Part I
Preliminaries

Chapter 1
Introducing R

Prerequisites and goals of this chapter

- You may find it useful to read the chapter on installing R in the Appendix first.
- This chapter presents the origins, objectives and specificities of R.

SECTION 1.1

Presentation of the Software

1.1.1 Origins

R is a piece of statistical software created by Ross Ihaka and Robert Gentleman [21]. R is both a programming language and a work environment. Commands are executed using descriptive code. Results are displayed as text and the plots are visualized directly in their own window. R is clone of the statistical software S-plus. S-plus is an object-oriented programming language S developed by AT&T Bell Laboratories in 1988 [3]. S-plus is used to manipulate data, draw plots and perform statistical analyses of data.

1.1.2 Why Use R?

First of all, R is **free** and **open-source**. It works under UNIX (and Linux), Microsoft Windows and Macintosh Mac OS: it is **cross-platform**. It is being developed in the

free software movement by a large and growing community of eager volunteers. 16
Anyone can contribute to and improve R by integrating more functionalities or 17
analysis methods. It is thus a quickly and constantly evolving piece of software. 18

19

R is a very powerful statistical tool. The learning curve in R is steeper than other 20
statistical software on the market such as SPSS, SAS or Minitab. R is not the kind of 21
statistical package, which you can use with a few clicks of the mouse in the menus. 22
In order to use it, you need to understand the statistical method that you are trying to 23
implement, so R is a didactic program. R is also very efficient and easy to implement 24
once you have mastered it. You will be able to create your own tools and you will 25
be able to handle and work on very sophisticated data analyses. 26

Warning

R is harder to comprehend than other software on the market. You need to
spend time learning the syntax and commands.

R is especially powerful for data manipulation, calculations and plots. Its features 27
include: 28

- an integrated and very well-conceived documentation system (in English)   29
- Efficient procedures for data treatment and storage;   30
- a suite of operators for calculations on tables, especially matrices;   31
- a vast and coherent collection of statistical procedures for data analysis;   32
- advanced graphical capabilities;   33
- a simple and efficient programming language, including conditioning, loops, 34
  recursion, and input-output possibilities. 35

Note

For the readers already used to SAS, SPSS or Stata, we advise to read the
books [32, 33] and also to consult the two following Internet websites:

- `http://rforsasandspssusers.com`
- `http://www.statmethods.net`

Note also that it is possible to call R functions directly from Matlab using
the `R.matlab` package and from Excel using the `RExcelInstaller` pack-
age. Reading of [20] might be useful in this context. Finally, a similar tool
for OpenOffice, called ROOo, exists; see the Internet website `http://rcom.
univie.ac.at`.

┌─ SECTION 1.2 ───┐
│   │
│   **R and Statistics**   │
│   │
└──┘ 36

Many classical and modern statistical techniques are implemented in R. The most 37
common methods for statistical analysis, such as 38

- descriptive statistics; 39
- hypothesis testing; 40
- analysis of variance; 41
- linear regression methods (simple and multiple) 42

are directly included at the core of the system. It should be noted that most 43
advanced statistical methods are also available through external packages. These 44
are easy to install, directly from a menu. They are all grouped and can be browsed 45
on the website of the *comprehensive R archive network* (CRAN) (`http://cran.` 46
`r-project.org`). This website also includes, for some large domains of interest, 47
a commented list of packages associated with a theme (called Task View). This 48
facilitates the search for a package on a specific statistical method. Furthermore, 49
detailed documentation for each package is available on the CRAN. 50
51
It should also be noted that recent statistical methods are added on a regular basis 52
by the statistics community itself. 53
54

See also

Section A.2, p. 532, gives details on the procedure to install a new package.

┌─ SECTION 1.3 ───┐
│   │
│   **R and Plots**   │
│   │
└──┘ 55

One of the main strengths of R is its capacity (much greater than that of other 56
software on the market) to combine a programming language with the ability to 57
draw high-quality plots. Usual plots are easily drawn using predefined functions. 58
These functions also include many parameters, for example to add titles, captions 59
and colours. But it is also possible to create more sophisticated plots to represent 60
complex data such as contour lines, volumes with a 3D effect, density curves, and 61
many other things. It is also possible to add mathematical formulae. You can arrange 62
or overlay several plots in the same window and use many colour palettes. 63
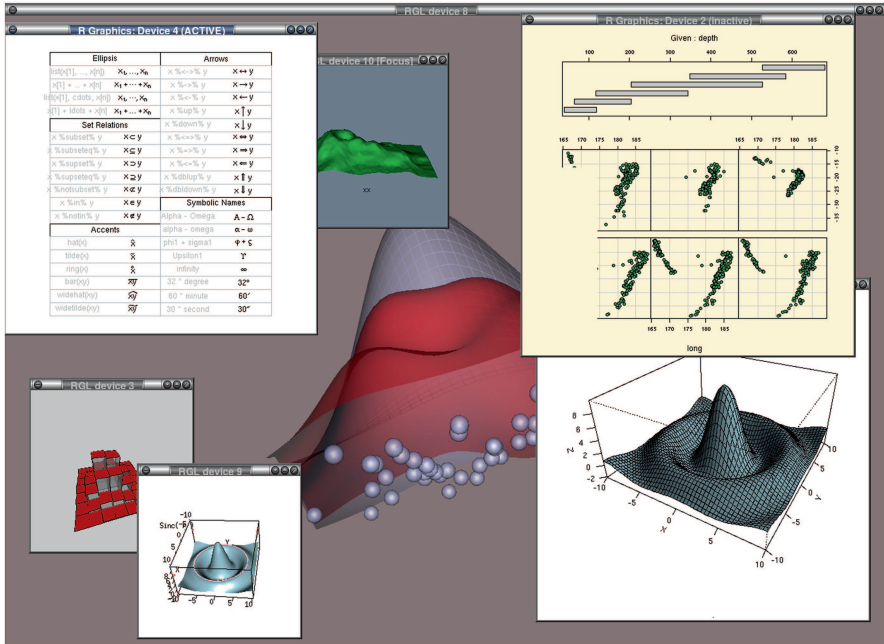
Fig. 1.1: A few of the graphical possibilities offered by **R**

You can get a demonstration of the graphical possibilities in **R** by typing in the 64
following instructions: 65

```
demo(image)
example(contour)
demo(graphics)
demo(persp)
demo(plotmath)
demo(Hershey)
require("lattice")  # Load the package, which you must have
                    # previously installed by using the menu
                    # Packages/Install packages.
demo(lattice)
example(wireframe)
require("rgl")      # Same remark as above.
demo(rgl)           # You can interact by using your mouse.
example(persp3d)
```

