APPENDIX: AN INTRODUCTION TO S AND S-Plus

This appendix gives the script for a possible introductory session intended for first time users of S-Plus. Imagine that you are trying to learn enough of S to use it in a Data Analysis course based on this book, and let us imagine that the instructor of the course decided to help the students by devoting one of her first lectures to a first introduction to S-Plus. This appendix should be understood as a summary of all the commands and examples used in this hypothetical introductory lecture.

But before we start, a little bit of terminology might be helpful to explain why we often use S and S-Plus interchangeably. The reader needs to keep in mind that S is the programming environment/language developed at AT&T, and S-Plus is the commercial package implementing this language. Consult the online manuals, and especially the *Guide to Statistics* for more information on S and S-Plus.

STARTING S-PLUS (UNDER Windows AND UNDER UNIX)

The instructions given below are restricted to versions 6.0 and higher of S-Plus.

Under Windows

After starting the program, you will be faced with a dialog box asking you to choose the S-Plus Chapter in which you want to work. If you are using the program for the first time, there is no existing chapter, and you will need to browse your hard disk and choose a directory (folder) – or possibly create one – in which you will work on this project. It is recommended that you choose a special chapter which you will use exclusively for the purpose of running the examples and doing the homework problems of this book. The notion of S-Plus chapter replaces the the notion of project or workspace of the previous Windows versions of S-Plus (for example in S-Plus 2000).

The first thing need to do is to go to the menu item **File** ▷ **Chapters** and choose **File** ▷ **Chapters** ▷ **Attach/Create New Chapter** if you want to create a new chapter

(which you will want to do the first time you run S-Plus), or if you want to attach an existing chapter to be able to access its functions and data. You should choose **File ▷ Chapters ▷ New Working Chapter ...** if you want to switch to an existing chapter different from the default chapter started with S-Plus.

Under Unix

Let us now imagine that the introductory session was intended for Unix or Linux users (we will use the notation $ to denote the Unix prompt), and let us imagine that the name of the class is ORF405.

The first thing we want to do is to create a subdirectory ORF405 in your home directory. The plan is to store all the files relative to the course in this directory, and in particular, to do our S-Plus based homework in this directory. Next we move to this directory, and run the command CHAPTER

```
$ mkdir ORF405
$ cd ORF405
$ S-Plus CHAPTER
Creating data directory for chapter .
S-PLUS chapter ORF405 initialized.
```

The purpose of this command is to customize the directory for future use. Among other things, it creates a subdirectory .Data, and all the objects created by S-Plus when you run it in the directory ORF405 will be saved (automatically) in this subdirectory .Data and all the objects present in this directory will be accessible each time you run S-Plus from ORF405. You can now start the program S-Plus.

```
$ S-Plus
S-PLUS : Copyright (c) 1988, 2002 Insightful Corp.
S : Copyright Lucent Technologies, Inc.
Version 6.1.2 Release 2 for Linux 2.2.12 : 2002
Working data will be in .Data
>
```

The four lines of text reproduced above are a sign that S-Plus started without problem. The next thing we do is to open a graphic window in which we shall display the graphic objects produced by S-Plus.

```
> trellis.device(motif)
```

At this stage a motif window should appear on the desktop. If this is not the case, especially if you get an error message stating that a display cannot be open, one reason could be that you are running S-Plus remotely, and you did not set up the environment correctly. In such a case, you need to type the following commands needed for the X window system to behave appropriately:

```
$ setenv DISPLAY ''network name of the terminal''
```

on the remote server on which you are running S-Plus to make sure that the server knows where to send the X graphic objects and

```
xhost +
```

to make sure that your terminal will accept the X commands to display the graphics.

Important Remark

Most every work with S can be described independently of the platform if one limits ourselves to the list of S-commands used to perform whatever tasks one is interested in. We will adhere to this practice. This is not much of an issue for Unix or Linux users who are traditionally used to doing most of the work through typing. But it is clear that doing so, we are missing on most of the *goodies* of the Graphic User Interface, also called GUI (and pronounced "gooohee"). In any case, I remain convinced that the users preferring the mouse to the keyboard will find easily, and take full advantage of, the GUI equivalents offered by S-Plus.

Despite the merits of this discourse on the virtues of platform independence, I will presumably err on the side of Windows users since after all, most of the participants in this hypothetical introductory class are working with a laptop running a version of the Windows operating system.

CREATING S OBJECTS

Typing the command

```
> X <- 1:16
```

in the Command Window, or at the S-Plus prompt under Unix or Linux, creates a vector of length 16 containing the first 16 integers in increasing order. Notice the arrow "$<-$" which is typed by using first the "less than" key "$<$" and then the minus sign "$-$". This combination reads "X gets ... to avoid the possible confusion with left arrow key. It is very important in S-Plus, since it provides an assignment command: whatever is on the left of this "gets" is an S object created by S-Plus with the result of the command on the right of the "gets"sign. If an object with this name already exists, it is automatically replaced. In fact, any symbol appearing on the left of the "gets" sign, will be used as the name of an object which S-Plus creates and saves in the local .Data directory. All these saved objects can be found in the Object Explorer under Windows.

