SPRINGER BRIEFS IN ECONOMICS

Vikram Dayal

An Introduction to R for Quantitative Economics

Graphing, Simulating and Computing

Springer

SpringerBriefs in Economics

More information about this series at http://www.springer.com/series/8876

Vikram Dayal

An Introduction to R for Quantitative Economics

Graphing, Simulating and Computing

Springer

Vikram Dayal
Institute of Economic Growth (IEG)
Delhi
India

ISSN 2191-5504 ISSN 2191-5512 (electronic)
SpringerBriefs in Economics
ISBN 978-81-322-2339-9 ISBN 978-81-322-2340-5 (eBook)
DOI 10.1007/978-81-322-2340-5

Library of Congress Control Number: 2015933817

Springer New Delhi Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer (India) Pvt. Ltd. is part of Springer Science+Business Media (www.springer.com)

For Ma and Papa

Acknowledgments

I thank the Institute of Economic Growth, where I work, for an environment conducive to exploration and discovery. Sitting in its green and peaceful campus, I first learnt about the versatility of R from Suresh, and Debajit gave me a short demonstration. Over several months, Ankila came over regularly to the Institute and we worked on R. Over the last year or so Ranu and I have talked about R, and I used his laptop to do this book. Sekhar commented on some of the chapters. I had received comments from Ankush on a very early version of this book. Varsha has encouraged and advised me. I would like to thank the R, RStudio and mosaic communities. It has been a pleasure to work with Springer.

Vikram Dayal

Contents

About the Author

Vikram Dayal is an Associate Professor at the Institute of Economic Growth, Delhi. He is the author of the book titled *The Environment in Economics and Development: Pluralist Extensions of Core Economic Models,* published in the SpringerBriefs in Economics series in 2014. In 2009 he co-edited the *Oxford Handbook of Environmental Economics in India* with Prof. Kanchan Chopra. He has been incorporating the use of software in teaching quantitative economics—his open access notes on *Simulating to understand mathematics for economics with Excel and R* are downloadable at http://textbookrevolution.org. His research on a range of environmental and developmental issues from outdoor and indoor air pollution in Goa, India to tigers and *Prosopis juliflora* in Ranthambhore National Park has been published in a variety of journals. He visited the Workshop in Political Theory and Policy Analysis in Bloomington, Indiana as a SANDEE (South Asian Network for Development and Environmental Economics) Partha Dasgupta Fellow in 2011. He studied economics in India and the USA and did his doctoral degree from the University of Delhi.

About the Book

This book gives an introduction to R to build up graphing, simulating and computing skills to enable one to see theoretical and statistical models in economics in a unified way. The great advantage of R is that it is free, extremely flexible and extensible. The book addresses the specific needs of economists, and helps them move up the R learning curve. It covers some mathematical topics, such as graphing the Cobb-Douglas function, using R to study the Solow growth model, in addition to statistical topics, from drawing statistical graphs to doing linear and logistic regression. It uses data that can be downloaded from the Internet, and which is also available in different R packages. With some treatment of basic econometrics, the book discusses quantitative economics broadly and simply, looking at models in the light of data. Students of economics or economists keen to learn how to use R would find this book very useful.

Chapter 1
Introduction

Abstract This book emphasizes three key skills—graphing, computing and simulating. We develop these skills in the context of such models as supply and demand, and the Solow growth model, moving between theory and data.

Keywords Graphing · Computing · Simulating

1.1 Three Key Skills

In his book *Macroeconomic Patterns and Stories* the distinguished econometrician Edward Leamer (2010, pp. 6–10) writes:

> Today, advances in medical science come from the joint effort of both theory and empirics, working together. That is what we need when we study how the economy operates: theory and empirical analysis that are mutually reinforcing ... Pictures, Words, and Numbers: In that Order ... We have enormous bandwidth for natural images, and much less for aural information, and hardly any for numbers and symbols.

In this book we use R to develop three key skills so that theory and empirical analysis reinforce each other—graphing, computing and simulating. We work with such economic models as demand and supply, the Cobb-Douglas production function and the Solow growth model, juxtaposing theory and data. Graphing, computing and simulating can help us understand and implement precise but abstract economic models. We learn by doing and develop an intuitive understanding of quantitative economics, to *complement* the formal and mathematical approach of textbooks.

With R, these three skills feed into each other. Most books on R emphasize its use for statistical applications; here we also use R for numerical mathematics. We need a map to traverse the rich world of R. Numerous sources exist that can be used as introductions to R but they are often general, or have computer code that is too complex for a person learning R. In contrast, this book focuses on economics and uses relatively simple code. We build up gradually, going slowly in the initial chapters. We rely a great deal on the mosaic package (Pruim et al. 2014), which while versatile has been designed by its authors keeping teaching in mind. We also use RStudio which greatly eases learning and using R.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_1

We focus on tools that are versatile and can be used in a variety of contexts. To illustrate, for graphs of univariate distributions, we use histograms and boxplots, eschewing quantile-quantile plots that are more precise but less intuitive. We repeatedly use logarithms—while illustrating elasticity, while transforming data and while plotting the long term growth experience of several countries. For mathematical functions, we use the commands `makeFun, plotFun` (from the mosaic package) across chapters. An advantage of `makeFun, plotFun` is that its structure is similar to the lm (linear model) command, used in R for regression.

This book is brief and selective (which should help the reader learn R). However, the focus on the three key skills—graphing, computing and simulating with R—mean that we can tackle a wide range of economic problems using these skills.

Graphing can help us understand and see, especially when a mathematical function is complex and nonlinear, or the data is not appropriately represented by a linear function. For example, the logarithm is a function that is used often in applied economics. We can graph the mathematical function: logarithm of x versus x. Or we can graph a scatterplot of data of one variable against another—this may suggest a logarithmic transformation. We shall see such an example in Chap. 5. In the last chapter we graph time series and see the rich variety of economic data—from seasonal air passenger traffic to volatile stock prices.

When we graph data, we learn from data. Deaton (1997, pp. 3–4) explains his approach to analyzing data:

> Rather than starting with the theory, I more often begin with the data and then try to find elementary procedures for describing them in a way that illuminates some aspect of theory or policy. Rather than use the theory to summarize the data through a set of structural parameters, it is sometimes more useful to present features of the data, often through simple descriptive statistics, or through graphical presentations of densities or regression functions, and then to think about whether these features tell us anything useful about the process whereby they were generated.

Today, computing is easy. We can use the computer for simple calculations or more complex regression. For example, we will compute total expenditures using prices and quantitities in Chap. 2 and use regression in several chapters.

We use simulation in this book in two ways. First, we use Monte Carlo simulation to understand statistical procedures and principles (Chap. 8). Second, we simulate difference equations (Chap. 13). In both cases, simulation greatly aids our understanding. According to Kennedy (2003, p. 24), 'a thorough understanding of Monte Carlo studies guarantees an understanding of the repeated sample and sampling distribution concepts, which are crucial to an understanding of econometrics.' With systems of nonlinear differential or difference equations numerical simulation and geometric investigation may be the only option. According to Strogatz (1994, p. 8), 'most nonlinear systems are impossible to solve analytically. ... Whenever parts of a system interfere, or cooperate, or compete, there are nonlinear interactions going on. Most of everyday life is nonlinear ...'.

Graphing, simulating and computing help us apply economics. They are not used in isolation, but feed into each other. When we see from the graph of data that a transformation of a variable is appropriate, we change the specification of a regression. When we use a simulation to see how the Central Limit Theorem works, we need to graph the results to understand and communicate the simulation.

The range of models that are taught and used in economics is vast. In this book a few key models and functions are used to convey the main ideas. If we develop the three key skills of graphing, simulating and computing, we are equipped to examine other models. Just as once we learn the basic rules of derivatives, we can use those rules on more complicated functions.

The models we consider play a key role in economics. For example, we start with supply and demand. But we not only plot the curves, we also compute equilibria, and confront the issue of identification when we plot data. The mathematics of supply and demand is relatively easy compared to estimating supply and demand from data.

We plot the Cobb-Douglas function in the earlier part of this book from different perspectives. Later in the book, we use the Cobb-Douglas function again when we work with the Solow growth model, and a model of fishing.

In Chap. 11 we journey into an area that plays a key role today in applied economics—evaluating programmes. We focus on one technique (matching), and use statistical graphs to get at the main idea: comparing the treatment group with the control group, which ideally differ only in the treatment.

In this book we constantly overlay theory with data. Economics is taught in separate courses that deal with all the ingredients of economic analysis. But how can we bring these together? Often, researchers learn how to put the ingredients together over several years as they journey towards their Ph. D. But a lot of people who study economics want to apply their skills far sooner—they may work in a non-profit organization, or in a consulting firm. In such a situation, the skills of graphing, computing and simulating are useful.

At the same time, the book is a stepping stone to cutting edge analyses with R; more advanced books, internet sources and relevant R packages are indicated at the end of chapters.

1.2 How to Use the Book

We learn R in the same way we would learn a language. We should start with Chap. 2 to get a feel for R. We can follow up with the "Exploring further" suggestions in Chap. 2. We should follow the book with RStudio open, typing in the R code and running the code. We should experiment with the code, and see what happens. It is a good idea to use Google when we have doubts and to refer to the Quick-R website (Kabacoff 2014).

1.3 Help

We can get help on a function in R by typing help followed by the function enclosed in parentheses; for example,

```
> help(mean)
```

opens a help page on that function in RStudio.

Typing help.start() and running the command will open a page with hyperlinked manuals and package references in RStudio.

1.4 R Code and Output

In this book, what follows the prompt > is R code, in typewriter font. The resulting output is also indicated (without the prompt) in typewriter font. If the R code goes over to the next line, a + appears; the + should not be input in the code in the script.

1.5 An Overview of Typical R Code

We can get lost in R code because there are so many commands and options; so we take a brief tour to get an overview.Typically, R code takes the form:

new object ← function (object or formula , object information , options)

Not all the above elements come into a given line of code; what we have above is a generalization.

A few examples help illustrate more specifically:

- `> Price <- c(21, 31, 34)`

  This makes a vector called price, the c function is concatenate.

- `> z <- makeFun(A * x ~ x, A = 2)`

  The makeFun function in the 'mosaic package' makes a function of x called z using the formula given and the information that A equals 2.

- `> xyplot(y ~ x, data = mydata, type ="p")`

  Here a scatter plot of y against x is generated by the xyplot function in the mosaic or lattice package, using the information that the dataframe for the variables is called mydata. The option exercised is the points option for the function xyplot.

We now consider some of the key R commands by type of objective.

Installing and Loading Packages

Packages extend R's capabilities. We need to install a package once, before we can load it. We use the following code to install the mosaic code:

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

We load the mosaic package when we need to use it:

```
> library(mosaic)
```

Vectors

We can create a vector

```
> Price <- c(2, 3, 4)
```

and get its third element with

```
> Price[3]
```

Data

We can get a data file called myfile into R, and name it myfile:

```
> myfile <- read.csv("myfile.csv")
```

we can access the second column with

```
> second.column <- myfile[, 2]
```

Graphs

We can draw a histogram of variable x with

```
> library(mosaic)  # to load the mosaic package
> histogram(~x, data = mydata)
```

We can draw a scatterplot of y against x with

```
> xyplot(y ~ x, data = mydata)
```

Regression

we can run a linear regression of y on x and z with

```
> reg.mod <- lm(y ~ x + z, data = mydata)
```

To get regression output we use:

```
> summary(reg.mod)
```

1.6 Exploring Further

Kennedy (2003) emphasizes the value of Monte Carlo simulation for understanding econometrics. Mukherjee et al. (1998) show how graphing the data is important. Stevens (2009) uses R for dynamic simulation of mathematical ecology models.

