie Bouvier, Marie-Anne Gruet, and Emmanuel Jolivet. *Statistical Tools for Nonlinear Regression*. Springer, New York, 2003. ISBN 0-387-40081-8
15. S. Mase, T. Kamakura, M. Jimbo, and K. Kanefuji. *Introduction to Data Science for engineers-Data analysis using free statistical software R (in Japanese)*. Suuri-Kogaku-sha, Tokyo, April 2004. ISBN 4901683128
16. Julian J. Faraway. *Linear Models with R*. Chapman and Hall/CRC, Boca Raton, FL, 2004. ISBN 1-584-88425-8  
The book focuses on the practice of regression and analysis of variance. It clearly demonstrates the different methods available and in which situations each one applies. It covers all of the standard topics, from the basics of estimation to missing data, factorial designs, and block designs, but it also includes discussion of topics, such as model uncertainty, rarely addressed in books of this type. The presentation incorporates an abundance of examples that clarify both the use of each technique and the conclusions one can draw from the results.
17. Richard M. Heiberger and Burt Holland. *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer Texts in Statistics. Springer, 2004. ISBN 0-387-40270-5  
A contemporary presentation of statistical methods featuring 200 graphical displays for exploring data and displaying analyses. Many of the displays appear here for the first

time. Discusses construction and interpretation of graphs, principles of graphical design, and relation between graphs and traditional tabular results. Can serve as a graduate-level standalone statistics text and as a reference book for researchers. In-depth discussions of regression analysis, analysis of variance, and design of experiments are followed by introductions to analysis of discrete bivariate data, nonparametrics, logistic regression, and ARIMA time series modeling. Concepts and techniques are illustrated with a variety of case studies. S-Plus, R, and SAS executable functions are provided and discussed. S functions are provided for each new graphical display format. All code, transcript and figure files are provided for readers to use as templates for their own analyses.

18. John Verzani. *Using R for Introductory Statistics*. Chapman and Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88450-9  
There are few books covering introductory statistics using R, and this book fills a gap as a true beginner book. With emphasis on data analysis and practical examples, 'Using R for Introductory Statistics' encourages understanding rather than focusing on learning the underlying theory. It includes a large collection of exercises and numerous practical examples from a broad range of scientific disciplines. It comes complete with an online resource containing datasets, R functions, selected solutions to exercises, and updates to the latest features. A full solutions manual is available from Chapman and Hall/CRC.
19. Fionn Murtagh. *Correspondence Analysis and Data Coding with JAVA and R*. Chapman and Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88528-9  
This book provides an introduction to methods and applications of correspondence analysis, with an emphasis on data coding - the first step in correspondence analysis. It features a practical presentation of the theory with a range of applications from data mining, financial engineering, and the biosciences. Implementation of the methods is presented using JAVA and R software.
20. Paul Murrell. *R Graphics*. Chapman and Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88486-X  
A description of the core graphics features of R including: a brief introduction to R; an introduction to general R graphics features. The base graphics system of R: traditional S graphics. The power and flexibility of grid graphics. Building on top of the base or grid graphics: Trellis graphics and developing new graphics functions.
21. Michael J. Crawley. *Statistics: An Introduction using R*. Wiley, 2005. ISBN 0-470-02297-3  
The book is primarily aimed at undergraduate students in medicine, engineering, economics and biology - but will also appeal to postgraduates who have not previously covered this area, or wish to switch to using R.
22. Brian S. Everitt. *An R and S-Plus Companion to Multivariate Analysis*. Springer, 2005. ISBN 1-85233-882-2  
In this book the core multivariate methodology is covered along with some basic theory for each method described. The necessary R and S-Plus code is given for each analysis in the book, with any differences between the two highlighted.
23. Richard C. Deonier, Simon Tavar, and Michael S. Waterman. *Computational Genome Analysis: An Introduction*. Springer, 2005. ISBN: 0-387-98785-1  
Computational Genome Analysis: An Introduction presents the foundations of key problems in computational molecular biology and bioinformatics. It focuses on computational

and statistical principles applied to genomes, and introduces the mathematics and statistics that are crucial for understanding these applications. All computations are done with R.