Figure 1.1 above shows a few of these plots.   66

67

The R Graphical User Interface

68

The R graphical user interface (GUI) (i.e. its set of menus) is very limited, and com- 69
pletely nonexistent on some operating systems, when compared to other standard 70
software. This minimality can set back some new users. However, this drawback is 71
limited since: 72

- it has the didactic advantage that it incites users to know well the statistical pro- 73
  cedures they wish to use; 74
- there are additional tools which extend the GUI. 75

In the next section, we present the package Rcmdr, which can be installed from 76
the menu Packages and which allows standard graphical and statistical analyses 77
with a more user-friendly interface, which includes drop-down menus. Furthermore, 78
the R instructions for the analysis chosen from the RCommander menus are displayed 79
in dedicated panel. This can be useful if you do not know (or remember) the R 80
instruction for a specific task. 81

> **Tip**
>
> Note that after you have learnt R thoroughly, you will be able to develop
> yourself tools similar to Rcmdr, made for a final users who do not desire to
> learn R but only to use, in the most user-friendly way, a procedure created by
> you. To this end, you can use the package tcltk.

> **Warning**
>
> Note that by using RCommander, we are distancing ourselves from what
> makes the strength and flexibility of R. We therefore advise against using such
> a tool if you wish to become an advanced user.

SECTION 1.5

First Steps in R

82

1.5.1 Using RCommander

83

In this section, we offer a brief introduction to the package Rcmdr. We then present 84
some functionalities given by this interface for statistical manipulations. We con- 85
clude by explaining how to add functionalities to the RCommander interface. 86

1.5.1.1 Launching `RCommander`   87

Follow these steps to start `RCommander`.   88

▶ Double-click on the **R** icon on your Desktop.   89
▶ In the console, type `install.packages("Rcmdr")`. Choose a nearby mirror.   90
▶ In the console, type `require("Rcmdr")`. Answer `Yes` to all the questions you   91
  may be asked. The `RCommander` graphical interface then opens. Another option   92
  is to click on the menu `Packages`, then `Load package...`, then `Rcmdr`.   93
▶ In the `Messages` panel, you should see `WARNING: the Windows version of`   94
  `R Commander works better under RGui with the single document`   95
  `interface (SDI)`.   96
▶ To remedy this issue, close `RCommander`.   97
▶ In RGui, go to `Edit`, then `Preferences`. Check SDI then click on `Save...` and   98
  on `Save`. You can take this opportunity to customize **R**.   99
▶ Close **R** and save an image of the session.   100
▶ Restart **R**, then `RCommander` by typing `require("Rcmdr")` in the **R** console. 101

> **See also**
>
> We refer the reader to Sect. A.2 which details how to install the package
> `Rcmdr`.

> **Mac**
>
> Macintosh users may find useful the instructions at `http://socserv.`
> `mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html`, after in-
> stalling package `tcltk` which is available on the CRAN.

The graphical interface of `RCommander` includes four parts as shown on Fig. 1.2:   102

(a) Drop-down menus to perform specific tasks   103
(b) A `Script window` which presents the code executed thanks to click on a drop-   104
    down menu   105
(c) An `Output window` which gives the output of the executed code   106
(d) A `Messages` window giving a message on the last task   107
                                                                                       108

1.5.1.2 Handling Data with `RCommander`   109

To perform statistical analyses, you need data.   110
                                                                                       111

● **Entering data by band**   112
                                                                                       113

Follow these steps to enter data by hand.   114

Fig. 1.2: The RCommander graphical interface

▶ In the menu Data, choose New data set....   115
▶ In the window New data table, choose a name for your data set, for example   116
  Data1.   117
▶ A data editor appears. Click on var1 and replace it with Name. Enter a few names 118
  for this variable: Peter, Jack, Ben (see Fig. 1.3).   119
▶ Create a variable Height of type numeric with the following values: 182, 184, 120
  190.   121
▶ Click on the cross (X) at the top-right corner of the active window to close the 122
  data editor.   123
▶ You can visualize your data set by clicking on View.   124

We can now calculate some basic statistics.   125

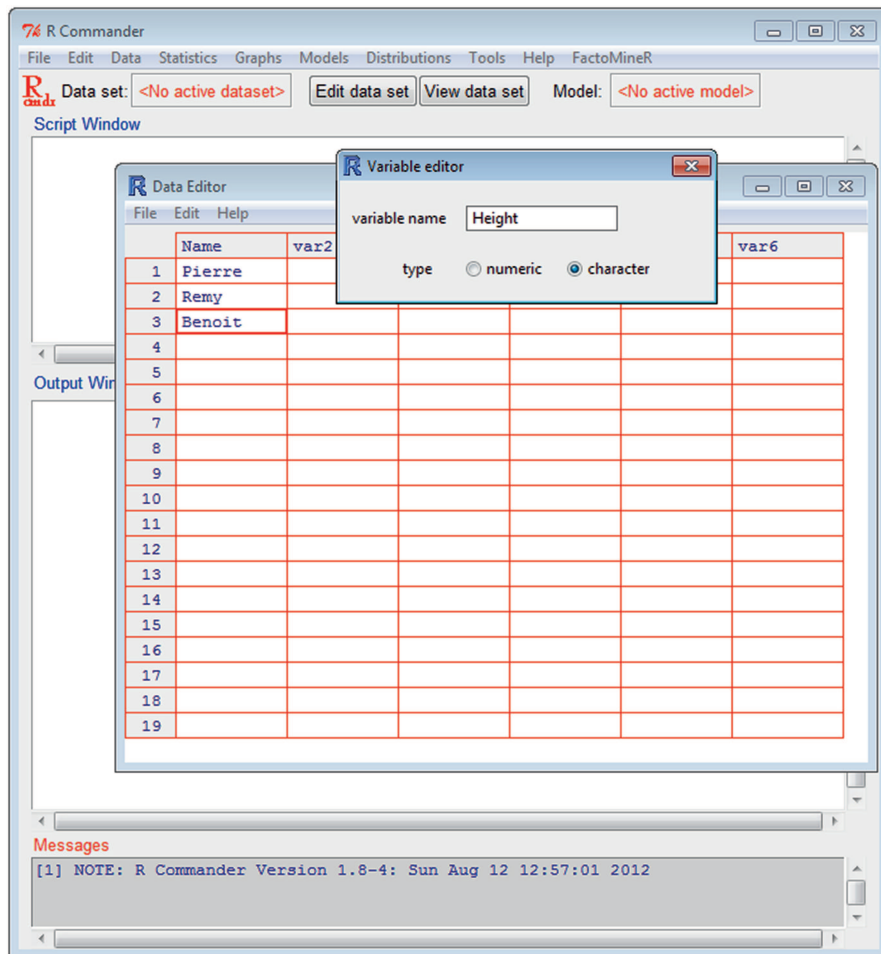Fig. 1.3: Entering data with the `RCommander` graphical interface