Note that the same result would be obtained with the command:

```
> X <- seq(from=1,to=16,by=1)
```

To understand what the function seq does, we can use the online help by typing:

```
> help(seq)
```

Now that we know what the function seq can do for us, we can type:

```
> help
```

and notice that, instead of getting a help file of some kind, we get something which looks more like code. This is a general rule: if one types the name of an S method or function without parentheses, the system returns the S-code of the function in question. Only if we add the parentheses, will we see the command be executed (or an error message returned telling us that we did not included the parameters expected by the function). We can experiment further by typing:

```
> ls()
```

which lists all the S objects contained in the _Data subdirectory, and

```
> ls
```

which merely gives the code of the ls method. The command:

```
> dim(X) <- c(4,4)
```

reshapes the one dimensional vector X into a 4 by 4 matrix. The command:

```
> Y <- X*X
```

creates a new S-object. It is also a 4 by 4 matrix. Its entries are obtained by multiplying X by itself *entry by entry*. The symbol * does not give the usual matrix product. The latter is obtained by sandwiching the * symbol in between % 's as in:

```
> Z <- X%*%X
```

Z is now the 4 by 4 matrix equal to the usual matrix product of X by itself. To illustrate further the difference between these products we can do:

```
> dim(X) <- c(8,2)
> Y <- X*X
> Z <- X%*%X
```

and even though Y is computed correctly as the *entry by entry* product of X by itself, the computation of Z is not performed and we get an error message. After all, one cannot multiply (in the usual sense of the product of matrices) a 8 by 2 matrix by a 8 by 2 matrix. But notice that the command:

```
>  Z <- X%*% t(X)
```

does not trigger an error message. This is because the S command t(X) computes the transpose of the matrix X, and that consequently, the dimensions of the matrices involved in this matrix product match the usual requirements of the product of matrices.

Exiting S-Plus can be done at any time with the menu item **File** ▷ **Exit**, or by typing the command:

```
> q()
```

in the command window, or even by closing the S-Plus window. There is not need to save your work if you are working under Unix or Linux. Indeed S-Plus automatically saves all the objects you created during the session. You can check that these objects have been saved by listing the objects in your .Data directory if you are working under Unix. As a consequence of this automatic save, the Unix users of S-Plus need to clean up house regularly in order to avoid using too much disk space with all the S objects saved silently by the program. Things are slightly different in the Windows environment. When you quit the program, a dialog box prompts you to decide which, among the objects created during the session, do you want to save to the disk.

RANDOM GENERATION AND WHITE NOISE

The commands

```
> WN <- rnorm(1024)
> tsplot(WN)
```

create a vector WN of length 1024, and produce a sequential plot of the entries of this vector. The entries of WN are realizations of independent normal random variables with the same distribution $N(0,1)$. We specified the length of the random vector but we did not specify the values of the parameters of the distribution. S-Plus uses the default values 0 and 1 for the mean and the standard deviation of the normal distribution. The function tsplot gives a sequential plot of the vector WN, by plotting its values against the variable which takes the values $1, 2, \ldots, 1024$.

NB: A command WN <- rnorm(1024,1.2,4.0) would have created a sample of the same size 1024 of realizations of i.i.d. variates from the normal distribution with mean 1.2 and standard deviation 4, in other words the distribution $N(1.2, 16)$ if we use the standard notation used in the text. The commands:

```
> par(mfrow=c(2,1))
> tsplot(WN)
> tsplot(WN[1:64])
> par(mfrow=c(2,1))
```

divide the graphics window into two subplot areas, one on the top of the other, give a time series plot of the full WN vector on top, a plot of the time series of the first 64 entries of the vector WN, and reset the graphic window to a single plotting area. The results are shown in Figure 7.21. The top plot shows what we should expect from a

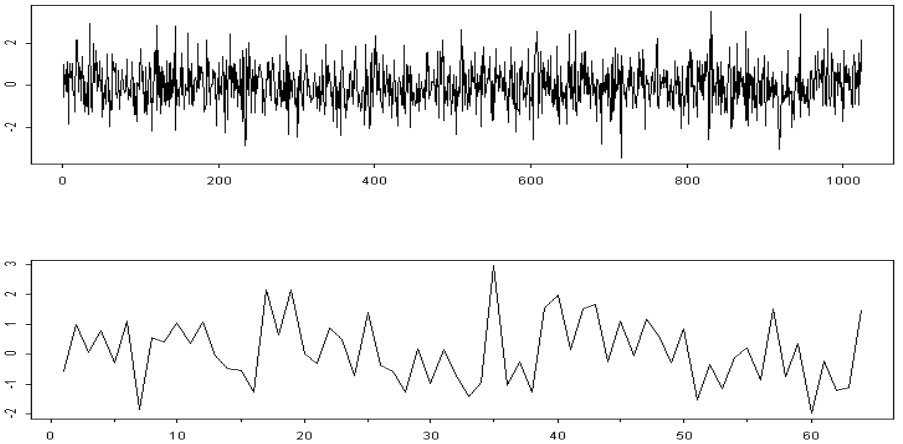

Fig. 7.21. Time series plot of the full white noise series WN (top) and of the series of its first 64 entries (bottom).

white noise: very erratic because of the lack of correlation of the successive entries of the series. Identifying a white noise is of crucial importance to a statistician. Indeed, most model fitting efforts are devoted to the identification and the extraction from the data of organized structures. This is usually done up until the remainder terms (which we call the residuals) form a white noise. From that point on, it is unreasonable to try to go any further, and statistical inference should take place at that time, and definitely before we start trying to fit the noise by mistake.

Strangely enough, the bottom plot of Figure 7.21 looks smoother. The reason is simple: it is viewed at a different scale. We plotted one sixteenth of the data on a plot with the same width! The *white noise look* which was identified earlier has practically disappeared. It is important to realize that, looking at the same data at different scales will leave different visual impressions. Don't be fooled by this artifact of the graphic settings used.

In the above commands, we did subscript the vector WN to consider only the set of its first 64 entries. To do so, we specified the range of the indices we wanted to keep. Subscripting is also conveniently used to extract sub-matrices of matrices. For example, the command `X[1:3,1:3]` will produce the 3 by 3 matrix in the left most

corner in the top of `X` (recall that `X` was defined earlier) while `X[,2]` will produce a vector equal to the second column of `X`.

