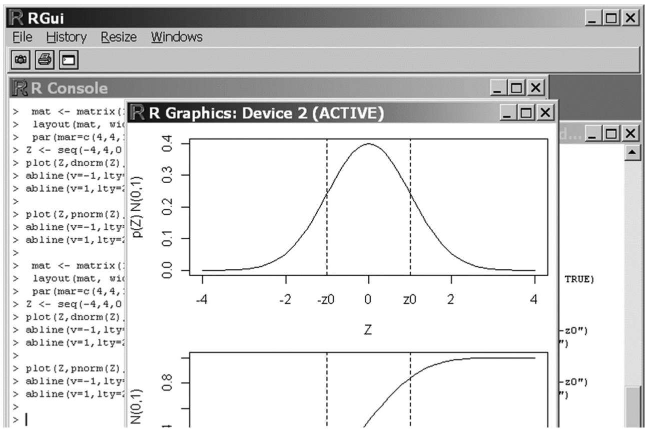


```
>  mat <- matrix(
>   layout(mat, wi
>   par(mar=c(4,4,
> Z <- seq(-4,4,0
> plot(Z,dnorm(Z)
> abline(v=-1,lty
> abline(v=1,lty=
>
> plot(Z,pnorm(Z)
> abline(v=-1,lty
> abline(v=1,lty=
>
>  mat <- matrix(
>   layout(mat, wi
>   par(mar=c(4,4,
> Z <- seq(-4,4,0
> plot(Z,dnorm(Z)
> abline(v=-1,lty
> abline(v=1,lty=
>
> plot(Z,pnorm(Z)
> abline(v=-1,lty
> abline(v=1,lty=
>
> |
```

```
TRUE)
-z0")
")
-z0")
")
```

FIGURE A.5 R graphics windows.

A.2.3 Graphics Device

This is where you receive graphical output interactively (Figure A.5). To print a hard copy of a graph, you can just select **File|Print** from the menu, while the graphics window is selected. To save in a variety of graphics formats, use **File|Save As**. These formats include metafile, postscript, bmp, pdf, and jpeg. To capture the image on the clipboard and then paste on another application, you could simply use **Copy and Paste** from the graphics window. To do this you can also use **File|Copy to clipboard**, and then paste the clipboard contents on the selected cursor position of the file being edited in the application. Notice that one can work with windows as usual, go to **Windows** in the menu bar; here you can **Tile** the windows or **Cascade** the windows, and so on. In addition, this is where you can open the console window.

A.2.4 Package Installation and Loading

Packages are set up to use by following two steps: (1) installation from the Internet and (2) loading it for use. First, to install from the RGui: Go to **Packages|Install package(s)**, select a mirror site depending on your location, and then select Install package. For example, select the **renpow** package, which is the package developed for this book. The download process starts and will give you messages about progress and success or not of the package installation. You do not need to repeat the installation as long as you do not reinstall the R software or want to update the package.

The second step is to load the package. From the RGui: Go to **Packages|Load package**, and then select the package. For example, browse and select renpow. Alternatively, you can run the function **library** from the R console by typing library (renpow). Help on functions provided by renpow would now be available. For example, go to HTML help, and when you click on packages, you will see renpow. Datasets in renpow are used throughout the book as examples. You can find the available datasets by typing data(package='renpow').

These datasets are based on data files in the folder **inst\extdata** of package renpow and are used throughout this book as examples. There are three different ways of using these data files. One way, is to get the data files from the CRAN website; for this, use the link Packages on the left side navigation list, then go to the table listing the package by name. Once you click on the package name, you will have the option of downloading the package source as *.tar.gz, then you can extract the inst \extdata folder from this archive. Copy this folder under your working directory labs, so that you will find a data file using path **extdata**. Another way is to read the file from the installation directory using the function system.file("extdata",datafilename,package="renpow") where datafilename refers to the target data file. ~~A third way is to read the file from the installation directory using the function~~

~~system.file("extdata",datafilename, package="renpow") where datafilename refers to the target data file.~~ A third way is to use command data( ) in R to load the data set corresponding to the file. Once the data set is loaded, you can use it and address its contents as a **data frame** or data set object. A shortcut is simply referring to the dataset as the file name excluding the extension. All three alternatives are explained in Section A.3.1 using a simple example.

Part of my intent writing package renpow, which supports this textbook, is to bundle many lengthy calculations into functions requiring a single call or at most a sequence of a few lines of code. Most functions of renpow have been described and used in the textbook in the context of specific examples. More options and details are available from the help of renpow.