126

● **Basic statistics**   127

128

Follow these steps to get some basic statistics on your data set: 129

▶ In the menu `Statistics`, choose `Summary`, then `Descriptive stati-`   130
   `stics ....`   131
▶ A window called `General statistics` opens up; the only `numeric` variable 132
   in our data set is the variable `Height`.   133
▶ Choose the statistics `Mean`, `Standard deviation` and `Quantiles` and click on 134
   `OK`.   135
▶ The result is displayed in the `Output window`. Note that you can check the **R** 136
   instruction which was used for this task in the `Script window` (see Fig. 1.4).   137
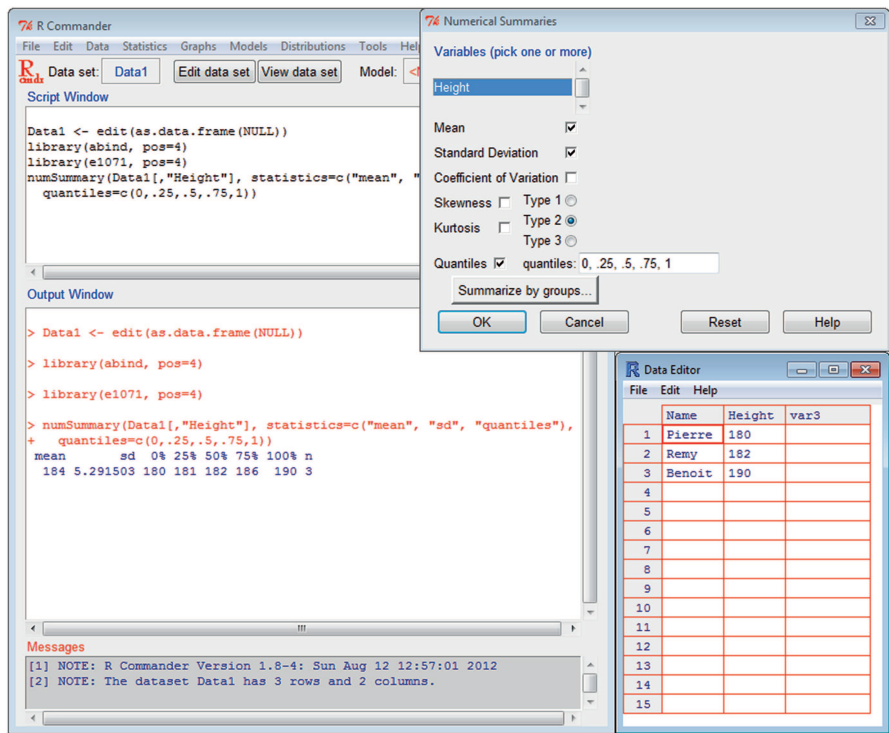
Fig. 1.4: Basic statistics with `RCommander`

Note that it is also possible to type an instruction directly in the `Script window` 138
without using a menu. Here is an example. 139

► Type in the `Script window`: 140

```
numSummary(Data1[,"Height"],statistics=c("mean","sd"))
```

► Click on that line so that the cursor is displayed there, then click on `Submit`. 141
► You have just computed the mean and standard deviation of variable `Height` 142
which contains 3 observations. The result appears in the `Output window`: 143

```
mean        sd % n
 184 5.291503 0 3
```

144

● **Manipulating the data set** 145

146

In our toy example, suppose that we also have the weight and wish to compute 147
the body mass index: $BMI = Weight/Height^2$ (height in metres). 148
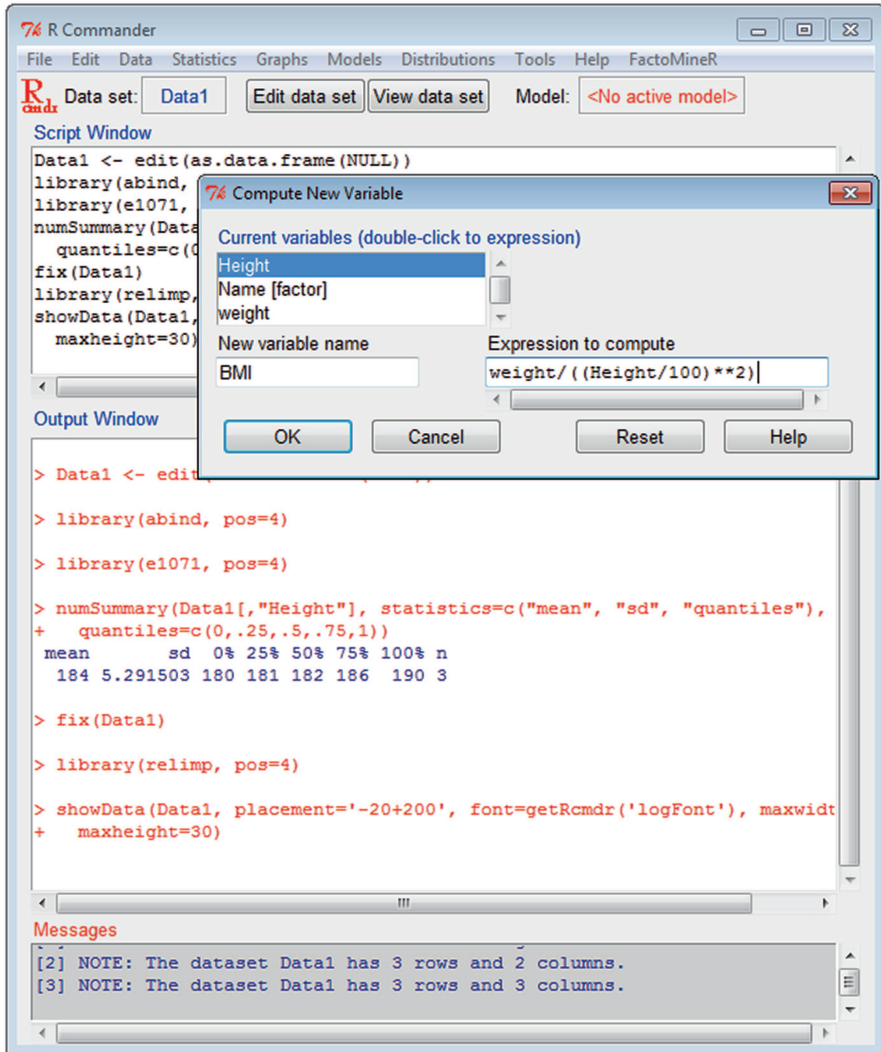
Fig. 1.5: Manipulating a data set with `RCommander`