MORE FUNCTIONS AND `for` LOOPS

The function `diff` should be viewed as a discrete analog of the operation of differentiation for functions of continuous variables. We learned in calculus that the inverse operation is integration. The discrete analog is given by the function `cumsum`. If we apply to the white noise vector `WN`:

```
> WN <- rnorm(1024)
> RW <- cumsum(WN)
```

we get a numeric vector `RW` which represents a sample of length 1024 from a random walk.. The sequential plot of `RW` (i.e. the plot of the values of `RW` against the succesive integers) is given in the left pane of Figure 7.22. Let us now try to create an object according to the Samuelson's model for stocks. According to this model, the time evolution of a stock is represented by the exponential of a random walk (with a given volatility) with drift given by the rate of appreciation of the stock. More precisely, we define the objects `SIG`, `MU`, `TIME` and `STOCK` by:

```
> DELTAT <- 1/252
> SIG <- .2*sqrt(DELTAT)
> MU <- .15*DELTAT
> TIME <- (1:1024)/252
```

for the volatility, the rate of return, the time (expressed in years) and an initial zero vector for the stock. With these objects at hand, one should be able to fill in the entries of the vector `STOCK` with the commands:

```
> for (I in 1:1024) STOCK[I] <- exp(SIG*RW[I]+MU*TIME[I])
```

This command is typical of the use of the `for` loops in `S-Plus`. Unfortunately, in `S-Plus` like in all other interpreted languages, computations with loops are very slow, and they should be avoided whenever possible. There is a very simple way to avoid the above loop. Indeed, most numerical functions in `S-Plus` when applied to a vector (resp. matrix, array, ...) will create a vector (resp. matrix, array, ...) with entries given by the computations of the function on the entries of the vector (resp. matrix, array, ...) passed as argument to the function. So in the above example, the command:

```
> STOCK <- exp(SIG*RW+MU*TIME)
```

will give the same result, and the computations will be much faster. The plot of the entries of this vector is given in the right pane of Figure 7.22. The latter was produced with the commands:

```
> par(mfrow=c(2,1))
> tsplot(RW)
> title("Sample of size 1024 from a random walk")
> plot(TIME,STOCK,type="l")
> title("Corresponding geometric random walk with drift")
> par(mfrow=c(1,1))
```

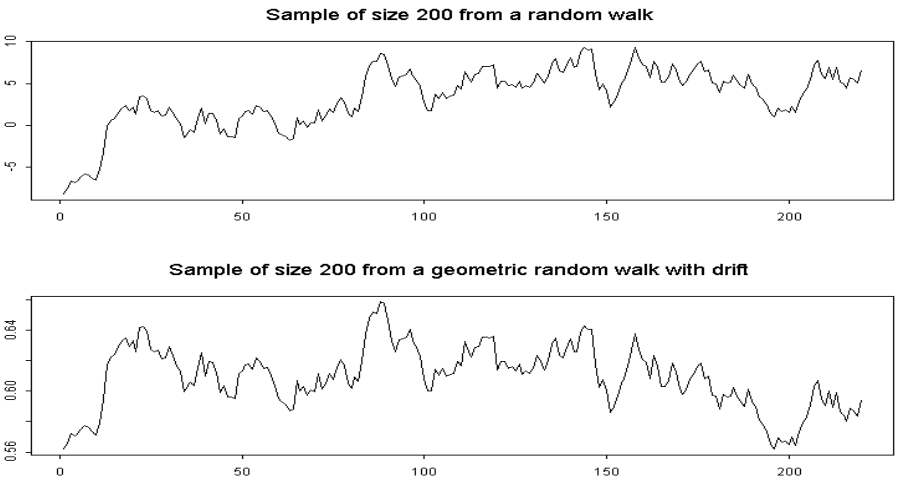

Fig. 7.22. Sequential plot of random walk sample `RW` (left) and of the corresponding geometric random walk with drift (right).

Notice that we use the command `plot` instead of `tsplot`. We need to pass two vectors to the command `plot`. The first one gives the values of the first coordinates of the points to plot while the second vector gives the second coordinates of these points. We use the option `type="l"` to joint the points by straight line segments to get the visual impression of a continuous curve instead of isolated points. The plots look very similar, until we notice the difference in scale on the vertical axes.

IMPORTING DATA

In most data analysis applications, the data are not created within the statistical package used for the analysis: they have to be imported. We give several examples as illustrations. These examples can be used as templates for further applications and to work out the problems (homework assignments) given in the text.

Using Data Already in S-Plus

In some instances in the text, we use data sets contained in the S-Plus distribution package. Let us consider for example, the data of the record times of the Scottish races. The data set is in the form of a data frame called hills, and it is part of the library mass which we need to link to our S-Plus session. See below for the way to link the library. We can choose to manipulate the variables by using their names hills$dist, hills$climb and hills$time, or we can decide to avoid having to type over and over the data frame name hills, in which case we need to "attach" the data frame to the S-Plus session. This is done in the following way.

```
> library(mass)
> attach(hills)
```

Doing so makes it possible to use all the variables in this frame by simply referring to their names. For example, the following commands produce the results shown in Figure 7.23.

```
> plot(climb,time)
> plot(dist,time)
```

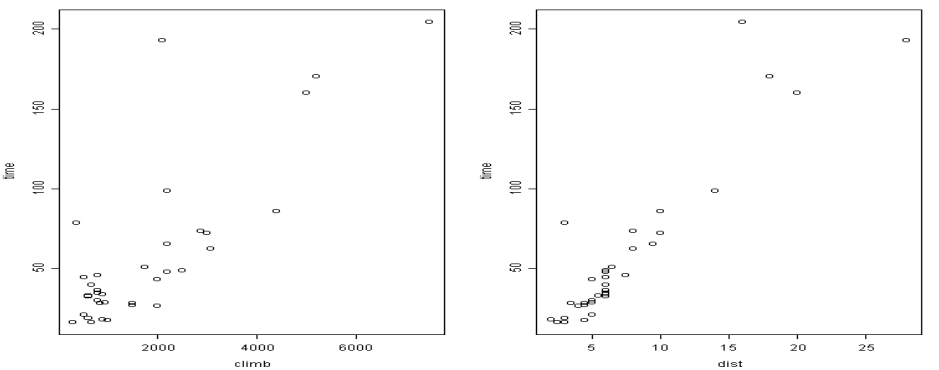

Fig. 7.23. Scatterplot of the record time versus climb (left) from the hills data frame, and of the time versus the distance variable dist (right).