References

Deaton A (1997) The analysis of household surveys. The Johns Hopkins University Press, London
Kabacoff R (2014) Quick-R. http://www.statmethods.net/index.html. Accessed 26 Aug 2014
Kennedy P (2003) A guide to econometrics, 5th edn. MIT Press, Cambridge
Leamer E (2010) Macroeconomic patterns and stories. Springer, Berlin
Mukherjee C, White H, Wuyts M (1998) Econometrics and data analysis for developing countries. Routledge, London
Pruim R, Kaplan D, Horton N (2014) Mosaic: project MOSAIC (mosaic-web.org) statistics and mathematics teaching utilities. R package version 0.9.1-3. http://CRAN.R-project.org/package=mosaic
Stevens MH (2009) A primer of ecology with R. Springer, Dordrecht
Strogatz S (1994) Nonlinear dynamics and chaos. Westview Press, Cambridge

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.

Keywords R · RStudio · Vector · Matrix · Dataframe · List

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.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_2

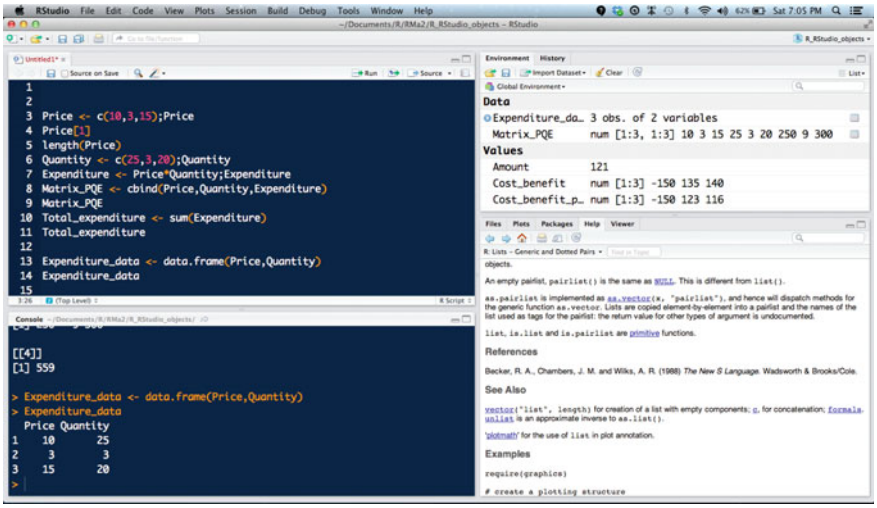

Fig. 2.1 RStudio windows

- 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

Chapter 3
Getting Data into R

Abstract We see how to get data into R with three examples. We like to convert the data into a csv (comma-separated values) file and then get it into RStudio. We see how we can have a quick look at the data in RStudio. We will work with these three datasets in later chapters.

Keywords Data · Carbon and livelihoods · Fish · Oil price

3.1 Introduction

Data is widely available today. Because of the emphasis on transparency by government agencies, and on reproducible research by researchers, data availability will increase. In this chapter, we see how we can get data into R with three examples. We will revisit these datasets in subsequent chapters.

3.2 Chhatre and Agrawal (2009) Data

This data is available at: http://www.ifriresearch.net/resources/data/.

We go to the Referenced Datasets section of the website, and the data corresponding to the Chhatre and Agrawal (2009) paper, "Carbon storage and livelihoods". I had last accessed it on 16 August 2014. It is a zip file, and contains a csv (comma separated variable) file.

After downloading it to the same directory as our current project, we go to the environment window and click on Import Dataset; then from text file, we choose file "ifri car liv", and select header. Alternatively, we can type in the following command, changing the file path as required. We use underscores in our name for the dataset to avoid blank spaces.

```
> ifri_car_liv <- read.csv("~/ifri_car_liv.csv")
```

We look at the top rows of the ifri dataframe using the `head` function.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_3

```
> head(ifri_car_liv)
```

```
  forest_id cid    zliv     zbio    livcar1  ownstate
1       217 NEP -0.6140  -0.4510          3         1
2       325 IND -0.6539  -0.3654          3         1
3        88 UGA -0.3383  -0.9704          3         1
4       174 NEP -0.7855  -1.3252          3         1
5       240 NEP -0.4502  -1.0492          3         1
6       287 TAN -0.1835  -0.8324          3         1
  distance sadmin rulematch lnfsize
1        2      0         0   4.431
2        1      1         0   8.197
3        1      3         0   4.942
4        2     26         0   5.288
5        2      3         1   4.344
6        1     40         1   6.215
```

We choose specific rows and columns, seeing the first five rows and second, fifth and sixth columns:

```
> ifri_car_liv[1:5, c(2, 5, 6)]
```

```
  cid livcar1 ownstate
1 NEP       3        1
2 IND       3        1
3 UGA       3        1
4 NEP       3        1
5 NEP       3        1
```

We now see the first five rows and specific variables cid and ownstate:

```
> ifri_car_liv[1:5, c("cid", "ownstate")]
```

```
  cid ownstate
1 NEP        1
2 IND        1
3 UGA        1
4 NEP        1
5 NEP        1
```

We see the structure of the data using the function *str*.

```
> str(ifri_car_liv)
```

```
'data.frame':       100 obs. of  10 variables:
 $ forest_id: int  217 325 88 174 240 287 324 321 216 82 ...
 $ cid      : Factor w/ 11 levels "","BHU","BOL",..: 9 5 11 9 9 10 5 9 9 11 ...
 $ zliv     : num  -0.614 -0.654 -0.338 -0.786 -0.45 ...
 $ zbio     : num  -0.451 -0.365 -0.97 -1.325 -1.049 ...
 $ livcar1  : int  3 3 3 3 3 3 3 3 3 3 ...
 $ ownstate : int  1 1 1 1 1 1 1 1 1 0 ...
```

```
$ distance : int  2 1 1 2 2 1 2 2 2 1 ...
$ sadmin   : int  0 1 3 26 3 40 8 0 0 0 ...
$ rulematch: int  0 0 0 0 1 1 0 0 1 1 ...
$ lnfsize  : num  4.43 8.2 4.94 5.29 4.34 ...
```

We will use this data in Chap. 10.

3.3 Graddy (2006) Data

This data is available at: http://people.brandeis.edu/~kgraddy/data.htmldata.

We choose the files that say Detailed Fulton Fish market data.

After downloading the file and saving it to our working directory (the same as our current project), we go to the environment window and click on Import Dataset, then from text file, choose fishmayreq, then select header. Alternatively, we can type in the following command, changing the file path as required.

```
> # reading in the file
> fishmayreq <- read.delim("~/Documents/R/RMa3/data_quandl/fishmayreq.txt")
```

We see some rows and columns:

```
> fishmayreq[1:5, c("pric", "quan")]
```

```
  pric quan
1 0.65  120
2 0.85  180
3 1.25  200
4 0.50   60
5 0.65   60
```

We will look at this data in Chap. 4.

3.4 Crude Oil Price Data

This data is available at: http://www.bp.com/en/global/corporate/about-bp/energy-economics/statistical-review-of-world-energy/statistical-review-downloads.html.

This is BP's Statistical Review, we can download the latest Statistical workbook from here. On downloading the workbook file, we select the sheet with oil crude prices since 1861. Then we remove all rows above the header row, and change the current price column header to current and the other to const_ 2013. We then bring it into R Studio.

We go to the environment window and click on Import Dataset; then from text file, choose Oil prices; then select header. Alternatively, you can type in the following command, changing the file path as required.

```
> Oil_prices <- read.csv("~/Oil_prices.csv")
```

We see the top of the data:

```
> head(Oil_prices)
```

```
  Year current const_2013
1 1861    0.49      12.65
2 1862    1.05      24.40
3 1863    3.15      59.36
4 1864    8.06     119.56
5 1865    6.59      99.88
6 1866    3.74      59.26
```

We will look at this data in Chap. 4.

3.5 Exploring Further

The Quick-R (Kabacoff 2014) website is a good place to go to for further information on reading data. The Quick-R website has a good section on missing values.

Coursera (2014) has an online course on getting and cleaning data with R.

References

Chhatre A, Agrawal A (2009) Trade-offs and synergies between carbon storage and livelihood benefits from forest commons. PNAS 106(42):17667–17670

Coursera (2014) Getting and cleaning data. https://www.coursera.org/course/getdata. Accessed 26 Aug 2014

Graddy K (2006) Markets: the Fulton fish market. J Econ Perspect 20(2):207–220

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

Chapter 4
Supply and Demand

Abstract In this chapter we graph mathematical demand and supply functions with the mosaic package. We also compute the equilibrium. Finally, we graph data related to outcomes in the Fulton fish market and the global oil market.

Keywords Supply · Demand · Mosaic package · Fish · Oil price

4.1 Introduction

Economists use supply and demand in a variety of situations. Supply and demand is mathematically simple, and a good starting point for learning to use R for quantitative economics.

4.2 Supply and Demand in General

Let the quantity demanded D be given by:

$D = \alpha_D - \beta_D P$

Similarly for supply S:

$S = \alpha_S + \beta_S P$

At equilibrium, demand = supply, i.e. $D = S$. This gives us the equilibrium price:

$P* = (\alpha_D - \alpha_S)/(\beta_D + \beta_S)$

We will use R to graph demand and supply and compute the equilibrium, using the mosaic package.

4.3 The Mosaic Package

R has a number of packages that extend its capabilities. There are thousands of R packages on the web. By installing, and then loading a specific package we can extend R to tackle our specific problem. We will use the mosaic package (authored

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_4

by Pruim et al. 2013) in this chapter. We first install mosaic. (Installation has to be done once; thereafter we only load the package.)

We can use the Packages menu or the following command:

```
> # install.packages('mosaic')
```

We used the hash (#) sign with the install command above because it was already installed; we should delete the hash and then run the command if mosaic is not installed.

It is a good idea to use hash liberally to put comments in our script.

We now load the package mosaic, with the command *library*.

```
> library(mosaic)
```

If we run the command help(mosaic) after loading it, a description will open in the help viewer in RStudio.

4.4 Demand

We set up an inverse demand function, pD = (125 − 6q) / 8. We have to provide this equation in the specific format required:

```
> pD <- makeFun(( aD - ( bD * q ) ) / cD ~ q,
+                aD = 125, bD = 6,cD = 8)
```

The mosaic package has the *makeFun* function which creates the demand function above. We can run help(makeFun) to learn more about makeFun. D denotes demand, p denotes price, q denotes quantity and a, b, and c are parameters; aD is the parameter related to the demand function. Note the use of the tilde (~); the expression to the left of it is a function of the variable on the right of it. We put in values of the parameters. To see what price in the inverse demand function corresponds to a quantity of 20, we type in the value of q as follows:

```
> pD(20)

[1] 0.625
```

So, when q = 20, p_D = 0.625. We will now plot the inverse demand function (Fig. 4.1):

```
> plotFun(pD, xlim = range(0, 30), ylim = range(5, 20),
+      lty = 2, lwd = 1.5)
```

The command *plotFun* plots the curve, pD above tells it what has to be plotted, xlim stands for the limits of x, ylim similarly. We can get a dotted line by using lty = 2; lty stands for line type. We use lwd to adjust the width of the line.

We make and plot another demand function, denoted by pD2 below; the only difference with pD is that aD is now equal to 150.