▶ Click on `Edit` (below the `RCommander` menus).   149
▶ The data editor opens up and you can add the numeric variable `Weight`, with the   150
  following values: 70, 72 and 75. Now close the data editor.   151
▶ In the `Data` menu, choose `Manage variables in the active data set`,   152
  then `Calculate a new variable....` A window opens.   153
▶ For `Name of new variable`, type `BMI` and for `Expression to calculate`: 154
  `Weight/((Height/100)**2)` (see Fig. 1.5). Click on `OK` to complete the calcu- 155
  lation.   156
▶ Click on `View` to see the result for your data set.   157

You are starting to feel tired and need a coffee break! But before you take one, follow these steps to save your data set.

▶ In the `Data` menu, choose `Active dataset`, then `Save active dataset...`.
▶ A window opens. You can choose a location to save your data set. We shall call it `BMI` and by default it has the `.RData` extension.
▶ Close `RCommander` and answer `OK` to the question `Do you wish to quit?`, `No` to `Save script file?` and `No` to `Save output file?`.
▶ You can now close R and answer `No` to the question `Save session image?`.

After a well-deserved break, you wish to add new data to your file `BMI.RData`.

▶ Open an R session. Type `library("Rcmdr")`.
▶ In the `Data` menu, choose `Load data set...`.
▶ A window opens. Navigate to and open the file `BMI.RData`.
▶ Click on `View` to display your data set.
▶ Add the information for a new person (`"Julia"`, `Height=150`, `Weight=52`) by clicking on `Edit`.
▶ After closing the editor, you can check the changes by clicking on `View`. You then see the value `NA` (*not available*) for Julia's BMI.
▶ To get Julia's BMI, you need to go through the steps of section *manipulating the data set* again. We shall see later on how to create a function which calculates the BMI in a more user-friendly fashion.

You now wish to send your data set to a colleague who does not use R yet.

▶ In the `Data` menu, choose `Active dataset`, then `Export active data set ...`.
▶ A first window opens. Uncheck the box `Write names of individuals (rows)` since we have not defined these. Choose `Spaces` for the field separator.
▶ Click on `OK`. A second window opens. You can choose a place to save your data set. We shall call it `BMI` and it has the default extension `.txt`.
▶ You can now send your data set `BMI.txt` to your colleague and use this opportunity to mention the wonderfulness of R, which has a rather user-friendly interface for data manipulation.

1.5.1.3 A Few Statistical Tasks with `RCommander`

In this section, we present a brief overview of how to use `RCommander` for statistical tasks. We start with a mean comparison test and a chi-square test of independence. We then show how to use `RCommander` to visualize the standard distributions (binomial, poisson, normal, gamma, etc.). We conclude with a linear model fit.

• **Mean comparison test**   195

196

We propose to use data already available in **R**. Follow these steps to load a data   197
set: 198

▶ In the `Data` menu, choose `Data in packages`, then `Read data from an`   199
  `attached package....`   200
▶ A window opens. Double-click on `datasets` in the `Package` section, then on   201
  `sleep` in the right column.   202
▶ `sleep` appears in the box `Enter a dataset name` (see Fig. 1.6).   203
▶ You can now click on `Help on the selected dataset` to have some infor-   204
  mation about it.   205
▶ Click on `OK` to close the previous window, then visualize the data set by clicking 206
  on `View`.   207

These data are used to compare the effect on sleep of a soporific drug, compared to   208
a control group. We shall first visualize the distribution of sleep gain in both groups, 209
then do a mean comparison test to see whether there is any statistical significant   210
difference between the drug and the control.   211

▶ In the `Graphs` menu, choose `Box plot....`   212
▶ A window opens. Click on `Plot by group...`, then on the variable `group`,   213
  then on `OK` twice.   214
▶ You can now see two box plots representing the sleep time gain in both groups.   215
▶ You can save this plot by clicking on `File`, then `Save as`. Several formats are   216
  possible.   217

You can also enhance this plot, for example, by adding colours. In the script window, 218
type   219

```
boxplot(extra~group,ylab="extra",xlab="group",data=sleep,
        col=c("red","blue"))
```

then click on `Submit`.   220

> **See also**
>
> Chapter 7 is dedicated to plots in **R**.

We now perform a mean comparison test.   221

▶ In the `Statistics` menu, choose `Means`, then `independent t-test....`   222
▶ Click on `group` in section `Groups (one)`. You now see specified the difference   223
  `1-2` (group 1 *vs.* group 2).   224
▶ Click on `OK` to see the result in the `Output window` (see also Fig. 1.6).   225

