Chapter 2
R and RStudio

Abstract We have an initial look at R and RStudio. In R we work with objects, using commands that have to be precise, (for example, we must be careful about where we use parentheses and brackets). We use four types of objects frequently—vectors, matrices, data frames and lists. We often act on the whole or part of an object, so we need to refer to the whole or part of the object precisely.

2.1 R and RStudio

R (R Core Team 2013) is a highly flexible software. It is free. We can download it from:

http://www.r-project.org/

In this book we work with R via RStudio, which makes our work easier. We can download RStudio (after installing R) from:

http://www.rstudio.com/

If we experience any difficulty while downloading R or RStudio, we can simply use Google. For example, we could just search in Google for "Installing R". In general, using Google is a good idea when working with R.

Once we have R and RStudio installed, we only need to run RStudio.

Figure 2.1 is a screenshot of RStudio—there are four windows:

- Script or editor window. The top left window with the dark background is the window with an R script. We should always type our commands in an R script. By highlighting select code and clicking on run, we can run the selected lines of code.
- Console window. The bottom left window with the dark background is the console window—this is where the output from R appears. There is a tab that says Console. We can type commands at the 'greater than' prompt, but it is better to use scripts.
- The Environment or History window. The top right window has Environment and History tabs—different objects appear here as you create them. Under the Environment tab is 'Import Dataset', which we will use to import data into RStudio.
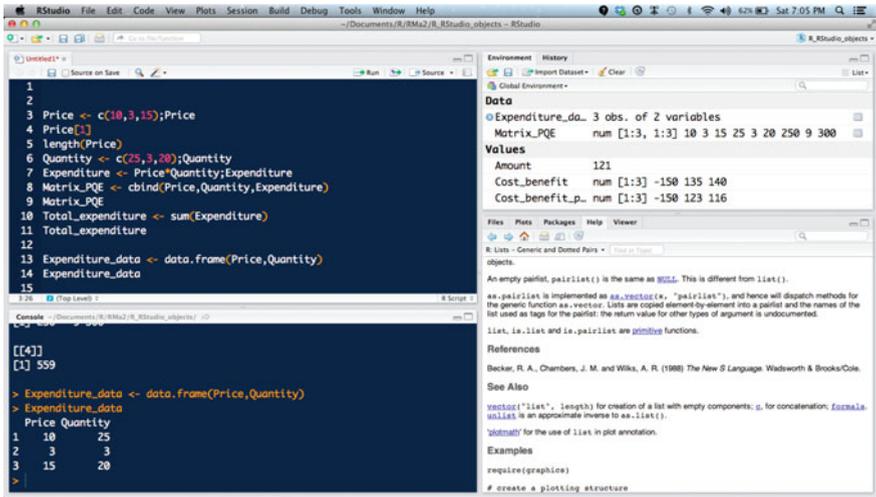
Fig. 2.1 RStudio windows

- Plots etc. window. The bottom right window has the following tabs: Files, Plots, Packages, Help, and Viewer. When graphs are made, they can be viewed here using the Plots tab. Packages can be installed with the Packages tab.

The four windows can be arranged depending on where we prefer to have them—- top or bottom, right or left.

2.2 Working Directory: Projects

One of the most useful features of RStudio is the projects facility. This helps us a great deal with housekeeping; files and directories are arranged for us. We can create a new project by going to File, then New Project. We can create a project and a new directory at the same time or we can create a new project in a directory. All output and files get saved in the same directory.

2.3 Script

We can start working with a script as follows. First, in RStudio we click on File, then New File, then Script. We can save it as 'Script'. We can type in $2 + 3$, and click on Run; RStudio prints the result in the Console window. We can save the Script.

```
> 2 + 3

[1] 5
```

2.4 Different Objects in R

In R, we work with objects of different types. Let us use a simple example to examine four important objects: vector, matrix, dataframe and list.

2.4.1 Vectors

We set up a vector called Price, consisting of three prices. We need to type the following in the script window, and then click on Run, which runs that line. Then the line appears in the console window.

```
> Price <- c(10, 3, 15)
```

In this book, R code follows the prompt (or greater than symbol) in typewriter font. The resulting output is also indicated (without the prompt) in typewriter font.

The three prices are equal to 10, 3 and 15. We use c which stands for concatenate, and parentheses enclose the values that are separated by commas.

When we run the command above, we don't see any output. R simply creates the object called Price, and you can see it in the Environment window. To print it, we need to type Price and run the line:

```
> Price
```

```
[1] 10  3 15
```

We notice that the output includes [1]; this only tells us that the first element is ten.