24. Robert Gentleman, Vince Carey, Wolfgang Huber, Rafael Irizarry, and Sandrine Dudoit, editors. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Statistics for Biology and Health. Springer, 2005. ISBN: 0-387-25146-4  
This volume's coverage is broad and ranges across most of the key capabilities of the Bioconductor project, including importation and preprocessing of high-throughput data from microarray, proteomic, and flow cytometry platforms.
25. Terry M. Therneau and Patricia M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Statistics for Biology and Health. Springer, 2000. ISBN: 0-387-98784-3  
This is a book for statistical practitioners, particularly those who design and analyze studies for survival and event history data. Its goal is to extend the toolkit beyond the basic triad provided by most statistical packages: the Kaplan-Meier estimator, log-rank test, and Cox regression model.
26. Brian Everitt and Torsten Hothorn. *A Handbook of Statistical Analyses Using R*. Chapman and Hall/CRC, Boca Raton, FL, 2006. ISBN 1-584-88539-4  
With emphasis on the use of R and the interpretation of results rather than the theory behind the methods, this book addresses particular statistical techniques and demonstrates how they can be applied to one or more data sets using R. The authors provide a concise introduction to R, including a summary of its most important features. They cover a variety of topics, such as simple inference, generalized linear models, multilevel models, longitudinal data, cluster analysis, principal components analysis, and discriminant analysis. With numerous figures and exercises, *A Handbook of Statistical Analysis using R* provides useful information for students as well as statisticians and data analysts.
27. Julian J. Faraway. *Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Chapman and Hall/CRC, Boca Raton, FL, 2006. ISBN 1-584-88424-X  
This book surveys the techniques that grow from the regression model, presenting three extensions to that framework: generalized linear models (GLMs), mixed effect models, and nonparametric regression models. The author's treatment is thoroughly modern and covers topics that include GLM diagnostics, generalized linear mixed models, trees, and even the use of neural networks in statistics. To demonstrate the interplay of theory and practice, throughout the book the author weaves the use of the R software environment to analyze the data of real examples, providing all of the R commands necessary to reproduce the analyses.
28. Jana Jureckova and Jan Picek. *Robust Statistical Methods with R*. Chapman and Hall/CRC, Boca Raton, FL, 2006. ISBN 1-584-88454-1  
This book provides a systematic treatment of robust procedures with an emphasis on practical application. The authors work from underlying mathematical tools to implementation, paying special attention to the computational aspects. They cover the whole range of robust methods, including differentiable statistical functions, distance of measures, influence functions, and asymptotic distributions, in a rigorous yet approachable manner. Highlighting hands-on problem solving, many examples and computational algorithms using the R software supplement the discussion. The book examines the characteristics of robustness, estimators of real parameter, large sample properties, and goodness-of-fit tests. It also

includes a brief overview of R in an appendix for those with little experience using the software.

29. Simon N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, Boca Raton, FL, 2006. ISBN 1-584-88474-6  
This book imparts a thorough understanding of the theory and practical applications of GAMs and related advanced models, enabling informed use of these very flexible tools. The author bases his approach on a framework of penalized regression splines, and builds a well-grounded foundation through motivating chapters on linear and generalized linear models. While firmly focused on the practical aspects of GAMs, discussions include fairly full explanations of the theory underlying the methods. The treatment is rich with practical examples, and it includes an entire chapter on the analysis of real data sets using R and the author's add-on package `mgcv`. Each chapter includes exercises, for which complete solutions are provided in an appendix.
30. Bernhard Pfaff. *Analysis of Integrated and Cointegrated Time Series with R*. Use R. Springer, 2006. ISBN 0-387-98784-3  
The book encompasses seasonal unit roots, fractional integration, coping with structural breaks, and inference in cointegrated vector autoregressive models.
31. Nhu D. Le and James V. Zidek. *Statistical Analysis of Environmental Space-Time Processes*. Springer, 2006. ISBN 0-387-26209-1  
This book provides a broad introduction to the subject of environmental space-time processes, addressing the role of uncertainty. It covers a spectrum of technical matters from measurement to environmental epidemiology to risk assessment. It showcases non-stationary vector-valued processes, while treating stationarity as a special case. In particular, with members of their research group the authors developed within a hierarchical Bayesian framework, the new statistical approaches presented in the book for analyzing, modeling, and monitoring environmental spatio-temporal processes. Furthermore they indicate new directions for development.
32. Peter J. Diggle and Paulo Justiniano Ribeiro. *Model-based Geostatistics*. Springer, 2006. ISBN 0-387-32907-2  
Geostatistics is concerned with estimation and prediction problems for spatially continuous phenomena, using data obtained at a limited number of spatial locations. The name reflects its origins in mineral exploration, but the methods are now used in a wide range of settings including public health and the physical and environmental sciences. Model-based geostatistics refers to the application of general statistical principles of modeling and inference to geostatistical problems. This volume is the first book-length treatment of model-based geostatistics.
33. Emmanuel Paradis. *Analysis of Phylogenetics and Evolution with R*. Use R. Springer, New York, 2006. ISBN 0-387-32914-5  
This book integrates a wide variety of data analysis methods into a single and flexible interface: the R language, an open source language is available for a wide range of computer systems and has been adopted as a computational environment by many authors of statistical software. Adopting R as a main tool for phylogenetic analyses ease the workflow in biologists' data analyses, ensure greater scientific repeatability, and enhance the exchange of ideas and methodological developments.
34. Sandrine Dudoit and Mark J. van der Laan. *Multiple Testing Procedures and Applications to Genomics*. Springer Series in Statistics. Springer, 2007. ISBN: 978-0-387-49316-9

This book provides a detailed account of the theoretical foundations of proposed multiple testing methods and illustrates their application to a range of testing problems in genomics.