Importing Text Files in S-Plus

As an example, we show how to import the text (ASCII) file HOWAREYOU which is part of the directory C:\My Documents\ORF405\book\data containing all

the data sets used in the book. We can open the file with a text editor to check its contents. We show the first seven rows of the file for the sake of illustration.

```
0.0359678408504965
0.0635196843758394
0.0750752080020106
0.0628095887777292
0.031177878417794
-0.00520235015344319
-0.0294474884975495
```

Under Windows, we import this file into S-Plus with the dialog created by the **File** ▷ **Import Data** ▷ **From File**, and clicking on the button BROWSE to navigate to the location where the file is, and selecting it. The full pathname of the file HOWAREYOU appears, its being an ASCII file also shows, and OK'ing all that allows S-Plus to create an S object called HOWAREYOU in the current data base. The same result could be obtained with the command:

```
> HOWAREYOU <- scan("C:\\My Documents\\ORF405\\book
                                        \\data\\HOWAREYOU")
```

which could also be used under Unix if the double backslashes \\ were to be replaced by single ones \. As the result of the next command shows, it is a vector of length 5151 and its first 8 elements are given by:

```
> length(HOWAREYOU)
[1] 5151
> HOWAREYOU[1:10]
 [1]  0.035968  0.063520  0.075075  0.062810
 [5]  0.031178 -0.005202 -0.029447 -0.031487 -0.013135
```

We use this vector in Chapter 5 to illustrate the construction of signal series objects in S-Plus

Importing S-Plus Dumps

The previous example was concerned with plain text files. Next, we consider the case of text files produced by S-Plus, and containing information about S-objects they were issued from. It is very easy to port files from one version of S-Plus to another. For example, files created and manipulated under UNIX or LINUX can be easily read and manipulated under Windows. S objects can be dumped to a text file with the dump command (see Help file for details), and the format of these text files is such that they can be read by any other version of S-Plus irrespective of the platform in question. A text file resulting from dumping S objects should be open under Windows from the dialog created by **File** ▷ **Open**, or by double-clicking the open folder icon in the task bar. S-Plus creates a script window, and displays the contents of the text file in this window. Hitting the **F10** key, or running the script

using the menu item **Script** ▷ **Run F10**, is what is needed to create the desired object. These dumps are text files and as such, they can be opened and edited with a text editor. We shall use the extension `.asc` to distinguish them from regular text files for which we try to use the extension `.txt`. Most of the data sets provided for illustration purposes or because they are needed for the problem sets, will be in the form of script files with the extension `.asc`.

Getting Data from the Web

We now show how to import data from the web into `S-Plus`. In the process, we show how starting from scratch, we can perform the analysis of the daily S&P 500 log-returns which we presented in Chapter 1.

We first retrieve the data from the internet. Let us start our favorite web browser and go to

`http://www.yahoo.com`.

Once there, we double click on the Finance special icon/button at the top of the page, or on the link Finance from the list of links right underneath the search box in the middle of the page. Once on the `Finance` page, we double click on the link US Indices under the search box. Next we double click on the Chart link of the Standard and Poor's 500 Index section. Finally, we double click on the link Daily Historical Quotes at the bottom of the page. We choose the start date January 1, 1960 and end date June 6, 2003 (now you know when this part of the book was prepared !) and we download the resulting table in spreadsheet format. We choose a location where to save the file. For the sake of definiteness, I call `filename` the full pathname of the file.

Warning. Web sites change frequently. Yahoo is no exception, so some of the detailed instructions given above may be obsolete by the time the book appear in print, or the reader tries to follow them: improvisation may be the only way out then !

The following instructions are for `Windows` only. See for example the next subsection below for instructions appropriate for the Unix/Linux platforms.

Next we switch to `S-Plus` and we open the dialog box under **File** ▷ **Import Data** ▷ **From File** and use the button **Browse** to get the pathname of the file to appear in the dialog box. Once `ASCII file - comma delimited (csv)` and `DSP500` appear in the `File format` and `Data set` spaces we click `OK`, and *voila* ! the data frame `SP500` appears on the desktop. A sequential plot of the daily close prices can be obtained with the command:

```
> tsplot(DSP500$Close)
```

Note that the dollar sign is used to extract a column from a data frame using the name of the column. The graphic window appears on the desktop and shows the graph reproduced in the left pane of Figure 7.24. We shall revisit this procedure when we study time series and we introduce the `timeSeries` objects used by `S-Plus` for

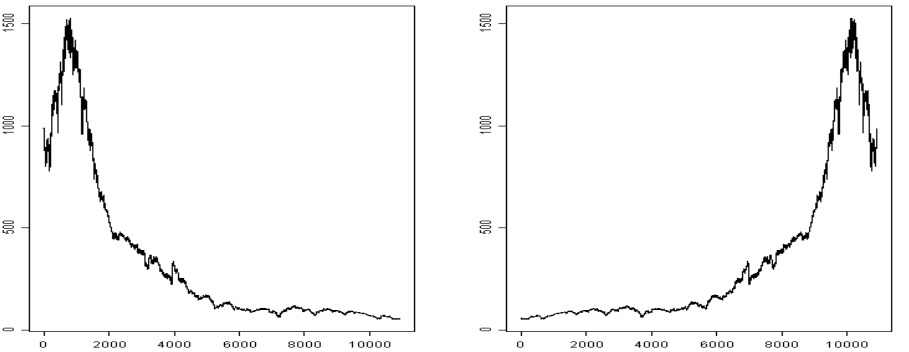

Fig. 7.24. Daily S&P 500 closing prices as imported from the web (left) and same data after we revert to the natural order (right).

their analyses. Obviously, something is wrong. The data were read in the reverse order of what would be natural. So we revert to the natural order using the function `rev` provided by `S-Plus`. We create a new `S` object of the numerical vector class:

```
> DSP <- rev(SP500$Close)
> tsplot(DSP)
```

and we compute and plot the daily log returns with the command: The result is reproduced in the right pane of Figure 7.24. Things look right now.

```
> DSPLR <- diff(log(DSP))
> tsplot(DSPLR)
> title("S&P 500 daily log-returns")
```

The logarithm of the daily close is computed with the function `log`, and the difference of the resulting vector is computed with the function `diff`. Check the help file for details on the function `diff`. Notice that we ignore the weekends (i.e. Monday's close follows the preceding Friday's close) and holidays, and more generally the days the market was close like the week of September 12, 2001 following the terrorist attack on the World Trade Center in New York.