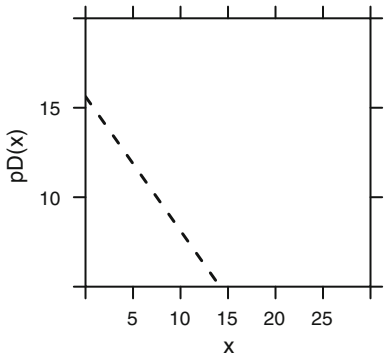

Fig. 4.1 Inverse demand function pD

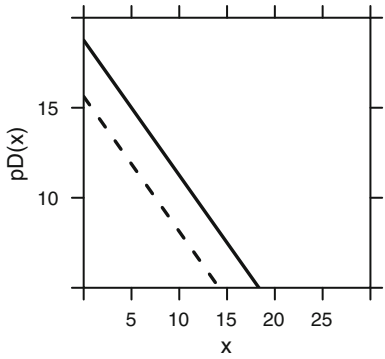

Fig. 4.2 Inverse demand functions pD and pD2

```
> pD2 <- makeFun((aD - (bD * q))/cD ~ q, aD = 150, bD = 6,
+     cD = 8)
```

We can now plot both the demand curves pD and pD2 together (Fig. 4.2).

```
> plotFun(pD, xlim = range(0, 30), ylim = range(5, 20),
+     lty = 2, lwd = 1.7)
> plotFun(pD2, xlim = range(0, 30), ylim = range(5, 20),
+     lwd = 1.7, add = TRUE)
```

Notice the use of the add is equal to TRUE above; this adds the second plot to the first.

4.5 Supply and Demand

Time to look at supply. We now make a supply function with S denoting supply.

```
> pS <- makeFun((aS + (bS * q))/cS ~ q, aS = 12, bS = 2,
+     cS = 5)
```

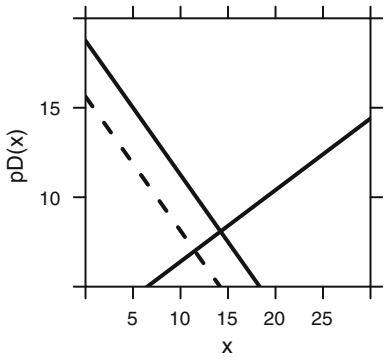

Fig. 4.3 Inverse demand functions and supply (pD, pD2 and pS)

We plot supply with the demand curves.

```
> plotFun(pD, xlim = range(0, 30), ylim = range(5, 20),
+     lty = 2)
> plotFun(pD2, xlim = range(0, 30), ylim = range(5, 20),
+     add = TRUE)
> plotFun(pS, xlim = range(0, 30), ylim = range(5, 20),
+     add = TRUE)
```

A rightward shift in demand increases equilibrium price and equilibrium quantity (Fig. 4.3). Although we can see the equilibrium from the point of intersection of D and S, we may want to calculate the equilibrium price and quantity.

4.6 Equilibrium

At the equilibrium, $D = S$, or $D - S = 0$.

We use the `findZeros` function and apply it to the excess demand:

```
> q.equil <- findZeros(((aD - (bD * q))/cD) - ((aS +
+     (bS * q))/cS) ~ q, aD = 125, bD = 6, cD = 8, aS = 12,
+     bS = 2, cS = 5)
```

We can now get the initial (before the demand curve shifted) equilibrium quantity:

```
> q.equil
```

```
   q
1 12
```

And the equilibrium price:

```
> pD(q.equil)
```

```
      q
1 6.625
```

4.7 Fish Data

Graddy (2006) collected data on price and quantities in the Fulton fish market, a large market in New York. The data is available on her website. We can read it into R (we had done this in Chap. 3):

```
> fish <- read.delim("~/Documents/R/RMa5/supply_demand/fish.out")
```

We then plot price against quantity in a scatter plot (Fig. 4.4 left). To plot price against quantity we use a function or command that is in the mosaic package, `xyplot`. We also plot price against stormy (stormy weather) (Fig. 4.4 right). We 'jitter' stormy, i.e. add some random noise to help distinguish different observations; we ask for type = p for points and = r for regression line.

```
> # Fig 4.4 left
> xyplot(price ~ qty,data=fish)
> # Fig 4.4 right
> xyplot(price ~ jitter(stormy), data = fish, type = c("p","r"))
```

Since both supply and demand were changing, we cannot "identify" a supply or demand curve only from the scatter plot (Fig. 4.4, left). Graddy (2006) found that stormy weather shifted the supply curve; so we can identify the demand curve using that information (Fig. 4.4, right). In other words, stormy weather serves as an 'instrumental variable'. Using only ordinary least squares, the estimated price elasticity of demand for fish was about −0.5; using the instrumental variables estimator, the estimated price elasticity was more than double that. The theoretical framework of demand and supply helps us see beyond the data, and using the econometric

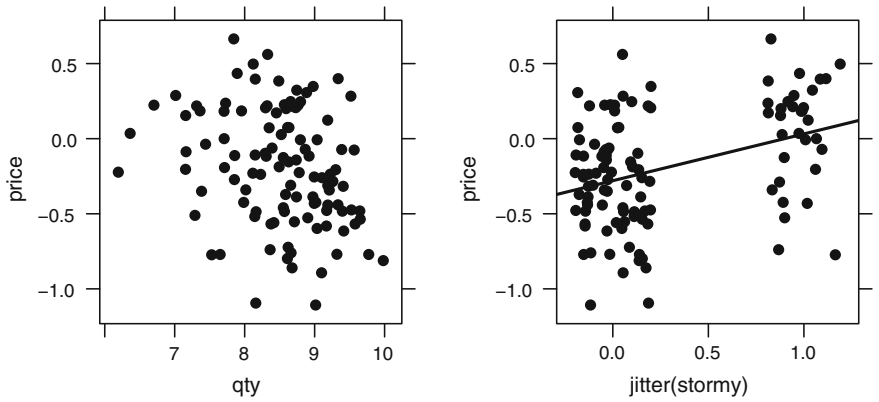

Fig. 4.4 Fulton fish market data, logarithm of equilibrium price versus equilibrium quantity (*left*) and log of equilibrium price conditional on stormy (*right*)

technique of instrumental variables helps us estimate the elasticity. We can also use supply and demand as a framework to think about price movements informally, as in the following case of the price of crude oil. We discuss elasticity in Chap. 5.

4.8 Crude Oil Price Data

We examine some data on the price of crude oil. We input the data (see Chap. 3) and then plot the price versus the year (Fig. 4.5).

```
> # will have to change the filepath
> crude <- read.csv("~/Documents/R/RMa3/supply_demand/Oil_prices.csv")
> xyplot(const_2013 ~ Year, data = crude, type ="l")
```

We use type = l to get a line graph. We can use supply and demand as a framework to help us interpret such a graph. In Fig. 4.5 the price increased after 1970 because of shifts to the left in supply. The increase after 2000 was due to shifts to the right in demand. According to Cowen and Tabarrok (2013, pp. 59–60),

> Supply had been increasing by about 7.5 percent per year in the previous decade, but between 1973 and 1974 production was dead flat. Prices shot up, increasing in real dollars from $14.50 to $46 per barrel in just one year. ... Prices can also fluctuate with shifts in demand. ... The economies of China and India have surged in the early twenty-first century to the point where millions of people are ... able to afford an automobile.

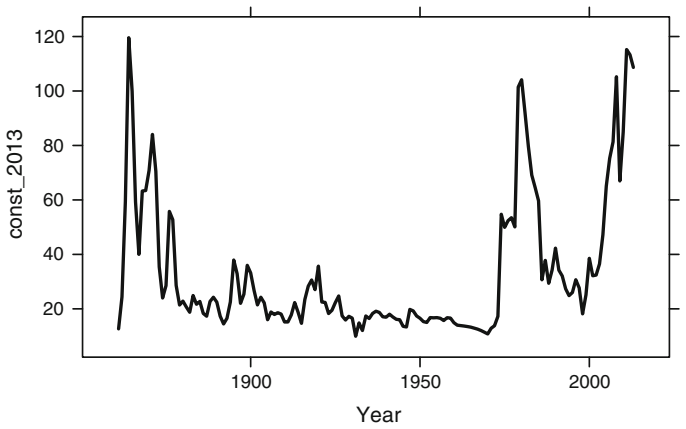

Fig. 4.5 Crude oil price, from 1861 to 2013 (2013 US dollars per barrel)

4.9 Exploring Further

The mosaic authors (Pruim et al. 2013) have prepared very good documents. Graddy's (2006) paper on the Fulton Fish Market in the Journal of Economic Perspectives is very interesting. The two textbooks by Hendry and Nielsen (2007) and Hill et al. (2011) also discuss the case of the Fulton fish market.

BP website (2014) has a wonderful graphic on the price of oil.

References

BP website (2014) http://www.bp.com/en/global/corporate/about-bp/energy-economics/statistical-review-of-world-energy/review-by-energy-type/oil/oil-prices.html.Accessed 28 Aug 2014

Cowen T, Tabarrok A (2013) Modern principles of economics, 2nd edn. Worth publishers, New York

Graddy K (2006) Markets: the Fulton fish market. J Econ Perspect 20(2):207–220

Hendry DF, Nielsen B (2007) Econometric modeling: a likelihood approach. Princeton University Press, Princeton

Hill RC, Griffiths WE, Lim GC (2011) Principles of econometrics, 4th edn. Wiley, New York

Pruim R, Kaplan D, Horton N (2013) Mosaic: project MOSAIC (mosaic-web.org) statistics and mathematics teaching utilities. R package version 0.8-3. http://CRAN.R-project.org/package=mosaic

Chapter 5
Functions

Abstract We briefly look at change, derivative and elasticity formulae. We then graph and compute functions (linear and log-log) using the mosaic package. We gain the ability to derive one mathematical function, often non-linear, from another. We are able to understand such non-linear functions better when we graph them. We see how we can use different functional forms while studying how the average level of carbon dioxide emissions per capita varies with gross national income per capita for different countries.

Keywords Derivative · Elasticity · Logarithm · Mosaic package · Carbon emissions

5.1 Introduction

In econometrics, we use different functional forms to study the relationship between variables. We often compute an elasticity on the basis of the estimated function, which is itself a function. Often, such estimated elasticities inform public debates or policy.

5.2 Change, Derivative and Elasticity

Often, we are interested in how variable y will change when variable x changes.

The change in a variable x is the difference between two values of x. If x is initially x^* and then is x^{**}, the change in x, is

$Change\ in\ x = x^{**} - x^*$
Or $\Delta x = x^{**} - x^*$
Let y be a function of x:
$y = f(x)$.

The rate of change of y with respect to x is the ratio of the change in y to the change in x. The rate of change of a function gives us the slope of the graph of the function.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_5

The derivative of y with respect to x is

$$dy/dx = \frac{lim}{h \to 0}[f(x+h) - f(x)]/h$$

The elasticity of y with respect to x is the ratio of the percent change in y divided by the percent change in x. Elasticity is often denoted by ϵ. $\epsilon = (\Delta y/y)/(\Delta x/x)$ or in terms of the derivative, $\epsilon = (x/y)\, dy/dx$.

5.3 Loading the Mosaic Package

We will work with the mosaic package Pruim et al. (2014) which we had installed (see Chap. 4); so we load it.

```
> library(mosaic)
```

5.4 Linear Function

The linear function (here, $y = f(x) = a + bx$) is straightforward. However, we will see that the elasticity of the linear function, which is itself a function, is not.

We use the following steps.

1. Make the linear function (use makeFun).
2. Plot the linear function (use plotFun).
3. Compute the derivative (a function) (use D).
4. Plot the derivative (use ladd).
5. Compute the elasticity (a function) using makeFun and steps 1 and 3.
6. Plot the elasticity.

We can use this 'recipe' for the linear function by editing the R code to also examine other functions, like the quadratic function. In each step, we use the mosaic package that we had used in Chap. 4.