35. Scott M. Lynch. Introduction to Applied Bayesian Statistics and Estimation for Social Scientists. Springer, New York, 2007. ISBN 978-0-387-71264-2

Introduction to Bayesian Statistics and Estimation for Social Scientists covers the complete process of Bayesian statistical analysis in great detail from the development of a model through the process of making statistical inference. The key feature of this book is that it covers models that are most commonly used in social science research—including the linear regression model, generalized linear models, hierarchical models, and multivariate regression models—and it thoroughly develops each real-data example in painstaking detail.

36. Jim Albert. Bayesian Computation with R. Springer, New York, 2007. ISBN 978-0-387-71384-7

Bayesian Computation with R introduces Bayesian modeling by the use of computation using the R language. The early chapters present the basic tenets of Bayesian thinking by use of familiar one and two-parameter inferential problems. Bayesian computational methods such as Laplace's method, rejection sampling, and the SIR algorithm are illustrated in the context of a random effects model. The construction and implementation of Markov Chain Monte Carlo (MCMC) methods is introduced. These simulation-based algorithms are implemented for a variety of Bayesian applications such as normal and binary response regression, hierarchical modeling, order-restricted inference, and robust modeling. Algorithms written in R are used to develop Bayesian tests and assess Bayesian models by use of the posterior predictive distribution. The use of R to interface with WinBUGS, a popular MCMC computing language, is described with several illustrative examples.

37. Jean-Michel Marin and Christian P. Robert. Bayesian Core: A Practical Approach to Computational Bayesian Statistics. Springer, New York, 2007. ISBN 978-0-387-38979-0

This Bayesian modeling book is intended for practitioners and applied statisticians looking for a self-contained entry to computational Bayesian statistics. Focusing on standard statistical models and backed up by discussed real datasets available from the book website, it provides an operational methodology for conducting Bayesian inference, rather than focusing on its theoretical justifications. Special attention is paid to the derivation of prior distributions in each case and specific reference solutions are given for each of the models. Similarly, computational details are worked out to lead the reader towards an effective programming of the methods given in the book. While R programs are provided on the book website and R hints are given in the computational sections of the book, The Bayesian Core requires no knowledge of the R language and it can be read and used with any other programming language.

38. Dianne Cook and Deborah F. Swayne. Interactive and Dynamic Graphics for Data Analysis. Springer, New York, 2007. ISBN 978-0-387-71761-6

This richly illustrated book describes the use of interactive and dynamic graphics as part of multidimensional data analysis. Chapters include clustering, supervised classification, and working with missing values. A variety of plots and interaction methods are used in each analysis, often starting with brushing linked low-dimensional views and working up to manual manipulation of tours of several variables. The role of graphical methods is shown at each step of the analysis, not only in the early exploratory phase, but in the later stages, too, when comparing and evaluating models. All examples are based on freely available software: GGobi for interactive graphics and R for static graphics, modeling, and programming. The printed book is augmented by a wealth of material on the web,

encouraging readers follow the examples themselves. The web site has all the data and code necessary to reproduce the analyses in the book, along with movies demonstrating the examples.

39. David Siegmund and Benjamin Yakir. *The Statistics of Gene Mapping*. Springer, New York, 2007. ISBN 978-0-387-49684-9  
This book details the statistical concepts used in gene mapping, first in the experimental context of crosses of inbred lines and then in outbred populations, primarily humans. It presents elementary principles of probability and statistics, which are implemented by computational tools based on the R programming language to simulate genetic experiments and evaluate statistical analyses. Each chapter contains exercises, both theoretical and computational, some routine and others that are more challenging. The R programming language is developed in the text.
40. Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299-314, 1996
41. Francisco Cribari-Neto and SpyrosG. Zarkos. R: Yet another econometric programming environment. *Journal of Applied Econometrics*, 14:319-329, 1999
42. Robert Gentleman and Ross Ihaka. Lexical scope and statistical computing. *Journal of Computational and Graphical Statistics*, 9:491-508, 2000
43. Paul Murrell and Ross Ihaka. An approach to providing mathematical annotation in plots. *Journal of Computational and Graphical Statistics*, 9:582-599, 2000
44. StephenP. Ellner. Review of R, version 1.1.1. *Bulletin of the Ecological Society of America*, 82(2):127-128, April 2001
45. BrianD. Ripley. The R project in statistical computing. *MSOR Connections*. The newsletter of the LTSN Maths, Stats and OR Network., 1(1):23-25, February 2001
46. Kurt Hornik and Friedrich Leisch, editors. *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*, Technische Universitt Wien, Vienna, Austria, 2001. ISSN 1609-395X
47. PauloJ. Ribeiro, Jr. and PatrickE. Brown. Some words on the R project. *The ISBA Bulletin*, 8(1):12-16, March 2001

## References

Becker, R. A. and Chambers, J. M. (1981) *S: A Language and System for Data Analysis*. AT&T Bell Laboratories, Murray Hill, NJ, USA.