Using the functions `mean` and `var` we compute the mean and the standard deviation of the daily log return. We use the function `sqrt` to compute the square root of the variance.

```
> mean(DSPLR)
[1] 0.0002717871
> sqrt(var(DSPLR))
[1] 0.009175474
```

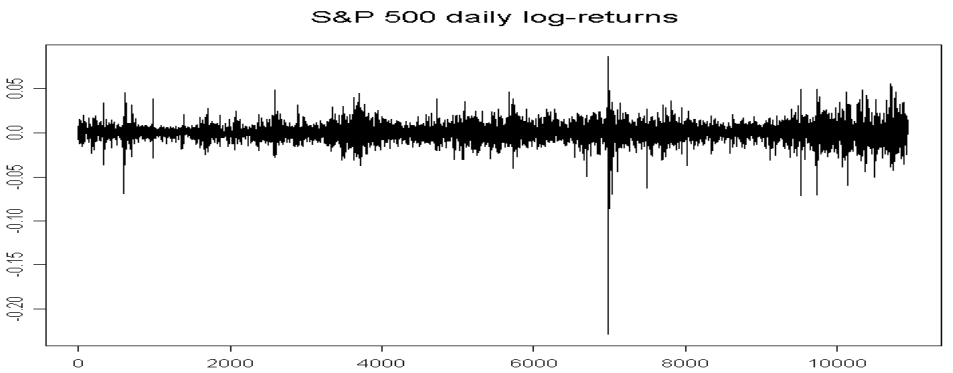

Fig. 7.25. Sequential plot of the daily S&P 500 log-returns.

Looking at the sequential plot of the daily log return (as reproduced in Figure 7.25) we notice we notice a few very large negative values. Looking at the largest of these down moves we see that:

```
> min(DSPLR)
[1]  -0.2289972
> (min(DSPLR)-mean(DSPLR))/sqrt(var(DSPLR))
[1]  -24.98716
```

which shows that this down move was essentially 25 standard deviations away from the mean daily move. So much for the normal distribution as a model for the daily moves of this index. This single day of October 1987, cannot be accounted for by a normal model for the daily log returns !

Still Another Example

Let us illustrate one more of the typical features of data sets which we have to face before we can actually begin the statistical analysis. We show how to handle the presence of headers. The text file `BondPrices61300.txt` contains information downloaded from `Data Stream` about the prices of the treasuries traded on June 6, 2000. Opening this file with a text editor shows the existence of a 12 lines header which we would like to ignore when importing the data into `S-Plus`. As illustrated before, the file can be imported using the menu or a command which can also be used under `Unix`. One can use **File** ▷ **Import Data** ▷ **from File** from the `S-Plus` menu if we are working under `Windows`, then browse the disk to locate the file as before, but this time, we should state after clicking the `Options` tab, that we want to start on row 13 only. Equivalently, one can use the command:

```
> BP61300 <- read.table('BondPrices61300.txt',skip=12,
```

```
                col.names=c('NAME','YEAR','EXPIRY','PRICE',
          'YIELD',RED.YIELD''ACCR.INT.','LIFE','VARIATION'))
```

which can also be used if we are working under Unix. This command creates a data frame, whose first five rows can be printed in the command window. We only reproduce the first six columns.

```
> BP61300[1:5,]
            NAME YEAR EXPIRY  PRICE YIELD RED.YIELD
 1 US.TRSY.BONDS 1997   2027  99.50  6.16     6.161
 2 US.TRSY.BONDS 1975   2005 100.03  8.25     8.241
 3 US.TRSY.BONDS 1977   2007 101.56  7.51     7.317
 4 US.TRSY.BONDS 1977   2007 102.78  7.66     7.375
 5 US.TRSY.BONDS 1978   2008 106.43  8.22     7.691
```

Notice that S-Plus labels the rows by the successive integers (in increasing order) by default, since we did not provide the read.table command with a value for the parameter row.names. The skip=12 part of the command line forced S-Plus to ignore the first 12 lines of the ASCII file w6.txt. We can get the same result under Windows by setting appropriately the relevant options appearing in the **Options** tab of the dialog box. In particular, one would have to decide which row should the reading start from, one should have to give the name of the column variables, ... If we want to plot the yield against the remaining life of these fixed income instruments, we can use the command:

```
> plot(BondPrices61300$LIFE,BondPrices61300$YIELD,type="l")
```

The result is shown in Figure 7.26. The extension $LIFE added to the name of the data.frame BondPrices61300 identifies the variable LIFE from this data frame. Similarly BondPrices61300$YIELD identify the YIELD variable. We used the option type="l" to join the points of the scatterplot by line segments, producing in this way a continuous curve, akin the graph of the yield against the time remaining to maturity. If we want to work for a while with these data without having to type the reference BondPrices61300 to the data frame over and over, we can attach the data frame with the command:

```
> attach(BondPrices61300)
```

and work with the variables in the columns of the data frame by referring directly to their names. For example the plot of Figure 7.26 can be equivalently produced by the command:

```
> plot(LIFE,YIELD)
```

NB: It is important to keep in mind the fact that, contrary to most Windows programs, S-Plus is case sensitive. This feature is presumably inherited from the Unix version of S-Plus. Consequently, BondPrices61300 is not the same thing as bondprices61300 !