The *p*-value of this test (greater than 5 %) does not allow us to conclude that   226
there is a significant difference between the sleep gains given by the drug and the   227
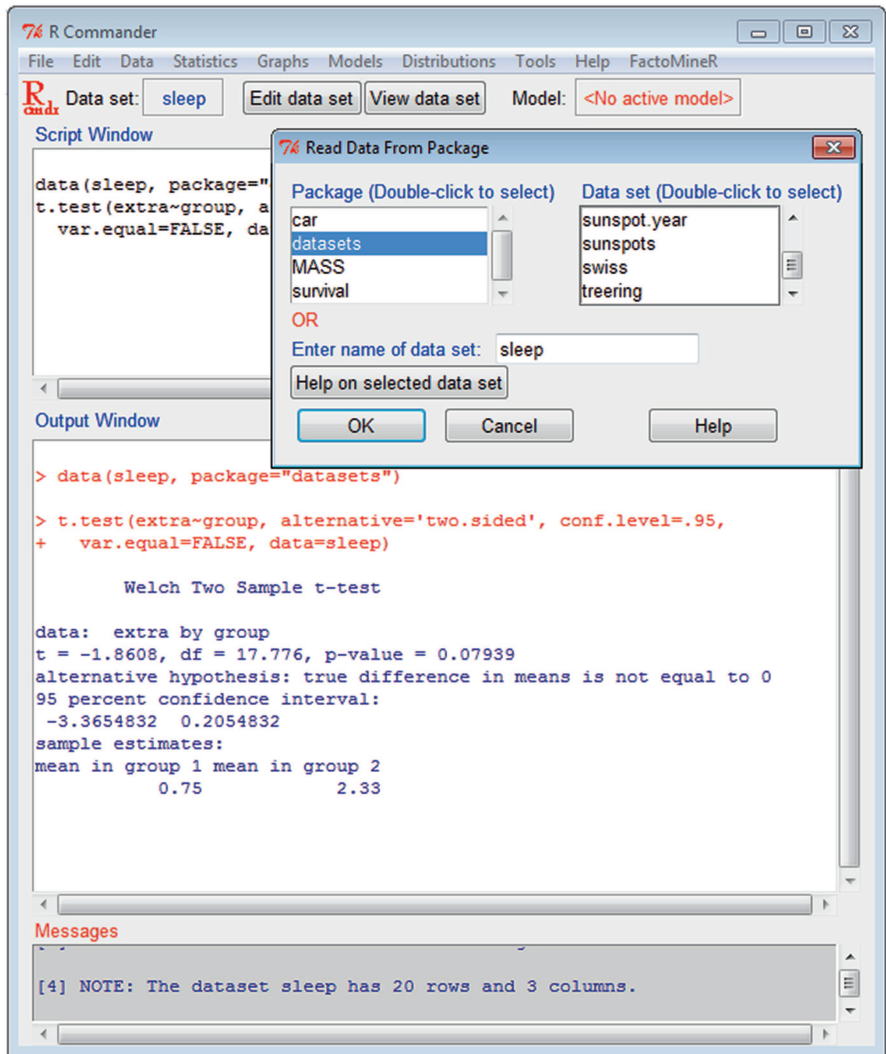control.   228

Fig. 1.6: Mean comparison test with RCommander

● Test on a double entry table

In a therapeutic test, the underlying question is whether a treatment on HIV-positive mothers has an effect on the HIV status of the child. If it does not, then the HIV status of the child is independent of the treatment taken by the mother. In this test, out of 391 children, 100 are HIV negative, 193 have mothers under treatment and 41 are HIV positive and have mothers under treatment. To know whether the treatment has an effect, follow these steps:
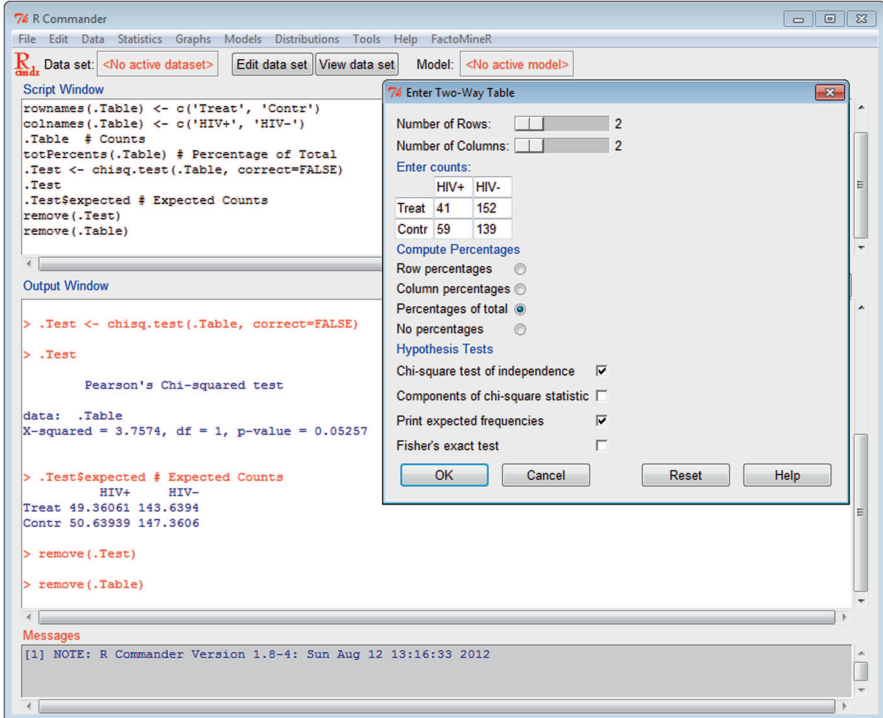
Fig. 1.7: Independence test with `RCommander`

► In the `Statistics` menu, choose `Contingency tables`, then `Fill and`   237
  `analyse a double entry table....`   238
► A window opens. Fill the table as indicated in Fig. 1.7. Choose   239
  `Total percentages` and `Print expected frequencies`.   240
► Click on `OK` to see the result in the `Output window`.   241

At the 5 % risk level, we cannot conclude that the treatment has an effect on the   242
child's HIV status.   243

                                                                              244
● **Exploring distributions**   245

                                                                              246
`RCommander` can be used to visualize standard distributions.   247

► In the `Distributions` menu, choose `Continuous distributions`, the   248
  `Normal distribution`, then `Plot of normal distribution....`   249
► A window opens. Specify a mean of 4 and a standard deviation of 2. Click on `OK`. 250
► The curve of the density of a normal distribution centred at 4 and with standard 251
  deviation 2 appears in a graphical window.   252

You can follow the same steps for other probability distributions.   253

                                                                              254

• Fitting a linear model   255

256

`RCommander` can be used to easily fit standard regression models. We illustrate   257
this with the linear model. We shall first download a data set from an Internet   258
address (URL). It contains the measures, for 80 patients with a disabling illness,   259
of the variables `GENDER` (1 = Male, 2 = Female), `WEIGHT` (in kg), `HEIGHT` (in   260
cm), `PAIN` (ordinal variable: a=least pain), `DISTANCE` (number of metres walked),   261
`MOBILITY` (self-evaluation of mobility; 1=most mobile) and `STAIRS` (number of   262
steps climbed).   263

▶ In the `Data` menu, choose `Import data`, then `from a text file, the`   264
    `clipboard or a URL....`   265
▶ A window opens. Call the data table `Illness`. Check the box `Internet link`   266
    `(URL)` in `Data file` and the box `Tabulations` for `Field separator`; click   267
    on `OK`.   268
▶ In the field `Internet link (URL)`, type   269
    `http://biostatisticien.eu/springeR/illness.txt`.   270
▶ Click on `OK` and you should see the following in the `Messages` window: `The`   271
    `illness data set contains 80 rows and 8 columns.`   272

We shall fit a multiple regression model. Follow these steps.   273

▶ In the `Statistics` menu, choose `Model fitting`, then `Linear regression`   274
    `....`   275
▶ Choose for example `Model.1` as your model name in the field `Enter a name`   276
    `for the model.`   277
▶ Choose variable `DISTANCE` as the response variable, and variables `WEIGHT` and   278
    `HEIGHT` as the explanatory variables (keep the `CTRL` key pressed).   279
▶ Click on `OK`. The result of your linear model adjustment appears in the `Output`   280
    window. This result corresponds to the instructions   281

```
Model.1 <- lm(DISTANCE~WEIGHT+HEIGHT,data=Illness)
summary(Model.1)
```

which are shown in the `Script window`.   282

See also

Chapter 14 presents the linear model in further detail.