Step 1. Make the linear function (use makeFun)

```
> y1 <- makeFun(a + b * x ~ x, a = 2, b = 2)
```

We can copy and change the line above to make another function with different values of a and b:

```
> y2 <- makeFun(a + b * x ~ x, a = 20, b = -0.5)
```

Step 2. Plot the linear functions (Fig. 5.1)

We can copy and modify the code that we used in Step 1.

```
> plotFun(y1, xlim = range(0, 30), ylab = "y1, y2")
> plotFun(y2, xlim = range(0, 30), add = TRUE, lty = 2,
+     lwd = 2)
```

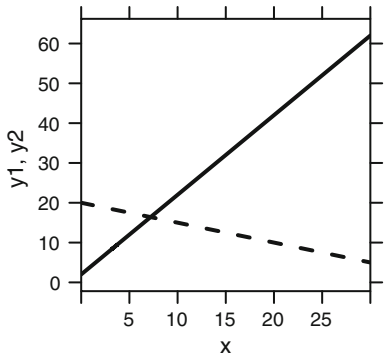

Fig. 5.1 Linear function; y1 and y2 (*dashed line*) versus x

We use a dashed line for y2 using lty (line type). We add the plot of y2 to the plot of y1.

Step 3. Compute the derivative (a function) (use D)

The D function in the mosaic package computes the derivative.

```
> dy1.dx <- D(a + b * x ~ x, a = 2, b = 2)
> dy1.dx

function (x, a = 2, b = 2)
b
```

The output indicates that the derivative of y1 is b, and b here is 2. We repeat for y2.

```
> dy2.dx <- D(a + b * x ~ x, a = 20, b = -0.5)
> dy2.dx

function (x, a = 20, b = -0.5)
b
```

The derivative of b2 is also b; here b is -0.5. Since the derivatives are themselves functions, the code is similar to that used in Step 2.

Step 4. Plot the derivative (Fig. 5.2). We use the command `ladd` and add the horizontal lines (dy1/dx $= 2$ and dy2/dx $= -0.5$).

```
> plotFun(dy1.dx, xlim = range(0, 30), ylim = range(-2,
+     5))
> ladd(panel.abline(a = 2, b = 0, lty = 1, col = "black"))
> # a here corresponds to b in makeFun
> ladd(panel.abline(a = -0.5, b = 0, lty = 2, col = "black"))
```

In Fig. 5.2 we see the derivatives are flat lines.

Step 5. Compute the elasticity (a function) using makeFun and substituting for x, y and dy/dx in the formula $\epsilon = (x/y)dy/dx$ from steps 1 and 3.

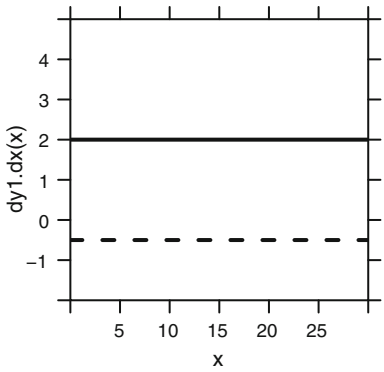

Fig. 5.2 Derivative of y1 and y2 (*dashed line*)

```
> ey1.x <- makeFun(b * x/(a + b * x) ~ x, a = 2, b = 2)
> ey2.x <- makeFun(b * x/(a + b * x) ~ x, a = 20, b = -0.5)
> ey1.x

function (x, a = 2, b = 2)
b * x/(a + b * x)
```

Step 6. Plot the elasticity (Fig. 5.3).

```
> plotFun(ey1.x, xlim = range(0, 30), ylim = range(-2,
+     2))
> plotFun(ey2.x, xlim = range(0, 30), ylim = range(-2,
+     2), add = TRUE, lty = 2, lwd = 2)
```

In Fig. 5.3 we see that the elasticity function corresponding to the linear function can be complex. It is nonlinear and changes in shape with a change in parameters a and b of the linear functions. With y1, when a $= 2$ and b $= 2$, the value of the elasticity of y1 increases rapidly and then flattens out (solid line). With y2, when a $= 20$ and b $= -0.5$, the elasticity of y2 first decreases gradually and then more rapidly (dashed line).

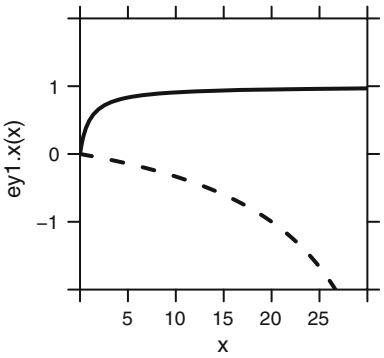

Fig. 5.3 Elasticity of the linear functions

5.5 Log-Log Function

In the case of the log-log function,

$ln(y) = a + b\, ln(x)$
$dy/dx = by/x$
$Elasticity = b$

We will use a property of exponentials and logs, namely that $exp\ (log(y)) = y$ to express y in terms of x while using makeFun.

We can repeat the steps we used for the last section.

Step 1. Make the log-log function (use makeFun)

```
> y1 <- makeFun(exp(a + b * log(x)) ~ x, a = 1.5, b = 1.8)
```

We can modify the code above to make another function with different values of a and b:

```
> y2 <- makeFun(exp(a + b * log(x)) ~ x, a = 1.5, b = -1.1)
```

Step 2. Plot the log-log function (Fig. 5.4).

```
> plotFun(y1, xlim = range(0, 30))
> plotFun(y2, xlim = range(0, 30), ylim = range(0, 5),
+      lty = 2, lwd = 2)
```

Step 3. Compute the derivative (a function) (use D)

The D function in the mosaic package computes the derivative.

```
> dy1.dx <- D(exp(a + b * log(x)) ~ x, a = 1.5, b = 1.8)
> dy1.dx

function (x, a = 1.5, b = 1.8)
exp(a + b * log(x)) * (b * (1/x))
```

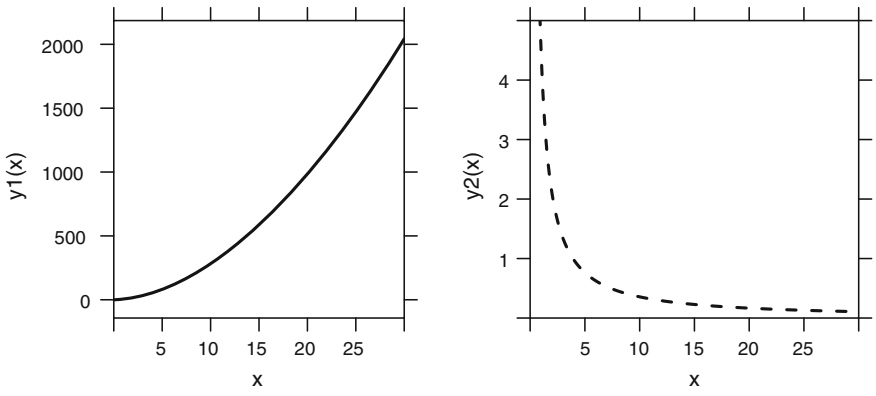

Fig. 5.4 Log-log function (log y $= 1.5 + 1.8$ log x (*left*), log y $= 1.5 - 1.1$ log x (*right*))

```
> dy2.dx <- D(exp(a + b * log(x)) ~ x, a = 1.5, b = -1.1)
> dy2.dx

function (x, a = 1.5, b = -1.1)
exp(a + b * log(x)) * (b * (1/x))
```

Step 4. Plot the derivative (Fig. 5.5).

```
> plotFun(dy1.dx, xlim = range(0, 30), ylim = range(0,
+     120))
> plotFun(dy2.dx, xlim = range(0, 30), ylim = range(1,
+     -6), lty = 2, lwd = 2)
```

Step 5. Compute the elasticity (a function) using makeFun and steps 1 and 3.

```
> ey1.x <- makeFun(b ~ x, b = 1.8)
> ey2.x <- makeFun(b ~ x, b = -1.1)
> ey1.x

function (x, b = 1.8)
b
```

Step 6. Plot the elasticity (Figs. 5.6 and 5.7).

```
> plotFun(ey1.x, xlim = range(0, 30))
> ladd(panel.abline(a = 1.8, b = 0, col = "black"))
> # a here is b in the makeFun

> # a here is b in the makeFun
> plotFun(ey2.x, xlim = range(0, 30), lty = 2, lwd = 2)
> ladd(panel.abline(a = -1.1, b = 0, col = "black", lty = 2))
```

The log-log function has a constant elasticity that can be represented by a horizontal line.

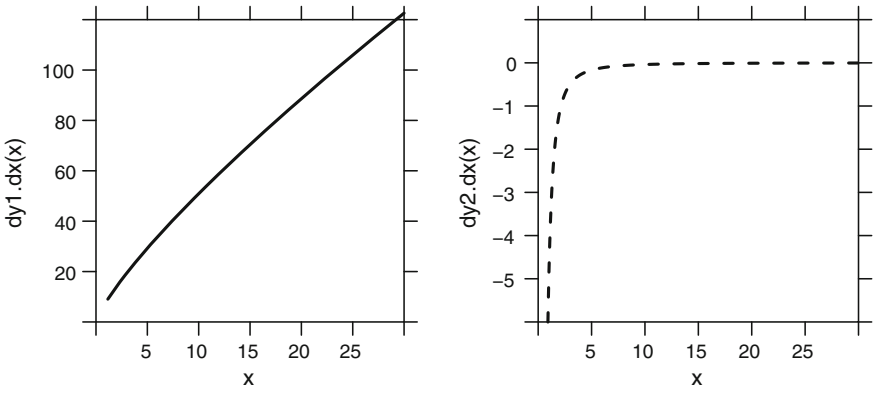

Fig. 5.5 Derivative of the log-log function

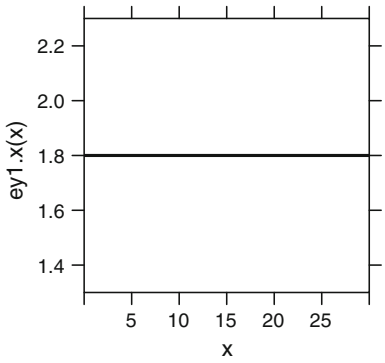

Fig. 5.6 Elasticity of the log-log function

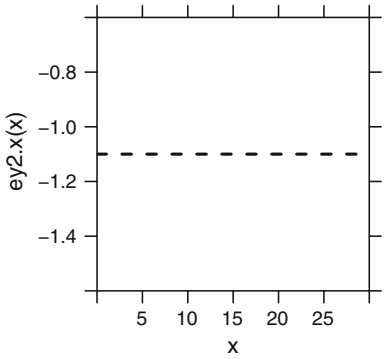

Fig. 5.7 Elasticity of the log-log function

5.6 Functions with Data

We need to choose a functional form when fitting a regression; looking at the data helps.

If we are looking at the bivariate association between carbon dioxide (CO2) emissions and per capita Gross National Income (GNI), we would plot a scatter and then see how the line fits.