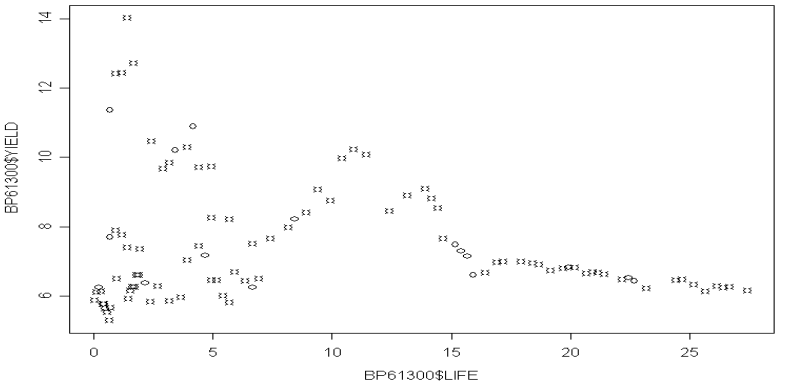

Fig. 7.26. Plot of the yield against the remaining life of the fixed income instruments contained in the text file `BondPrices61300`.

PROGRAMMING IN S-Plus: A FIRST FUNCTION

When we suspect that a set of S-Plus commands will be needed frequently, it is advantageous to encapsulate the individual commands in a script which we can save as a script, and which we will be able to re-use at our convenience. We can re-run individual commands or entire sets of commands by selecting them (highlighting them) and hitting the key **F10** or using the **Script** ▷ **Run F10** menu.

We can also structure the set of commands to be able to take parameters as arguments and save it as an S object which will be called a function whenever it acts on an argument. We did just that to create functions `qexp` and `myqqnorm` which we used earlier in the first chapter. We now give the details of an example which we borrow from the S-Plus manual. , we define function `eda.shape` defined by typing the following in the `command window`.

```
> eda.shape <- function(x)
+ {
+ par(mfrow = c(2,2))
+ hist(x)
+ boxplot(x)
+ iqd <- summary(x)[5] -summary(x)[2]
+ plot(density(x,width=2*iqd),xlab="x",ylab="",type="l")
+ qqnorm(x)
+ qqline(x)
+ par(mfrow=c(1,1))
+ }
```

The argument of this function needs to be a numeric vector x containing the numeric data which we want to visualize. Except for the boxplots which are hardly used in the text, all of these commands are used extensively throughout the book. We illustrate the use of this function with a sample of size 1024 which we generate from the standard normal distribution.

```
> eda.shape(rnorm(1024))
```

The result is reproduced in Figure 7.27. As expected, the histogram tries to reproduce

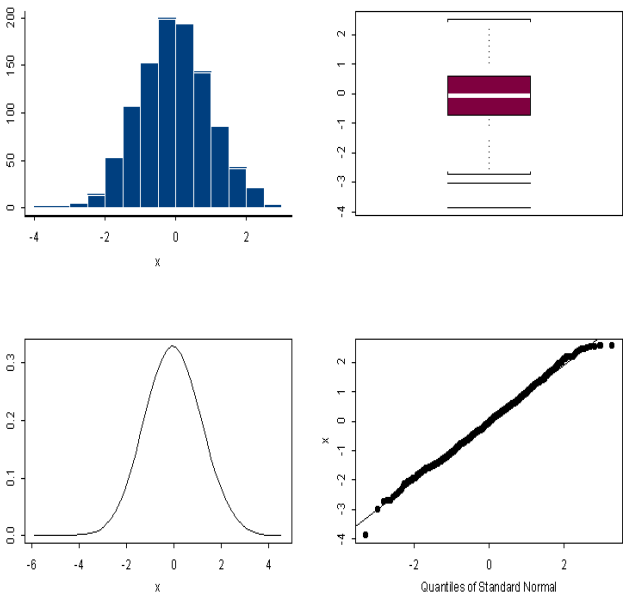

Fig. 7.27. Exploratory data analysis of a sample of size 1024 from the standard normal distribution.

the bell shape normal density, the kernel density estimate gives a faithful representation of the normal density function, and the points of the Q-Q plot are tightly packed around the diagonal. This shows that the random number generator used to create this sample did a reasonable job. At this stage, it is important to emphasize that the function `eda.shape` is useful for samples whose distributions are nearly normal. This is especially true for the histogram and density estimation parts of the exploratory data analysis. We will use it to check graphically, if the distribution of a sample is normal or does not depart too much from a normal distribution. Indeed, it can give strange results for distributions with drastically different features.

NOTES & COMPLEMENTS

When I first started to use `S-Plus` in teaching, I used to recommend the small book of Phil Spector to newcomers to `S` and `S-Plus`, especially if they were to work under `Unix`. More recently, I changed my recommendation to the thicker book by Krause and Olson [55]. It is best for an introduction to the language `S`, and to the functionalities (especially of the `Windows` version) of `S-Plus`. Advanced users can consult the book entitled `S` Programming by Venables and Ripley, however the *bible* remains the green book Programming with Data by John Chambers.[23]