We now visualize the least squares plane corresponding to the fitted model.   283

▶ In the `Plot` menu, choose `3D plot`, then `3D scatterplot....`   284
▶ Choose variable `DISTANCE` as the response variable and the variables `WEIGHT`   285
    and `HEIGHT` as explanatory variables (use the `CTRL` key).   286
▶ Choose `Ordinary least squares` as the surface to fit. Click on `OK`.   287
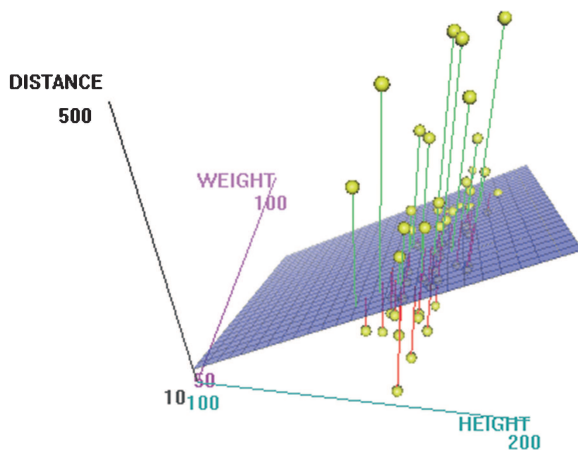
Fig. 1.8: Least squares plane

You can now see the 3D scatterplot (shown in Fig. 1.8) and the least squares   288
plane. You can move the image with your mouse.   289

1.5.1.4 Adding Functionalities to the RCommander Interface   290

Some packages available on the official **R** website can also be integrated to 291
the RCommander menus. They are easy to identify: their names start with   292
RcmdrPlugin. We now illustrate how to use such a package.   293

> **See also**
>
> You can read the article [17] which explains how to build a package for
> RCommander integration.

● **The TeachingDemos package**   294
                                                                                  295
The RcmdrPlugin.TeachingDemos package can be used to illustrate some sta-   296
tistical concepts.   297

► Type install.packages("RcmdrPlugin.TeachingDemos") in the Script   298
  window. Click on Submit and choose a nearby mirror. Once the installation is   299
  complete, close and reopen RCommander using the instruction Commander().   300
► In the Tools menu, choose Load Rcmdr plug-ins..., click on OK and answer   301
  Yes to the question Restart now?.   302
► There is a new menu called Demos. In this menu, you can choose for example the 303
  submenu Simple Correlation and explore the notion of correlation.   304

This plug-in also adds submenus to pre-existing menus. For example, in the   305
`Distributions` menu, you can now choose `Visualize distributions`, then   306
`t distributions`. By checking `Show Normal Distribution`, and by playing   307
with the `d.f.` (*degree of freedom*) cursor, you can visualize the closeness of the   308
Student distribution and the normal distribution.   309

   310

● **The `sos` package**   311

   312

The `RcmdrPlugin.sos` package can be used to ease the search for help on a   313
given concept or function. Follow the same steps as before to install this plug-in. A   314
new submenu called `Search R Help ... (sos)` appears in the `Help` menu. Ex-   315
plore this new `Rcommander` functionality, for example, by typing `linear model`.   316

See also

Chapter 6 describes how to search for information about R.

1.5.2 *Using R with the Console*   317

In the previous subsection, we saw how to use R through menus. In fact, this way   318
of proceeding is far from optimal, since it imposes many limitations on the possi-   319
bilities offered by R. Many analyses, either deeper or more recent and innovative,   320
are not available in the `RCommander` menus. It is thus very useful to escape from   321
the "button clicking" approach and master the R programming language. You will   322
then be able to perform simulations and to code repetitive tasks. We have already   323
encountered a few R instructions when using `RCommander`, which is itself a tool   324
written in the R language. We now propose a brief introduction to a few elements   325
of the R syntax, first through an analysis of complex data arising from a functional   326
magnetic resonance imaging (MRI) experiment, then by letting the reader type a   327
few R commands and think about the output.   328

1.5.2.1 The Strength of R Shown on an Example   329

Some neuroscientists work on finding which part of the brain deals with visual   330
information on colour. To this end, a visual stimulus, consisting in an alternance   331
of coloured and non-coloured moving patterns, is shown to a subject. During this   332
time, volumic images of the subject's brain are acquired at time $t = 1, \ldots, T$ with   333
an MRI scanner. Each 3D image is in fact a large (Rubik's!) cube made of many   334
voxels, the 3D equivalents of 2D pixels. At time $t = 1, \ldots, T$, each voxel con-   335
tains an electromagnetic measurement value $x(t)$. We can thus consider that in   336
each voxel, we have observed a time series $\{x(t); t = 1, \ldots, T\}$ representing

electromagnetic variations. The acquired data (given in file Mond4D.nii, produced   337
during a Mondrian experiment performed by M. Dojat and J. Huppé) thus consist in   338
a 4-dimensional array, the concatenation of several volumic brain images measured   339
through time.   340

                                                                                   341

We used R to find, in each brain slice, which voxel had temporal variations   342
most correlated with the stimulus signal. The code below can be downloaded from   343
http://biostatisticien.eu/springeR/brain-code.R and opened, thanks   344
to the submenu Open script... of the File menu in R. The key combination   345
CTRL+R then executes one by one the instructions of this script. You can try to   346
execute these instructions to visualize the results. This will help you familiarize   347
yourself with some of the possibilities offered by R.   348

                                                                                   349

We first download the data files we need (the files Mondanat.img and   350
Mondanat.hdr contain an anatomical image of the subject's brain).   351

```
> getfile <- function(myfile)
+  download.file(paste("http://biostatisticien.eu/springeR/",
+  myfile,sep=""),paste(getwd(),"/",myfile,sep=""),mode="wb")
> getfile("Mond4D.nii")
> getfile("Mondanat.hdr")
> getfile("Mondanat.img")
```