We can download this data for the year 2000 from the World Bank (2014) online Databank, World Development Indicators, as a csv file. CO2 emissions are in metric tons per capita, GNI (Gross National Income) per capita are in purchasing power parity constant 2005 international dollars. We read the data into R Studio.

```
> CO2 <- read.csv("~/Documents/R/RMa2/elasticity/CO2pc_gnipc2005ppp.csv")
```

We plot CO2 per capita against GDP per capita (Fig. 5.8):

```
> xyplot(CO2pc ~ GNIpc, data = CO2, ylim = c(0, 30))
```

We fit a linear function and plot it. To fit a linear function, we use the `lm` function, creating an object that we name mod1. The syntax is similar to the `makeFun`

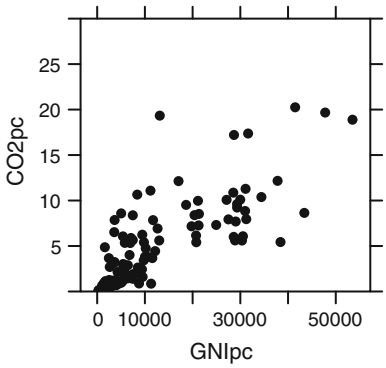

Fig. 5.8 CO2 per capita versus GNI per capita

function. We divide GNIpc by 1000 to get a reasonable size of coefficient, and use I in the model formula.

```
> mod1 <- lm(CO2pc ~ I(GNIpc/1000), data = CO2)
```

We could use the *summary* function to see the regression output but prefer the *display* function in the arm package.

```
> library(arm)
> display(mod1)
```

```
lm(formula = CO2pc ~ I(GNIpc/1000), data = CO2)
              coef.est coef.se
(Intercept)   0.87     0.32
I(GNIpc/1000) 0.32     0.02
---
n = 134, k = 2
residual sd = 2.75, R-Squared = 0.66
```

The fitted regression line is CO2pc $= 0.87 + 0.00032$ GNIpc $+$ error. The coefficient of GNIpc is statistically significant.

We use makeFun on the linear model object mod1 to create CO2mod which we will use for plotting. We first plot the points and then the regression line (Fig. 5.9). We use *makeFun* on mod1, to make CO2mod, and that is used by *plotFun*.

```
> CO2mod <- makeFun(mod1)
> xyplot(CO2pc ~ GNIpc, data = CO2, ylim = c(0, 30))
> plotFun(CO2mod(GNIpc) ~ GNIpc, add = TRUE)
```

We see that there is a cluster of overlapping points in the lower left corner, and a spread as we move to the right. Since the points are so tightly clustered the cluster of points provides less information to guide the fit of the line than if the points in the cluster were spread out.

Plots of residuals alert us to problems in regressions. We plot the residuals against the fitted values, using *xyplot, resid, fitted* respectively. We see the same

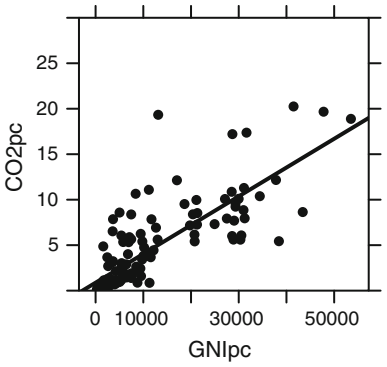

Fig. 5.9 CO2 per capita vs GNI per capita

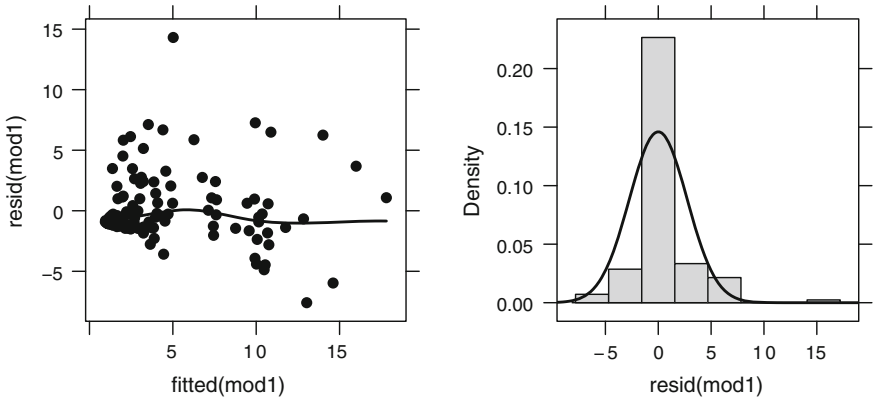

Fig. 5.10 Residuals versus fitted (*left*) and histogram of residuals (*right*)

clustering (Fig. 5.10 left). The histogram of the residuals shows a spike in the centre (Fig. 5.10 right).

```
> xyplot(resid(mod1) ~ fitted(mod1), type = c("p", "smooth"))
> histogram(~resid(mod1), fit = "normal")
```

We could use a log-log function instead of a linear function. When we look at the histograms of the variables we see that both are very positively skewed (Figs. 5.11 and 5.12). When we take the log of the variables, their histograms more closely resemble a normal distribution.

```
> histogram(~CO2pc, data = CO2, fit = "normal")
> histogram(~log(CO2pc), data = CO2, fit = "normal")

> histogram(~GNIpc, data = CO2, fit = "normal")
> histogram(~log(GNIpc), data = CO2, fit = "normal")
```

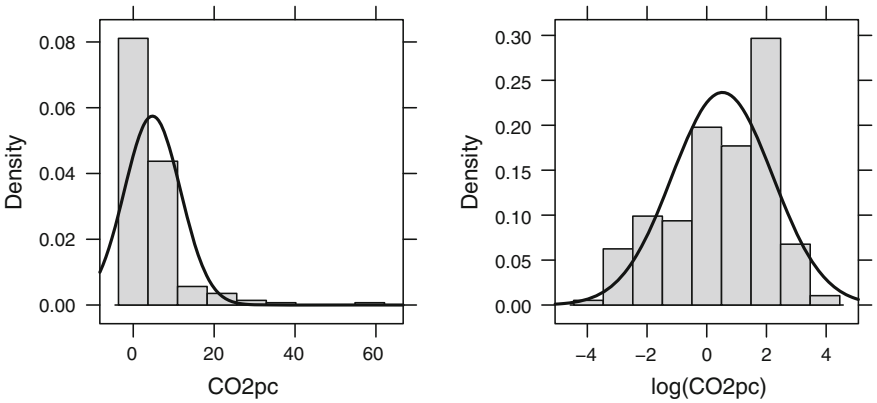

Fig. 5.11 Histogram of CO2 per capita (*left*) and log of CO2 per capita (*right*)

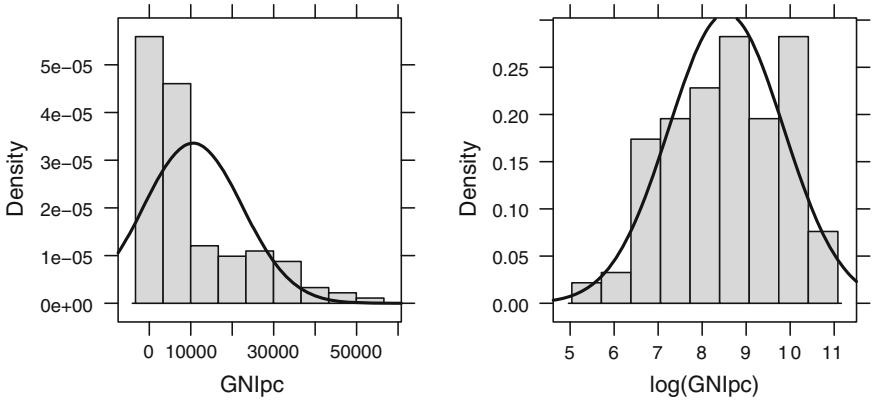

Fig. 5.12 Histogram of Gross National Income (GNI) per capita (*left*) and log of GNI per capita (*right*)

We fit a log-log function:

```
> l.CO2pc <- log(CO2$CO2pc)
> l.GNIpc <- log(CO2$GNIpc)
> mod2 <- lm(l.CO2pc ~ l.GNIpc, data = CO2)
> display(mod2)

lm(formula = l.CO2pc ~ l.GNIpc, data = CO2)
            coef.est coef.se
(Intercept) -9.21     0.41
l.GNIpc      1.14     0.05
---
```

```
n = 134, k = 2
residual sd = 0.72, R-Squared = 0.81
```

The fitted regression line is $\log(\text{CO2pc}) = -9.21 + 1.14\ \log(\text{GNIpc}) + \text{error}$.

We make a scatter plot of log of CO2 per capita versus log of GNI per capita. We choose type p for points and r for regression line. We also make another scatter plot, but choose smooth for a loess nonparametric smoother.

```
> xyplot(l.CO2pc ~ l.GNIpc, data = CO2, type = c("r",
+     "p"))
> xyplot(l.CO2pc ~ l.GNIpc, data = CO2, type = c("p",
+     "smooth"))
```

The scatter and line of fit has a more even distribution of points and the fit is more satisfactory (Fig. 5.13 left). There appears to be a bit of curvature which is captured by the loess smoother (Fig. 5.13 right).

The residuals are now more evenly distributed though there is a hint of curvature in the residuals versus fitted plot (Fig. 5.14).

```
> xyplot(resid(mod2) ~ fitted(mod2), type = c("p", "smooth"))
> histogram(~resid(mod2), fit = "normal")
```

We motivated the use of the log-log function in terms of making the distributions of the predictor and the response more evenly distributed and getting a better fit. The log-log also has a more convenient interpretation. If one country had a gross national income that was 10 % greater than another country, on average its emissions of CO2 per capita were 11.4 % greater.

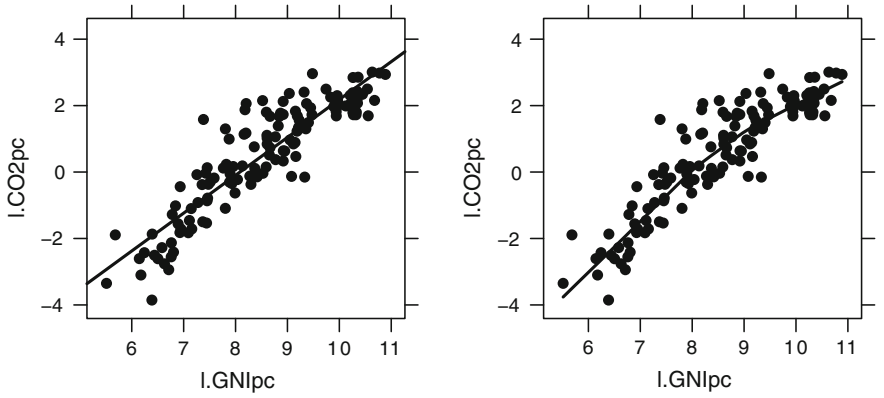

Fig. 5.13 Log of CO2 per capita versus log of GNI per capita

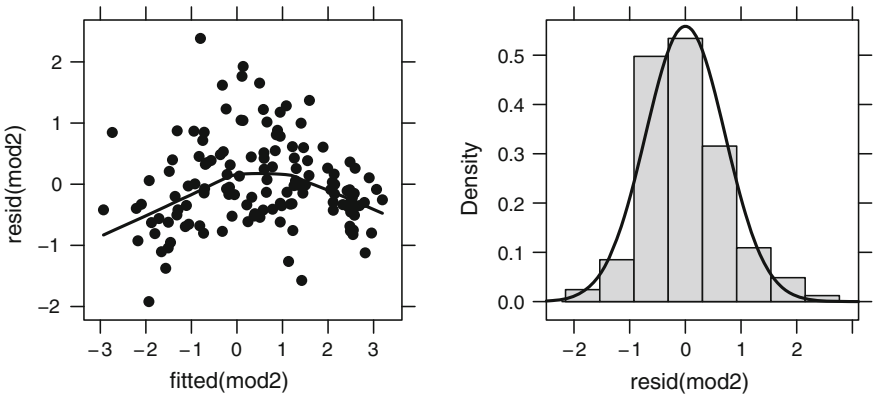

Fig. 5.14 Residuals versus fitted (*left*) and histogram of residuals (*right*) for mod2

5.7 Exploring Further

Hill et al. (2011) have a very clear exposition of different functional forms used in econometrics. Mukherjee et al. (1998) show that plotting histograms and scatterplots and then choosing the functional form can make a vital difference.