References

1. Y. Ait Sahalia and A.W. Lo: Nonparametric Estimation of State Price Densities Implicit in Financial Asset Prices. *The Journal of Finance* **53** (1998) 499-547.
2. S. Amari: *Differential Geometrical Methods in Statistics.* Lect. Notes in Statistics **# 28** (1985) Springer Verlag, New York, NY.
3. N. Anderson, F. Breedon, M. Deacon, A. Derry, and G. Murphy: *Estimating and Interpreting the Yield Curve.* (1996) Wiley, Chichester.
4. A. Antoniadis, J. Berruyer and R. Carmona: *Régression Non-linéaire et Applications.* (1992) Economica.
5. S.P. d'Arcy and V.G. France: Catastrophe Futures: A Better Hedge for Insurers. *J. of Risk and Insurance,* **59** (1992)
6. L. Bachelier: *Théorie de la spéculation.* Thesis, (1900) Paris.
7. Bank for International Settlements: Zero-coupon Yield Curves. Technical Report March (1999).
8. E. Banks ed.: *Weather Risk Management.* (2002) Palgrave, New York, NY.
9. T. Bollerslev: Generalized Auto Regressive Heteroskedasticity. *J. of Econometrics* **31** (1986) 307-327.
10. T. Bollerslev, R.F Engle and J.M. Wooldridge: ARCH Models. In *Handbook of Econometrics,* **IV** (1994) 2959-3038.
11. G.E.P. Box and G.M. Jenkins: *Time Series Analysis: Forecasting and Control.* Revised Edition, (1976) Holden Day, San Francisco.
12. L. Breiman, J.H. Friedman, R.A. Olshen and C.I. Stone: *Classification and Regression Trees.* Wadsworth and Brooks/Cole. (1984) Monterey, CA.
13. P.J. Brockwell and R.A. Davis (1996): *Introduction to Time Series and Forecasting.* Springer Texts in Statistics, Springer Verlag, New York, NY.
14. R.L. Brown, J. Durbin and J.M. Evans: Techniques for testing the constancy of regression relationship over time (with comments). *J. Royal Statistical Society, ser. B* **37** (1975) 149-192.
15. A. Bruce and H.Y. Gao: *Applied Wavelet Analysis with S-plus.* Springer Verlag (1996) New York, N.Y.
16. J.Y. Campbell, A.W. Lo, and A.C. MacKinlay: *The Econometrics of Financial Markets.* Princeton University Press (1997) Princeton, NJ.
17. R. Carmona and J. Morrisson (2000): EVANESCE, an `S-Plus` Library for Heavy Tail Distributions and Copulas. *Dept. of Operations Research & Financial Engineering, Princeton University*. `http://www.princeton.edu/ rcarmona`.

18. R. Carmona, C. Chen, R.D. Frostig and S. Zhong: Brain Function Imaging: a Comparative Study. *Tech. Rep.* (1996) Princeton Univ.
19. R. Carmona, W. Hwang and R.D. Frostig: Wavelet Analysis for Brain Function Imaging. *IEEE Trans. on Medical Imaging* **14** (1995) 556-564.
20. R. Carmona, W. Hwang and B. Torresanni: *Time Frequency Analysis: Continuous Wavelet and Gabor Transform, with an implementation in* `Splus` (1998) Academic Press, New York, N.Y.
21. R. Carmona: *Interest Rates Mathematical Models: from Parametric Statistics to Infinite Dimensional Stochastic Analysis.*SIAM Monographs (2003) (to appear) SIAM, Philadelphia, PA
22. R. Carmona: Stochastic Analysis for Financial Engineering. Lecture Notes, Princeton University (2001) Princeton NJ.
23. J.M. Chambers: *Programming with Data: A Guide to the S Language.* MathSoft (1998) Seattle WA
24. N.H. Chan: *Time Series: Applications to Finance.* John Wiley & Sons, (2002) New York, N.Y.
25. Chicago Board of Trade: *A User Guide: PCS Catastrophe Insurance Options.* (1995)
26. W.S. Cleveland: *The Elements of Graphing Data.* Hobart Press, (1994) Summit, N.J.
27. L. Clewlow and C. Strickland: *Energy Derivatives, Pricing and Risk management.* Lacima Publications (2000) London.
28. C. de Boor: *A Practical Guide to Splines.* Springer Verlag, (1978) New York, NY.
29. D. Drouet Mari and S. Kotz: *Correlation and Dependence.* Imperial College Press (2001) London
30. B. Dupire: Pricing with a Smile. *Risk Magazine,***7** (1994).
31. P. Embrechts, C. Kluppelberg and T. Mikosch: *Modelling Extremal Events for Insurance and Finance.* Springer Verlag (1997) New York, N.Y.
32. R.F. Engle: Autoregressive conditional heteroskedasticity with estimates of the variance of the United Kingdom inflation. *Econometrica,* **50** (1982) 987-1006.
33. R.F. Engle and T. Bollerslev: Modeling the Persistence of Conditional Variances. *Econometric Reviews,* **5** (1986) 1-50.
34. R.F. Engle and C. Granger: Cointegration and Error-Correction: Representation, Estimation and Testing. *Econometrica,* **55** (1987) 251-276.
35. E.F. Fama and R.R. Biss: The Information in Long-Maturity Forward Rates. *American Economic Rev.* **77** (1987) 680-692.
36. J. Fan and Q. Yao: *Nonlinear Time Series: Nonparametric and Parametric Methods.* Springer Verlag (2003) New York, N.Y.
37. H. Föllmer and A. Schied: *Stochastic Finance: An Introduction in Discrete Time.* (2002) De Gruyter Studies in Mathematics. Berlin.
38. J.P Fouque, G. Papanicolaou and R. Sircar: *Derivatives in Financial Markets with Stochastic Volatility.* (2000) Cambridge University Press, London.
39. R. Gençay,F. Selçuk and B. Whitcher: *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics.* Academic Press (2002) New York, NY.
40. C. Gouriéroux: *ARCH Models and Financial Applications.* Springer Verlag (1997) New York, N.Y.
41. C. Gouriéroux and J. Jasiak: *Financial Econometrics.* Princeton University Press (2002) Princeton, N.J.
42. W. Haerdle: *Non-parametric Regression.* Springer Verlag (1994) New York, N.Y.
43. J.D. Hamilton: *Time Series Analysis.* Princeton University Press (1994) Princeton, NJ.