We then install the package to read the data.   352

```
> install.packages("AnalyzeFMRI") # Choose a mirror.

> # File names.
> file.func <- paste(getwd(),"/","Mond4D.nii",sep="")
> file.anat <- paste(getwd(),"/","Mondanat.img",sep="")

> # Brain slice number.
> slice <- 10
```

The next instructions read the data.   353

```
> anat.slice <- f.read.nifti.slice(file.anat,slice,1)
> class(anat.slice)
[1] "matrix"
> dim(anat.slice)
[1] 128 128
> func.slice <- f.read.nifti.slice.at.all.timepoints(file.func,
   slice)
> class(func.slice)
[1] "array"
> dim(func.slice)
[1] 128 128 125
```

We now create the coding of the visual stimulus signal (1=colour, 0=no colour).   354

```
> stimulus <- c(rep(c(1,1,1,1,1,0,0,0,0,0,0,0,0,0,0),8),1,1,1,1,1)
> plot(stimulus,type="l")
```
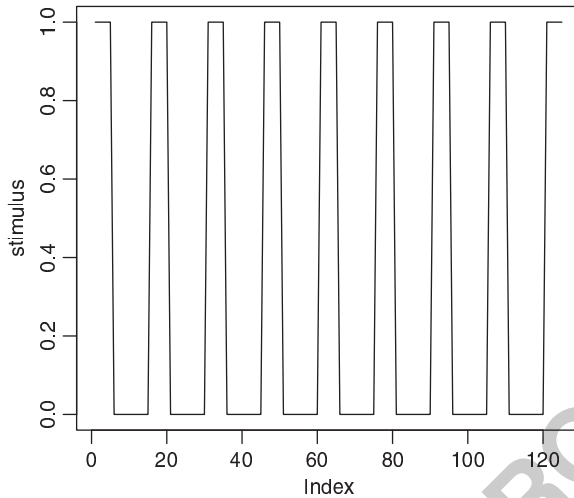
We compute correlations between the observed time series in each voxel and the stimulus series.

```
> corMat <- matrix(NA,nrow=128,ncol=128)
> for (i in 1:128) {
+    for (j in 1:128) {
+      corMat[i,j] <- cor(func.slice[i,j,],stimulus)
+    }
+ }
```

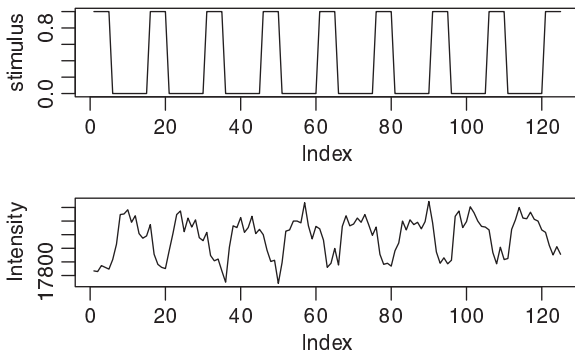We can now compute the coordinates of the voxel most strongly correlated with the stimulus

```
> which(abs(corMat)==max(abs(corMat),na.rm=TRUE),arr.ind=TRUE)
     row col
[1,]  67 117
```

and the correlation value of this voxel

```
> corMat[67,117]
[1] -0.6675017
```

We can then plot the time series observed in this voxel.

```
> par(mfrow=c(2,1))
> plot(stimulus,type="l")
> plot(func.slice[67,117,],type="l",ylab="Intensity")
```

362

We are now able to identify on the anatomical image of the brain the most active   363
voxel for the visual stimulus.   364

```
> image(as.matrix(rev(as.data.frame(t(anat.slice))))),
     col=gray((0:32)/32))
> points(117/128,67/128,col="red",cex=2,pch=19)
```
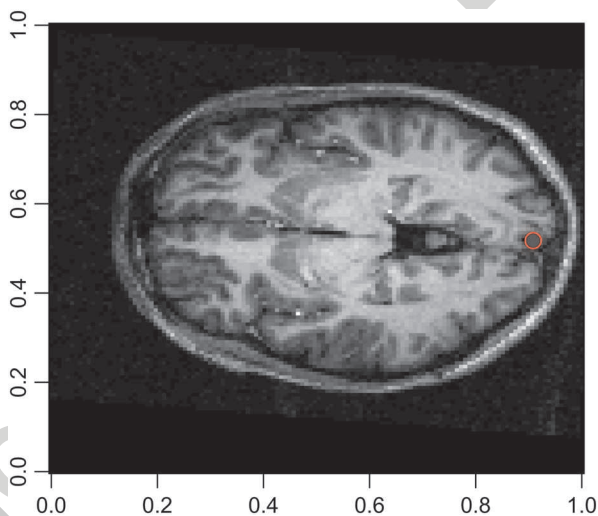

365

Note that you can also visualize these data in 3D. The following instructions,   366
taken from the help file for the function contour3d() from package misc3d, give   367
an interactive 3D view of the brain.   368

```
> install.packages("misc3d")

> require("misc3d")
> a <- f.read.analyze.volume(system.file("example.img",
+                package="AnalyzeFMRI"))
> a <- a[,,,1]
> contour3d(a,1:64,1:64,1.5*(1:21),lev=c(3000, 8000, 10000),
+          alpha=c(0.2,0.5,1),color=c("white","red","green"))
```
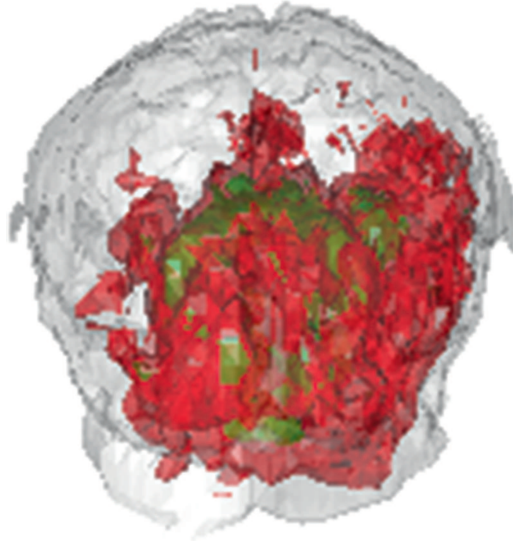
You can try to move the image with your mouse.