R will distinguish between Price and price; if we are not careful we get an error message.

```
> # Price and price are different
> price
```

```
Error: object 'price' not found
```

In R, a parenthesis ( ) is different from a bracket [ ]—each has to be used in the right way depending on the context.

```
> Price <- c[10, 3, 15]
```

```
Error: object of type 'builtin' is not subsettable
```

We can create a long vector in R with:

```
> 1:40
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
[16] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
[31] 31 32 33 34 35 36 37 38 39 40
```

Returning to our vector Price, we can find out its length:

```
> length(Price)
```

```
[1] 3
```

We can extract the first element:

```
> Price[1]
```

```
[1] 10
```

and the second and third elements

```
> Price[2:3]
```

```
[1]  3 15
```

We create a vector for corresponding quantities and print it:

```
> Quantity <- c(25, 3, 20)
> Quantity
```

```
[1] 25  3 20
```

We can multiply the Price and Quantity vectors, which gives us Expenditure.

```
> Expenditure <- Price * Quantity
> Expenditure
```

```
[1] 250    9 300
```

The sum of the elements of Expenditure gives us total Expenditure.

```
> Total_expenditure <- sum(Expenditure)
> Total_expenditure
```

```
[1] 559
```

2.4.2 Matrices

The Price, Quantity and Expenditure vectors can be bound into the columns of a matrix using the `matrix` function:

```
> Matrix_PQE <- matrix(data = cbind(Price, Quantity,
+       Expenditure), ncol = 3)
> Matrix_PQE
```

```
     [,1] [,2] [,3]
[1,]   10   25  250
[2,]    3    3    9
[3,]   15   20  300
```

We used the R function *matrix* above, and also the function *cbind*, which binds the vectors into columns.

We print the first row of the matrix.

```
> Matrix_PQE[1, ]
```

```
[1]  10  25 250
```

and then the second column.

```
> Matrix_PQE[, 2]
```

```
[1] 25  3 20
```

First row, second column:

```
> Matrix_PQE[1, 2]
```

```
[1] 25
```

The first number between the brackets indicates the row, the second the column. We discuss matrices in R in a later chapter.

2.4.3 Data Frames

We can create a data frame and print it:

```
> Exp_data <- data.frame(Price, Quantity)
> Exp_data
```

```
  Price Quantity
1    10       25
2     3        3
3    15       20
```

We print the second column.

```
> Exp_data[, 2]
```

```
[1] 25  3 20
```

We can also refer to the second column of the data frame by using a dollar sign and the name of the column:

```
> Exp_data$Quantity
```

```
[1] 25  3 20
```

We discuss getting data into R in the next chapter.

2.4.4 Lists

A list is a collection of heterogeneous objects. We create a list containing some of
the expenditure objects we have created.

```
> Expenditure_list <- list(Price, Quantity, Expenditure,
+       Total_expenditure)
> Expenditure_list

[[1]]
[1] 10   3 15

[[2]]
[1] 25   3 20

[[3]]
[1] 250    9 300

[[4]]
[1] 559
```

The index for a list uses a double bracket. We print the second element below.

```
> Expenditure_list[[2]]

[1] 25   3 20
```

2.5 Example: Net Present Value

We calculate the present value of a sum of money (121) received two years from
now, when the discount rate is 10 %. First, we tell R what the values are:

```
> Amount <- 121
> discount_rate <- 0.1
> time <- 2
```

Then we tell R how to calculate the net present value.

```
> Net_present_value <- Amount/(1 + discount_rate)^time
> Net_present_value

[1]100
```

Another example. We now calculate the net present value of several sums of
money. A cost of 150 is incurred now, and benefits of 135 and 140 are received after
one and two years. The discount rate continues to be 10 %. We use the concatenate
(i.e. `c()`) function.

```
> Cost_benefit_profile <- c(-150, 135, 140)
> time_profile <- c(0, 1, 2)
```

We give R the formula for the profile of discounted costs and benefits.

```
> Cost_benefit_present_value_profile <- Cost_benefit_profile/(1 +
+     discount_rate)^time_profile
```

We sum the values

```
> Net_present_value <- sum(Cost_benefit_present_value_profile)
> Net_present_value
```

```
[1] 88.43
```

We need to be careful while working with vectors, paying attention to their dimensions. Below, we add a vector Three with three elements to a vector Two with one element.

```
> Three <- c(3, 3, 3)
> Two <- 2
> Five <- Three + Two
> Five
```

```
[1] 5 5 5
```

What if we add the vector Three with three elements to a vector Mix with two elements?

```
> Mix <- c(2, 9)
> Mix
```

```
[1] 2 9
```

```
> ThreeandMix <- Three + Mix
```

```
Warning: longer object length is not a multiple of shorter object
length
```

```
> ThreeandMix
```

```
[1]  5 12  5
```

After issuing the warning, R 'recycles' Mix; since the third element is missing it goes back to the first.

2.6 Exploring Further

Torfs and Brauer (2014) have a good short document on R and RStudio. Lander (2014) is an up to date book that will provide a good reference for an economist interested in R and RStudio.

Quick R (Kabacoff 2014) is a useful online reference; it is good to refer to it while working. Datacamp (Cornelissen 2014) has a useful set of online interactive tutorials/courses for R.

This book uses a wonderful package called knitr, which allows us to merge text, R commands and R output seamlessly (Xie 2013).

References

Cornelissen J (2014) Introduction to R. https://www.datacamp.com/courses. Accessed 26 Aug 2014

Kabacoff R (2014) Quick-R. http://www.statmethods.net/index.html. Accessed 26 Aug 2014

Lander JP (2014) R for everyone. Addison-Wesley, New York

R Core Team (2013) R: a language and environment for statistical computing. R foundation for statistical computing, Vienna, Austria. http://www.R-project.org/

Torfs B, Brauer C (2014) A (very) short introduction to R. http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf. Accessed 26 Aug 2014

Xie Y (2013) knitr: a general-purpose package for dynamic report generation in R. http://cran.r-project.org/package=knitr