44. Hastie, Tibshirani, J. Friedman: *The Elements of Statistical Learning, Data mining, Inference and Prediction.* Springer Verlag (2001) New York, N.Y.
45. Haussdorff: Dimension und äusseres Mass. *Math. Annalen* **79** (1919) 157-179.
46. P.J. Huber: *Robust Statistics.* J. Wiley & Sons (1981) New York, NY.
47. J.C. Hull: *Options, Futures, and Other Derivatives.* Prentice Hall, 5th edition (2002) New York, NY
48. J.M. Hutchinson, A.W. Lo and T. Poggio: A Nonparametric Approach to the Pricing and Hedging of Derivative Securities via Learning Networks. *Journal of Finance,* **49** (1994)
49. S. Johansen: *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models.* Oxford University Press (1995) Oxford
50. P. Jorion: *Value at Risk: The New Benchmark for Managing Financial Risk.* 2nd ed. McGraw Hill, (2000) New York, NY
51. S. Karlin and H.W. Taylor: *A First Course in Stochastic Processes.* 2nd ed. Academic Press (1975) New York, N.Y.
52. C.J. Kim and C.R. Nelson: *State-Space Models with Regime Switching.* MIT Press, (1999) Cambridge MA.
53. G. Kitagawa: Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics,***5**, (1996) 1-25.
54. T. Kohonen: *Self-organizing Maps.* Springer Verlag (1995) New York, NY
55. A. Krause and M. Olson: *The Basics of S and S-PLUS.* 2nd Edition. Springer Verlag (2000) New York, N.Y.
56. D. Lamberton and B. Lapeyre: *Introduction to Stochastic Calculus Applied to Finance.* (1996) CRC Press.
57. R. Litterman and J. Scheinkman: Common factors affecting bond returns. *J. of Fixed Income,* **1**, 49-53.
58. B.B. Mandelbroit: New Methods in Statistical Economics. *J. of Political Economy,* **71** (1963) 421-440.
59. B.B. Mandelbroit: The Variation of Certain Speculative Prices. *J. of Business,* **36** (1963) 394-419.
60. B.B. Mandelbroit: *Fractal, Form Dimension and Chance.* W.H. Freeman & Co. (1977) San Francisco, CA.
61. K.V. Mardia, J.T. Kent and J.M. Bibby: *Multivariate Analysis.* Academic Press, (1979) New York, NY
62. T.C. Mills: *The Econometric Modelling of Financial Time Series.* 2nd Ed. Cambridge University Press, (1999) London
63. K.R. Miltersen and E.S. Schwartz: Pricing of Options on Commidity Futures with Stochastic Term Structure of Convenience Yields and Interest Rates. *J. of Financial and Quantitative Analysis,* (2000)
64. D.C. Montgomery and E.A. Peck: *Introduction to Linear Regression Analysis.* 2nd ed. John Wiley & Sons (1992) New York, NY
65. R.B. Nelsen: *An Introduction to Copulas.* Lect. Notes in Statistics **139** (1999) Springer Verlag, New York, NY.
66. C.R. Nelson and A.F. Siegel: Parsimonious Modeling of Yield Curves. *J. of Business* **60** (1987) 473-489.
67. D. Pilipovic: *Energy Risk: Valuing and Managing Energy Derivatives.* Mc Graw Hill (1997) New York, NY
68. S.R. Pliska: *Introduction to Mathematical Finance. Discrete Time Models.* Blackwell (1997) Oxford, UK.

69. M.B. Priestley: *Nonlinear and Non-stationary Time Series Analysis.* Academic Press, (1988) New York, NY.
70. R. Rebonato: *Interest-Rate Option Models: Understanding, Analyzing and Using Models for Exotic Interest-Rate Options.* Wiley (1996) New York, NY
71. P.A. Ruud: *An Introduction to Classical Econometric Theory.* Oxford University Press, (2000) New York, NY.
72. E. Renault and N. Touzi (1996): Option Hedging and Implied Volatility in a Stochastic Volatility Model. *Mathematical Finance* **6** 215-236.
73. J.Rice: *Mathematical Statistics and Data Analysis.* 2nd ed. Duxbury Press (1995) Belmont CA
74. R. Roll: A Critique of the Asset Pricint Theory's Test, Part One: Past and Potential Testability of the Theory. *Journal of Financial Economics* **4** (1977) 129-176.
75. M. Rosenblatt: *Gaussian and Non-Gaussian Linear Time Series and Random Fields.* Springer Verlag (2000), New York N.Y.
76. P.J. Rousseeuw and A.M. Leroy: *Robust Regression and Outlier Detection.* J. Wiley & Sons (1984) New York, NY.
77. L.O. Scott: Option princing when the variance changes randomly: Theory, estimation and application.*Journal of Financial and Quantitative Analysis* **22** (1987) 419-438.
78. R.H. Shumway and D.S. Stoffer: *Time Series Analysis and Its Applications.* Springer Verlag (2000) Newy York, NY.
79. B.W. Silverman: *Density Estimation for Statistics and Data Analysis.* Chapman & Hall (1986) London.
80. P. Spector: *An Introduction to S and S-Plus.* Duxbury Press (1994) Belmont, CA.
81. J.M. Steele: *Stochastic Calculus and Financial Applications.* Springer Verlag (2000) New York NY
82. L.E.O. Svensson: Estimating and Interpreting Forward Interest Rates: Sweden 1992-94. *NBER working Paper Series #* **4871** (1994)
83. R.S. Tsay: *Analysis of Financial Time Series.* John Wiley & Sons, (2002) New York, N.Y.
84. O. Vasicek and G. Fong: Term structure estimation using exponential splines.*J. of Finance,* **38** (1982) 339-348.
85. W.N. Venables and B.D. Ripley: *Modern Applied Statistics with S-PLUS.* 2nd Edition. Springer Verlag (1997) New York, N.Y.
86. H. von Storch and F.W. Zwiers: *Statistical Analysis in Climate Research.* Cambridge University Press (1999) New York, NY.
87. M.V. Wickerhaüser: *Adapted Wavelet Analysis: from Theory to Software.* A.K.Peters LTd (1994) Wellesley, MA
88. I.H. Witten and E. Frank: *Data Mining.* Academic Press (2000) San Diego CA.
89. E. Zivot and J. Wang: *Modeling Financial Time Series with S-PLUS.* Springer Verlag (2003) New York, N.Y.

Notation Index

Data Set Index

S-Plus Index

Author Index

Subject Index