1.5.2.2 A Brief Introduction of R Syntax Through Some Instructions to Type

● **Basic operations**

We advise the reader to play with these commands and try to understand how
they work.

```
> 1*2*3*4
[1] 24
> factorial(4)
[1] 24
> cos(pi)
[1] -1
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> exp(x)
 [1]     2.718282      7.389056     20.085537     54.598150
 [5]   148.413159    403.428793   1096.633158   2980.957987
 [9]  8103.083928 22026.465795
> x^2
 [1]   1    4    9   16   25   36   49   64   81 100
> chain <- "R is great!"
> chain
[1] "R is great!"
> nchar(chain)
[1] 11
> ?nchar
```

```
> M <- matrix(x,ncol=5,nrow=2)
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> M[2,3]
[1] 6
> L <- list(matrix=M,vector=x,chain=chain)
> L[[3]]
[1] "R is great!"
> while(TRUE) {
+   toguess <- sample(1:2,1)
+   {cat("Guess a number among 1, 2, 3: ") ; value <- readline()}
+    if (value == toguess) {print("Well done!") ; break()}
+    else print("Try again.")
+ }
> ls()
[1] "chain" "L"       "M"       "x"
> rm(chain)
```

The following commands perform matrix operations: 376

```
> A <- matrix(runif(9),nrow=3)
> 1/A
         [,1]      [,2]      [,3]
[1,] 2.270797 1.546875 1.422103
[2,] 1.268152 1.957924 1.057803
[3,] 1.642736 5.273120 2.174020
> A * (1/A)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
> B <- matrix(1:12,nrow=3)
> A * B
Error in A * B : non-conformable arrays
> A %*% B
          [,1]       [,2]       [,3]       [,4]
[1,] 3.842855  9.212923 14.582990 19.95306
[2,] 4.646105 11.380053 18.114001 24.84795
[3,] 2.367954  6.143031  9.918107 13.69318
> (invA <- solve(A))
          [,1]       [,2]       [,3]
[1,]  1.145642 -3.376148  5.187347
[2,]  4.379786 -4.641906  2.844607
[3,] -3.321872  6.381822 -5.863772
> A %*% invA
              [,1]          [,2] [,3]
[1,]  1.000000e+00 0.000000e+00    0
[2,]  0.000000e+00 1.000000e+00    0
[3,] -2.220446e-16 4.440892e-16    1
```

```
> det(A)
[1] 0.04857799
> eigen(A)
$values
[1]  1.6960690+0.000000i -0.1424863+0.091319i
[3] -0.1424863-0.091319i
$vectors
               [,1]                    [,2]                    [,3]
[1,] 0.5859852+0i   0.6140784-0.1816841i   0.6140784+0.1816841i
[2,] 0.7064296+0i   0.2234155+0.2505528i   0.2234155-0.2505528i
[3,] 0.3969616+0i  -0.6908020+0.0000000i  -0.6908020+0.0000000i
```

● **Statistics**

Here are a few statistical calculations.

```
> weight <- c(70,75,74)
> mean(weight)
[1] 73
> height <- c(182,190,184)
> mat <- cbind(weight,height)
> mat
     weight height
[1,]     70    182
[2,]     75    190
[3,]     74    184
> apply(mat,MARGIN=2,FUN=mean)
  weight   height
 73.0000 185.3333
> ?apply
> colMeans(mat)
  weight   height
 73.0000 185.3333
> names <- c("Peter","Ben","John")
> data <- data.frame(Names=names,height,weight)
> summary(data)
   Names       height          weight
 Ben  :1   Min.   :182.0   Min.   :70.0
 John :1   1st Qu.:183.0   1st Qu.:72.0
 Peter:1   Median :184.0   Median :74.0
           Mean   :185.3   Mean   :73.0
           3rd Qu.:187.0   3rd Qu.:74.5
           Max.   :190.0   Max.   :75.0
```
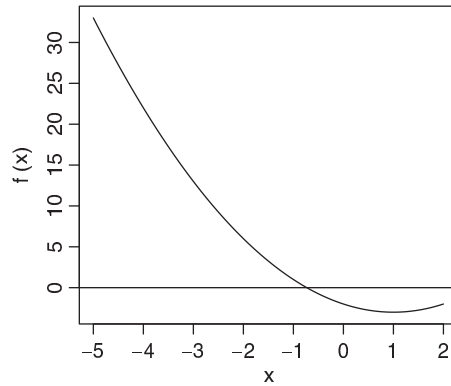
● **Some plots**

```
> f <- function(x) x^2-2*x-2
> curve(f,xlim=c(-5,2));abline(h=0)
> locator(1) # Click on the intersection of the two curves.
```
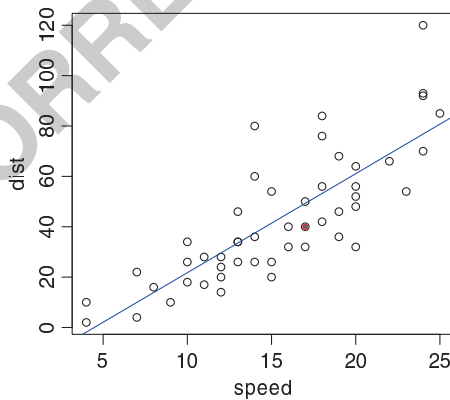
377
378
379

380
381

```
> uniroot(f,c(-5,2))
$root
[1] -0.7320503
$f.root
[1] -1.874450e-06
$iter
[1] 8
$estim.prec
[1] 6.103516e-05

> plot(cars)
> abline(lm(dist~speed,data=cars),col="blue")
> points(cars[30,],col="red",pch=20)
```
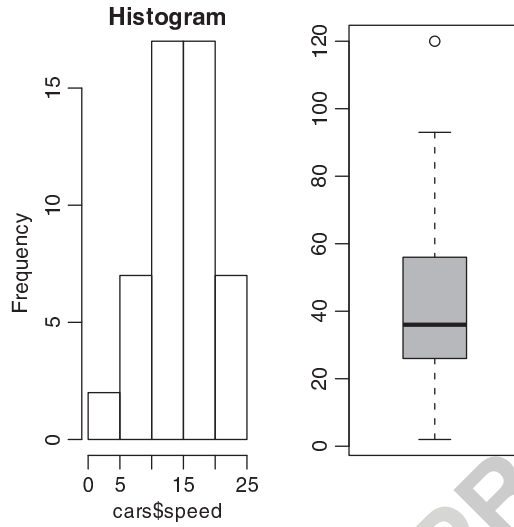
```
> par(mfrow=c(1,2))
> hist(cars$speed,main="Histogram")
> boxplot(cars$dist,col="orange")
```

Histogram

This link points to a reference card of the most useful R functions `http://cran.r-project.org/doc/contrib/Short-refcard.pdf`