References

Hill RC, Griffiths WE, Lim GC (2011) Principles of econometrics, 4th edn. Wiley

Mukherjee C, White H, Wuyts M (1998) Econometrics and data analysis for developing countries. Routledge, London

Pruim R, Kaplan D, Horton N (2014) Mosaic: project MOSAIC (mosaic-web.org) statistics and mathematics teaching utilities. R package version 0.9.1-3. http://CRAN.R-project.org/package=mosaic

World Bank (2014) World development indicators. http://databank.worldbank.org/data/home.aspx. Accessed 5 Feb 2014

Chapter 6
The Cobb-Douglas Function

Abstract We use the mosaic package to view a two input function—the Cobb-Douglas function—from different angles. We see how in the Cobb-Douglas production function output as a function of labour changes as we change the amount of capital or the level of technology. We see how we can graph isoquants.

Keywords Mosaic package · Cobb-Douglas production function

6.1 Introduction

Cobb-Douglas functions are used to model production and utility. They are simple and versatile. The Cobb-Douglas function is used in the Solow model in Chap. 12 and the fishing model in Chap. 13.

According to Hoover (2012, p. 348), "The Cobb-Douglas production function ($Y = AL^a K^{1-a}$) provides a useful representation of aggregate supply." Hoover (2012, p. 331) uses US data to find that aggregate supply for the US economy in 2008 can be represented by $Y = 9.63\, L^{0.67} K^{0.33}$. The units for L are million worker-hours per year, for K are $ billion, and for Y are $ billion.

6.2 Cobb-Douglas Production Function

We can write the Cobb-Douglas Production function as

$$Y = AL^a K^{1-a}$$

where Y is production, A is an index of technology, L is labour and K is capital.

We see immediately that when L and K are zero, Y is zero. We will explore some other properties of this function graphically. We load the mosaic package.

```
> library(mosaic)
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_6

Let A = 5 and a = 0.7. We now plot Y as a function of L taking K to be equal to 20, using *plotfun*.

```
> plotFun(A * (L^0.7) * (K^0.3) ~ L, K = 20, A = 5, ylim = range(-5,
+     101), xlim = range(-1, 21))
```

We see that as we increase L the amount of increase in Y diminishes (Fig. 6.1).

We can now see how the curve relating aggregate production to L changes as we change the amount of K. We plot two curves for Y versus L; one with K = 20 and the other with K = 40.

```
> plotFun(A * (L^0.7) * (K^0.3) ~ L, K = 20, A = 5, ylim = range(-5,
+     151), xlim = range(-1, 21))
> plotFun(A * (L^0.7) * (K^0.3) ~ L, K = 40, A = 5, ylim = range(-5,
+     151), xlim = range(-1, 21), lty = 2, add = TRUE)
```

An increase in K shifts the Y versus L curve up—increasing K helps L become more productive (Fig. 6.2).

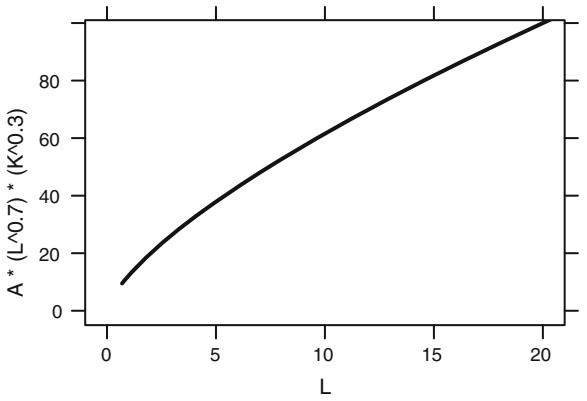

Fig. 6.1 The Cobb-Douglas production function, Y versus L

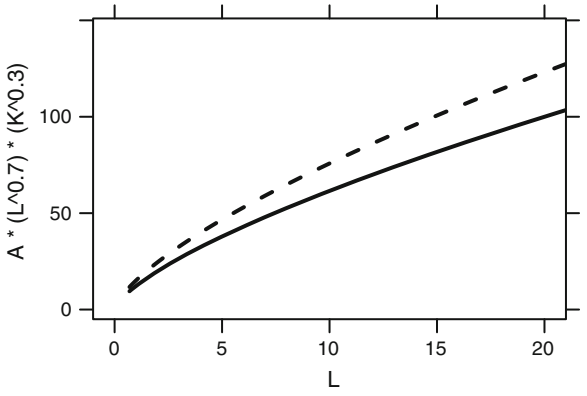

Fig. 6.2 The Cobb-Douglas production function, Y versus L, K = 20 *solid line*, K = 40 *dashed line*

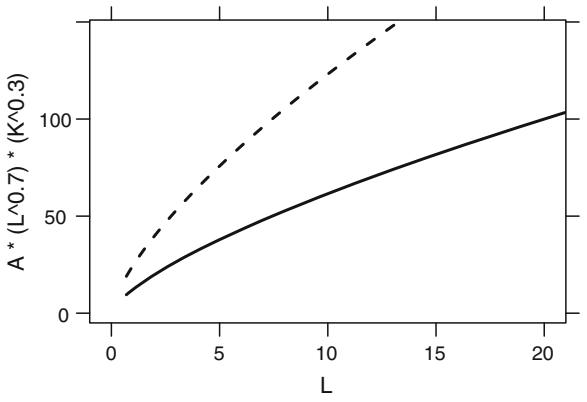

Fig. 6.3 The Cobb-Douglas production function, Y versus L, A = 5 *solid line*, A = 10 *dashed line*

We can see how the curve relating aggregate production to L changes as we change the level of A. As A increases from 5 to 10 the Y versus L curve shifts up (Fig. 6.3). We think of A as the level of technology; an important way we get more output is through increases in A.

```
> plotFun(A * (L^0.7) * (K^0.3) ~ L, K = 20, A = 5, ylim = range(-5,
+     151), xlim = range(-1, 21))
> plotFun(A * (L^0.7) * (K^0.3) ~ L, K = 20, A = 10,
+     ylim = range(-5, 151), xlim = range(-1, 21), lty = 2,
+     add = TRUE)
```

We now plot Y as a function of K taking L to be equal to 20. Again we see that as we increase K, the corresponding increase in Y diminishes (Fig. 6.4).

```
> plotFun(A * (L^0.7) * (K^0.3) ~ K, L = 20, A = 5, ylim = range(-5,
+     101), xlim = range(-1, 21))
```

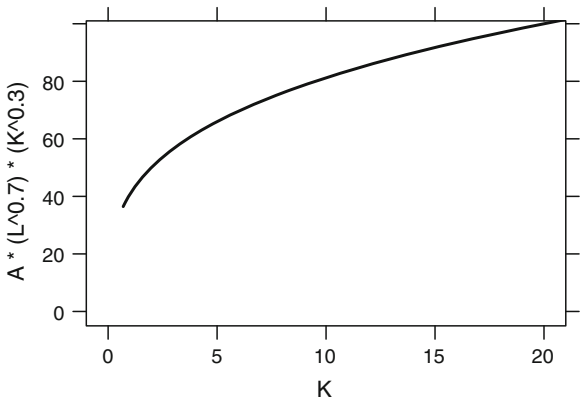

Fig. 6.4 The Cobb-Douglas production function, Y versus K, L = 20

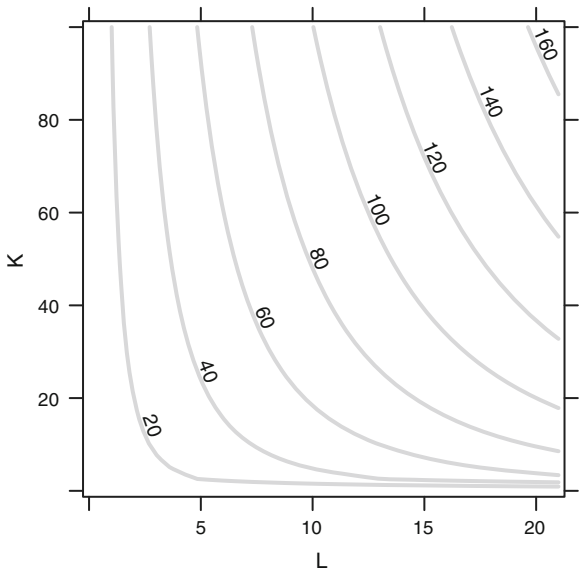

Fig. 6.5 The Cobb-Douglas production function, isoquants

We now plot isoquants— combinations of K and L that give us certain values of Y (Fig. 6.5). When using the plotFun command, we now use L & K to the right of the tilde:

```
> plotFun(A * (L^0.7) * (K^0.3) ~ L & K, A = 5, filled = FALSE,
+     xlim = range(0, 21), ylim = range(0, 100))
```

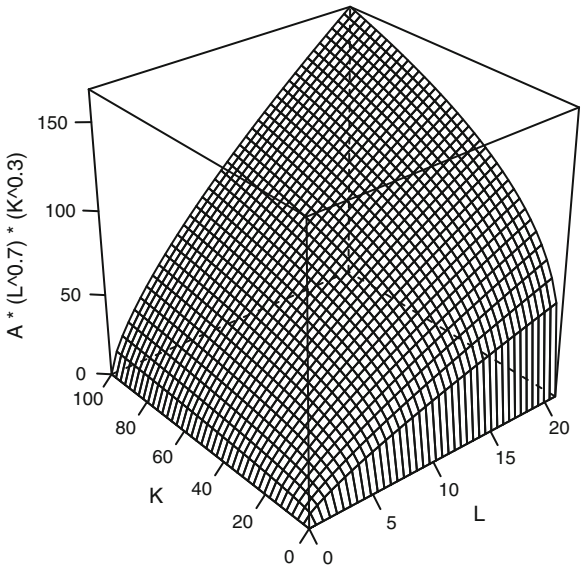

Fig. 6.6 The Cobb-Douglas production function, three-dimensional view

We can use the following code to produce a three-dimensional view (Fig. 6.6). We use the same command that we used for the isoquants, but now indicate that we want a surface plotted (surface = TRUE):

```
> plotFun(A * (L^0.7) * (K^0.3) ~ L & K, A = 5, filled = FALSE,
+     xlim = range(0, 21), ylim = range(0, 100), surface = TRUE)
```

6.3 Exploring Further

Hoover (2012) and Varian (2003) provide good expositions of Cobb-Douglas functions. Hoover (2012) has a very useful chapter titled, "A Guide to Working with Economic Data."

References

Hoover KD (2012) Applied intermediate macroeconomics. Cambridge University Press, New York

Varian HR (2003) Intermediate microeconomics: a modern approach, 6th edn. W. W. Norton and Company, New York

Chapter 7
Matrices

Abstract We combine matrix algebra computations with those of statistics. We first use R for some simple vector operations relating to variances and covariances. Then, we look at some simple matrix operations. Finally, we use matrix operations in R for regression.

Keywords Matrices · Statistics · Regression

7.1 Introduction

We use matrices to store data and matrix manipulations underlie econometric estimation. They help us generalize formulae to many variables.

7.2 Simple Statistics with Matrices

We start with a vector x, which has the following elements:

```
> x <- c(1, 1.25, 2, 2.5, 3)
```

The mean of x, $\overline{x}$ is:

```
> mean(x)

[1] 1.95
```

The formula for the variance of x is $var_x = [1/(n-1)] \sum_{i=1}^{n} (x_i - \overline{x})^2$

We can first calculate a vector of the deviation of each value of x from the mean of x, square the deviations, add them up, and then divide by n − 1.

We can get the value of n by using the command `length`:

```
> length(x)

[1] 5
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_7

The deviations of values of x from their mean is given by:

```
> dev.x <- x - mean(x)
> dev.x

[1] -0.95 -0.70  0.05  0.55  1.05
```