A.3 SIMPLE TASKS: DATA FILES, GRAPHS, ANALYSIS

A.3.1 Read a Simple Text Data File

Before you write code to import or read a data file, it is a good practice to examine the file and understand its contents. Assuming that you copied the folder extdata under your working directory labs, we can use the file extdata/test100.txt as an example. For instance, use the Notepad to look at file extdata/test100.txt. You will see that it has a header x specifying the variable name and that it is just one field per line with no separator between fields (Figure A.6, left-hand side).

A more convenient text editor is Vim, an open source program that you can download from the Internet (www.vim.org). Some nice features are that you get line numbers and position within a line (Figure A.6, center), a more effective find/replace, and tool and color codes for your script, if you choose to write the scripts on this editor. Another option is the Notepad++ editor available in Windows, which provides line numbers and convenient programming setup for a variety of languages (Figure A.6, right-hand side).

Make sure you set **File|Change Dir** to the working folder labs so that the path to the file is relative to this folder. If you have personalized the R shortcut to start in folder labs, then the path to the file is relative to this folder, for example, in this case extdata/test100.txt. Therefore, you could use this name to scan the file.

```
> scan("extdata/test100.txt", skip=1)
```

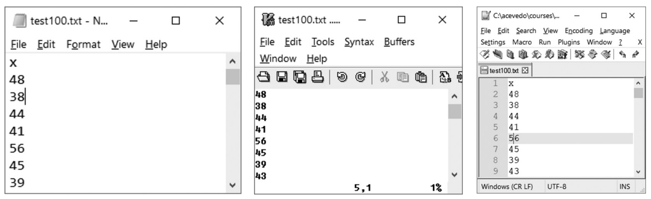
On the console, we receive the response

```
Read 100 items
  [1] 48 38 44 41 56 45 39 43 38 57 42 31 40 56 42 56 42 46 35 40 30 49 36 28 55
 [26] 29 40 53 49 45 32 35 38 38 26 38 26 49 45 30 40 38 38 36 45 41 42 35 35 25
 [51] 44 39 42 23 44 42 52 55 46 44 36 26 42 31 44 49 32 39 42 41 45 50 39 55 48
 [76] 49 26 50 46 56 31 54 26 29 32 34 40 53 37 27 45 37 34 32 33 35 50 37 74 44
```

Alternatively using the file from the installed package directory we can type

```
scan(system.file("extdata","test100.txt", package="renpow"),skip=1)
```

Next, create an object x100 by scanning an input data file


test100.txt - N...
File Edit Format View Help
test100.txt .....
File Edit Tools Syntax Buffers
Window Help
C:\acevedo\courses\...
File Edit Search View Encoding Language
Settings Macro Run Plugins Window ?
test100.txt
Windows (CR LF) UTF-8 INS

```
x
48
38
44
41
56
45
39
```

```
48
38
44
41
56
45
39
43
          5,1          1%
```

```
1 x
2 48
3 38
4 44
5 41
6 56
7 45
8 39
9 43
```

FIGURE A.6 Example of a text file. Viewed with the notepad text editor (left), the Vim text editor (center), and the Notepad++ (right).

```
500    1.357
1000   1.321
```

these

We will see how to read a file like this, and create a data set or data frame object containing ~~this~~ data. For this purpose, we use R function **read.table** in the following manner, which specifies that we have a header line and the values are separated by blanks.

```
X <- read.table("extdata/CpCvT.txt",header=TRUE,sep="")
```

or alternatively

```
X <- read.table(system.file("extdata","CpCvT.txt", package="renpow"),
header=TRUE,sep="")
```

Using the third alternative, we could have obtained the object directly by typing X <- CpCvT at the console.

The header strings become the names of the variables in the data set and each row is assigned a number, as we can see when we query X:

```
> X
   TC  cpcv
1  -40 1.401
2  -20 1.401
3    0 1.401
4    5 1.401
5   10 1.401
6   15 1.401
7   20 1.401
…
```

We can refer to one of the variables by using the name of the data set followed by $ and the name of the variable. For instance, X$TC is temperature.

```
> X$TC
 [1]  -40  -20    0    5   10   15   20   25   30   40   50   60   70   80   90
[16]  100  200  300  400  500 1000
>
```

You can use **dim** to check the dimensions of the data set to be 21 rows and two columns

```
> dim(X)
[1] 21  2
>
```

For brevity, the names of the components can be addressed directly once we **attach** the data frame.

```
> attach(X)
```

For instance, to plot variable cpcv as a function of TC (Figure A.9, left-hand side) we can refer to TC and cpcv directly instead of X$TC and X$cpcv

```
> plot(TC,cpcv)
```

You can use a line graph by adding an argument **type="l"** to the plot function. Careful, this is a lowercase letter "l" for line not the number one "1" (Figure A.9, right-hand side).

```
> plot(TC,cpcv, type="l")
```