To square these values and add the squared values we multiply the transpose of dev.x by dev.x, using not star but % * %, the symbol for matrix multiplication in R.

```
> t(dev.x) %*% dev.x

     [,1]
[1,]  2.8
```

We now calculate var_x, the variance of x:

```
> Var_calc_x <- t(dev.x) %*% dev.x/(length(x) - 1)
> Var_calc_x

     [,1]
[1,]  0.7
```

We can check that the variance of x has been calculated correctly by using the R function for variance, `var`, directly:

```
> var(x)

[1] 0.7
```

We input a vector y now

```
> y <- c(1.5, 2.5, 3.4, 3.5, 4)
> var(y)

[1] 0.977
```

The covariance of y and x is given by $cov_{xy} = [1/(n-1)] \sum_{i=1}^{n} (y_i - \overline{y})(x_i - \overline{x})$

It is easy to modify the code that we used to calculate the variance of x above to calculate the covariance of y and x:

```
> mean(y)

[1] 2.98

> length(y)

[1] 5

> dev.y <- y - mean(y)
> dev.y

[1] -1.48 -0.48  0.42  0.52  1.02
```

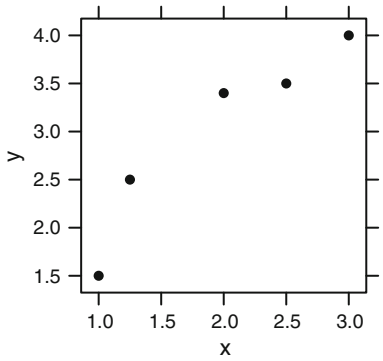

Fig. 7.1 Scatter plot of y against x

```
> covar_calc_yx <- t(dev.y) %*% dev.x/(length(y) - 1)
> covar_calc_yx
```

```
     [,1]
[1,] 0.78
```

We now calculate the covariance of y and x using the R function for covariance directly:

```
> cov(y, x)
```

```
[1] 0.78
```

We plot y and x:

```
> library(mosaic)
```

```
> xyplot(y ~ x)
```

Figure 7.1 shows that x and y are positively correlated. We use the *cor* function and then calculate the correlation:

```
> cor(y, x)
```

```
[1] 0.9432
```

7.3 Simple Matrix Operations with R

A matrix is an array of numbers with rows and columns. We use the matrix command to make a matrix. We indicate the number of columns with ncol:

```
> A <- matrix(c(2, 3, 3, 4), ncol = 2)
> A
```

```
     [,1] [,2]
[1,]    2    3
[2,]    3    4
```

```
> B <- matrix(c(2, 3, 4, 5), ncol = 2)
> B
```

```
     [,1] [,2]
[1,]    2    4
[2,]    3    5
```

The number of columns was two in A and B.
We can add the matrices A and B.

```
> M <- B + A
> M
```

```
     [,1] [,2]
[1,]    4    7
[2,]    6    9
```

The transpose of a matrix simply switches the rows and columns:

```
> t(A)
```

```
     [,1] [,2]
[1,]    2    3
[2,]    3    4
```

```
> t(B)
```

```
     [,1] [,2]
[1,]    2    3
[2,]    4    5
```

We can use matrices to represent a set of equations compactly; let us say $2w+3z = 7$ and $3w + 4z = 10$. We can represent this with matrices as A D = C. Here D is a column vector with elements w and z and C is also a column vector with elements 7 and 10. Since $AD = C$, $D = A^{-1}C$. We multiply matrices with the symbol percent star percent, and invert by using 'solve' in R.

```
> C = c(7, 10)
> D <- solve(A) %*% C
> D
```

```
     [,1]
[1,]    2
[2,]    1
```

We see that $w = 2$ and $z = 1$.

7.4 Regression

We make a data frame called Dat as a different object but with the same values as y and x from Sect. 7.2. In the data frame Dat they are called Response and Predictor (Response and Predictor variables).

```
> Cons = c(1, 1, 1, 1, 1)
> Dat <- data.frame(Response = y, Cons = Cons, Predictor = x)
> X <- cbind(Cons, x)
> Dat
```

```
  Response Cons Predictor
1      1.5    1      1.00
2      2.5    1      1.25
3      3.4    1      2.00
4      3.5    1      2.50
5      4.0    1      3.00
```

We regress Response on Predictor:

```
> fit <- lm(Response ~ Predictor, data = Dat)
> coef(fit)
```

```
(Intercept)   Predictor
     0.8071      1.1143
```

We can use the following formulae for the coefficients of a simple bivariate regression:

```
> b1 = cov(y, x)/var(x)
> b1
```

```
[1] 1.114
```

```
> b0 = mean(y) - (b1 * mean(x))
> b0
```

```
[1] 0.8071
```

We can also use the matrix formula for least squares; $B = (X^T X)^{-1} X^T y$

```
> matcoeff <- solve(t(X) %*% X) %*% t(X) %*% y
> matcoeff
```

```
       [,1]
Cons 0.8071
x    1.1143
```

We plot the scatter and line of fit below (Fig. 7.2).

```
> fitM <- makeFun(fit)
> xyplot(Response ~ Predictor, data = Dat)
> plotFun(fitM, add = TRUE)
```

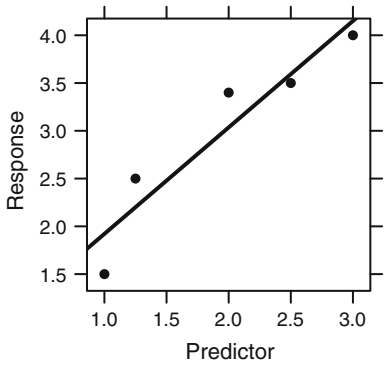

Fig. 7.2 Scatter plot and line of fit

7.5 Exploring Further

Maddala and Lahiri (2009) patiently discuss matrix algebra and its use in econometrics. Hojsgaard (2005) has a comprehensive document on Linear Algebra in R available online.

References

Hojsgaard S (2005) Linear algebra in R. http://modelosysistemas.azc.uam.mx/texts/ProgramaR/LinearAlgebra-inR.pdf. Accessed 30 Aug 2014

Maddala GG, Lahiri K (2009) Introduction to econometrics. Wiley, New Delhi

Chapter 8
Statistical Simulation

Abstract We use R to generate synthetic data from different probability distributions. We then generate data and use simple regression and a t-test on this data. Finally, we simulate logit regression.

Keywords Simulation · Probability · Central limit theorem · Logit regression

8.1 Introduction

One of R's strengths is the ease with which we can carry out statistical simulation. It can help us understand statistics and econometrics.

8.2 Probability Distributions

We can easily generate synthetic data from probability distributions.

8.2.1 Normal Distribution

We load the mosaic package which we will use for plotting graphs.

```
> library(mosaic)
```

We start with the normal distribution. We need to indicate the sample size (n), the mean (mu), and the standard deviation (sd).

```
> n <- 1000
> mu <- 800
> sd <- 35
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_8

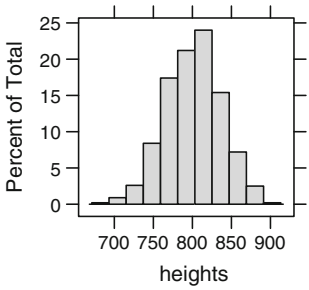

Fig. 8.1 Histogram of heights (normal distribution)

We generate our variable called heights with the R function *rnorm* (r for random, norm for normal).

```
> heights <- rnorm(n = n, mean = mu, sd = sd)
```

We print the first ten heights.

```
> heights[1:10]
```

```
 [1] 793.1 774.3 767.3 813.3 794.1 829.4 827.4
 [8] 777.6 772.8 828.8
```

We plot the histogram of heights (Fig. 8.1).

```
> histogram(~heights, type = "percent")
```

8.2.2 Uniform Distribution

We move to a uniform distribution with sample size n equal to 1000, lower limit a equal to zero and upper limit b equal to 100.

```
> n <- 1000
> a <- 0
> b <- 100
```

We generate our variable called measures from the uniform distribution with the R command *runif*.

```
> measures <- runif(n = n, min = a, max = b)
> measures[1:10]
```

```
 [1] 44.6352  0.9314 18.7787 73.8897 19.3040
 [6] 24.7675 89.7296 47.1936 36.0293 17.7515
```

We plot the histogram of measures (Fig. 8.2).

```
> histogram(~measures, type = "percent")
```

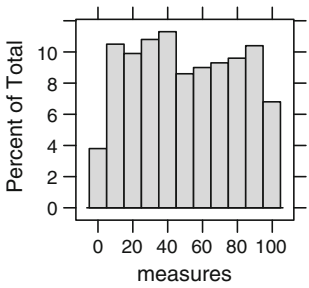

Fig. 8.2 Histogram of measures (uniform distribution)

8.2.3 Binomial Distribution

We generate a variable called females from the binomial distribution. If we have classrooms with 30 students (N), and the probability of any single student being female (p) is half, what is the distribution of females per class that we observe in a thousand classrooms?

```
> n <- 1000
> N <- 30
> p <- 0.5
> females <- rbinom(n = n, size = N, prob = p)
> females[1:10]
```

```
 [1] 14 18 11 13 18 14 16 17 17 15
```

We plot the histogram of females (Fig. 8.3).

```
> histogram(~females, type = "percent")
```

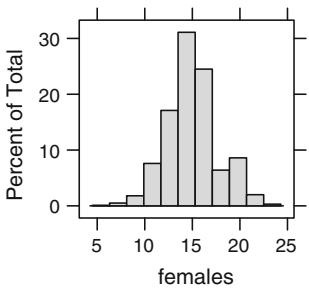

Fig. 8.3 Histogram of females

8.3 Central Limit Theorem

The Central Limit Theorem gives us this remarkable result: the distribution of an average tends to be Normal, even when the distribution from which the average is computed is quite different from a Normal distribution.

Let y be our population; y consists of a thousand values from a uniform distribution with a minimum value of 0 and a maximum value of 100.

```
> y <- runif(n = 1000, min = 0, max = 100)
> histogram(~y, type = "percent", breaks = 10)
```

The histogram of y (Fig. 8.4) looks like that of measures (compare with Fig. 8.2). We first see what happens when we take samples of size 2 each.

```
> sample(y, size = 2)

[1] 69.91 36.06

> sample(y, size = 2)

[1] 61.16 36.83
```

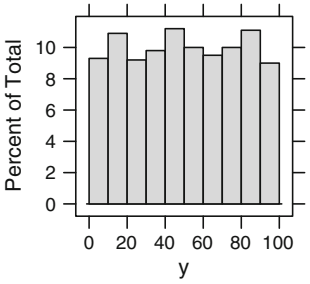

Fig. 8.4 Histogram of y (population, uniform distribution)

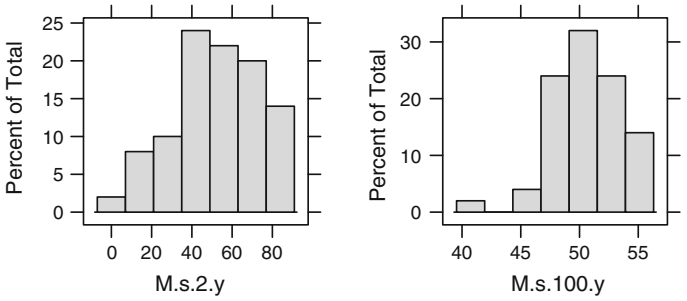

Fig. 8.5 Histogram of 50 samples of y each of size 2 (*left*), and histogram of 50 samples of y each of size 100 (*right*)

We now take samples of size 2, calculate the mean and repeat this 50 times with the *do* function in the mosaic package. We then plot the histogram of the sample means (Fig. 8.5 left).

We do the same with samples of size 100 (Fig. 8.5 right).

```
> M.s.2.y = do(50) * mean(sample(y, 2))
> histogram(~M.s.2.y, type = "percent")
> M.s.100.y = do(50) * mean(sample(y, 100))
> histogram(~M.s.100.y, type = "percent")
```

8.4 The t-Test

Assume that we have wages of 48 women (n.w) and 52 men (n.m) and want to see how they differ. The average wage of women (mu.w) is 100, and of men (mu.m) is 90. We assume that the wages of men and women have the same standard deviation (sigma) equal to 2.

We first give R the parameters:

```
> n.w <- 48
> n.m <- 52
> mu.w <- 100
> mu.m <- 90
> sigma <- 2
> n <- n.w + n.m
```

We then generate the data:

```
> y.w <- rnorm(n.w, mu.w, sigma)
> y.m <- rnorm(n.m, mu.m, sigma)
```

We put the data on men and women together:

```
> wages <- c(y.w, y.m)
```

We create a gender indicator variable:

```
> gender <- rep(c(0, 1), c(n.w, n.m))
```

We use boxplots to see the variation in wages by gender (Fig. 8.6):

```
> bwplot(factor(gender) ~ wages)
```

Fig. 8.6 Boxplot of wages by gender

and run a regression:

```
> fit.wages <- lm(wages ~ gender)
> library(arm)
> display(fit.wages)
```

```
lm(formula = wages ~ gender)
            coef.est coef.se
(Intercept)  99.73   0.28
gender      -10.20   0.39
---
n = 100, k = 2
residual sd = 1.93, R-Squared = 0.88
```

We estimate a difference which is statistically significant. Because we generated the data, we known the actual value is 10. When we analyze real world data, we only have the data and attempt to discover an approximation to the data generating process.

8.5 Logit Regression

If our dependent variable is qualitative we can use logit. Here we consider purchase (yes or no), as a function of income. When the latent variable, desire to purchase, which is a function of income, crosses a threshold, we purchase.

We generate the income data and the latent variable:

```
> income <- 1:100
> late.dd <- 1 + 0.02 * income
> late.star <- late.dd + rnorm(n = 100, mean = 1, sd = 0.5)
```

We make a scatter plot of late.star versus late.dd (Fig. 8.7):

```
> xyplot(late.star ~ late.dd)
```

We then generate the purchase data; if late.star exceeds 2.5, we purchase. If not, we don't.

```
> purchase <- ifelse(late.star > 2.5, 1, 0)
> purchase
```

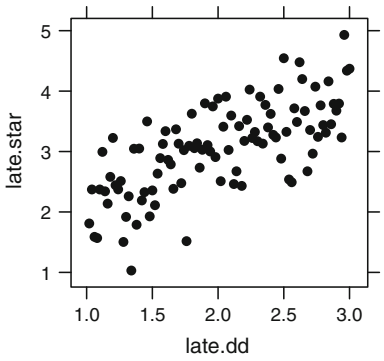

Fig. 8.7 Scatter plot of late.star versus late.dd

```
  [1] 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0
 [23] 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1
 [45] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1
 [67] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 [89] 1 1 1 1 1 1 1 1 1 1 1 1
```

We make a scatter plot of purchase versus income (Fig. 8.8). We might observe data on whether a consumer made a purchase and the consumer's income, but we don't observe the latent variable.

```
> xyplot(jitter(purchase) ~ income)
```

We run a logit regression, with the R function glm (generalized linear model), setting the argument family to binomial:

```
> mylogit <- glm(purchase ~ income, family = "binomial")
> display(mylogit)

glm(formula = purchase ~ income, family = "binomial")
            coef.est coef.se
(Intercept) -1.59     0.55
 income       0.07     0.02
```

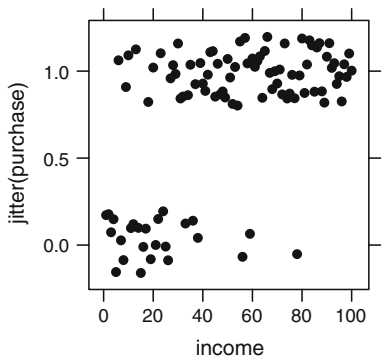

Fig. 8.8 Scatter plot of purchase versus income, purchase jittered

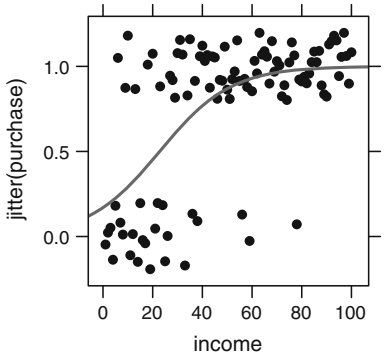

Fig. 8.9 Plot of logistic fit of purchase to income

```
---
  n = 100, k = 2
  residual deviance = 73.4, null deviance = 112.5 (difference = 39.0)

> mylo <- makeFun(mylogit)
```

Then we plot the fitted curve (Fig. 8.9).

```
> xyplot(jitter(purchase) ~ income)
> plotFun(mylo, add = TRUE)
```

8.6 Exploring Further

Gelman and Hill (2007) explain various uses of simulation and provide R code. Cowpertwait and Metcalfe (2009) use simulation while introducing time series with R.

References

Cowpertwait PSP, Metcalfe AV (2009) Introductory time series with R. Springer, Dordrecht
Gelman A, Hill J (2007) Data analysis using regression and multilevel/hierarchichal models. Cambridge University Press, New York

Chapter 9
Anscombe's Quartet: Graphs Can Reveal

Abstract We look at Anscombe's, (Am Stat 27(1):17–21, 1973) data which is one of the datasets that come with the R software. We compute different linear regressions of Anscombe's four sets of data—they give us the same results. When we look at the scatterplots corresponding to Anscombe's four sets of data we see that they are very different.

Keywords Anscombe · Graphs

9.1 Introduction

In 1973, the statistician Anscombe published a lovely paper in which he argued that we should use statistical graphs (p.17):

> A computer should make *both* calculations *and* graphs. Both sorts of output should be studied; each will contribute to understanding. ... Most kinds of statistical calculation rest on assumptions about the behavior of the data. Those assumptions may be false, and then the calculations may be misleading. We ought always to try to check whether the assumptions are reasonably correct; and if they are wrong we ought to be able to perceive in what ways they are wrong. Graphs are very valuable for these purposes.

Anscombe (1973) provided some synthetic data to illustrate this. Anscombe's data illustrates the importance of visualization, of examining the data graphically.

9.2 The Data: 4 Sets of xs and ys

R comes with some datasets; one of these is anscombe. We call the dataset ans and see what it contains:

```
> ans <- anscombe
> ans
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_9

```
   x1  x2  x3  x4     y1    y2    y3    y4
1  10  10  10   8   8.04  9.14  7.46  6.58
2   8   8   8   8   6.95  8.14  6.77  5.76
3  13  13  13   8   7.58  8.74 12.74  7.71
4   9   9   9   8   8.81  8.77  7.11  8.84
5  11  11  11   8   8.33  9.26  7.81  8.47
6  14  14  14   8   9.96  8.10  8.84  7.04
7   6   6   6   8   7.24  6.13  6.08  5.25
8   4   4   4  19   4.26  3.10  5.39 12.50
9  12  12  12   8  10.84  9.13  8.15  5.56
10  7   7   7   8   4.82  7.26  6.42  7.91
11  5   5   5   8   5.68  4.74  5.73  6.89
```

The xs have means $= 9$ and standard deviations $= 3.317$ as can be verified by using the functions *mean* and *sd*. The ys have means $= 7.5$ and standard deviations $= 2.03$.

9.3 Same Regressions of ys on xs

When we regress y1 on x1 and y2 on x2 etc. using *lm*, and see the output with *display* the coefficients and other regression output are the same:

```
> fit1 <- lm(y1 ~ x1, data = ans)
> fit2 <- lm(y2 ~ x2, data = ans)
> fit3 <- lm(y3 ~ x3, data = ans)
> fit4 <- lm(y4 ~ x4, data = ans)
> library(arm)
> display(fit1)

lm(formula = y1 ~ x1, data = ans)
            coef.est coef.se
(Intercept) 3.00     1.12
x1          0.50     0.12
---
n = 11, k = 2
residual sd = 1.24, R-Squared = 0.67

> display(fit2)

lm(formula = y2 ~ x2, data = ans)
            coef.est coef.se
(Intercept) 3.00     1.13
x2          0.50     0.12
---
n = 11, k = 2
residual sd = 1.24, R-Squared = 0.67
```

```
> display(fit3)

lm(formula = y3 ~ x3, data = ans)
            coef.est coef.se
(Intercept) 3.00     1.12
x3          0.50     0.12
---
n = 11, k = 2
residual sd = 1.24, R-Squared = 0.67

> display(fit4)

lm(formula = y4 ~ x4, data = ans)
            coef.est coef.se
(Intercept) 3.00     1.12
x4          0.50     0.12
---
n = 11, k = 2
residual sd = 1.24, R-Squared = 0.67
```

9.4 Very Different Scatter Plots

We load the mosaic package.

```
> library(mosaic)
```

We use xyplot to plot scatters for the 4 sets of ys and xs and choose points (p) and regression lines (r):

```
> xyplot(y1 ~ x1, data = ans, type = c("p","r"))
> xyplot(y2 ~ x2, data = ans, type = c("p","r"))
> xyplot(y3 ~ x3, data = ans, type = c("p","r"))
> xyplot(y4 ~ x4, data = ans, type = c("p","r"))
```

When we examine the lines of fit they are the same, but the scatter plots are very different (Fig. 9.1). In Fig. 9.1, a quadratic term should be used for y2 versus x2 because there is curvature. In the scatter of y3 against x3, one outlying point is tilting the fitted line upwards. In the scatter of y4 against x4, if we drop the extreme point, there is no relationship between y4 and x4.

We choose (p) and loess smoother lines (smooth) to get Fig. 9.2.

```
> xyplot(y1 ~ x1, data = ans, type = c("p","smooth"))
> xyplot(y2 ~ x2, data = ans, type = c("p","smooth"))
> xyplot(y3 ~ x3, data = ans, type = c("p","smooth"))
> xyplot(y4 ~ x4, data = ans, type = c("p","smooth"))
```

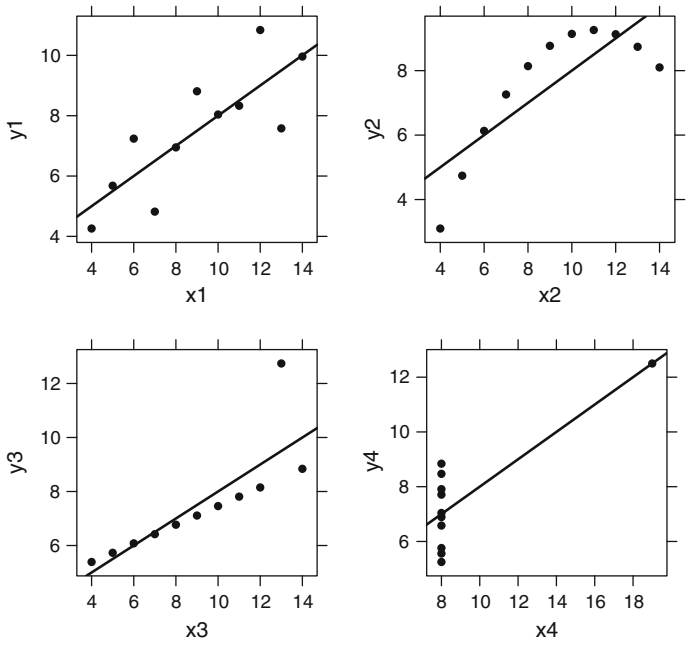

Fig. 9.1 Scatterplots and linear regressions for the four datasets in Anscombe (1973)

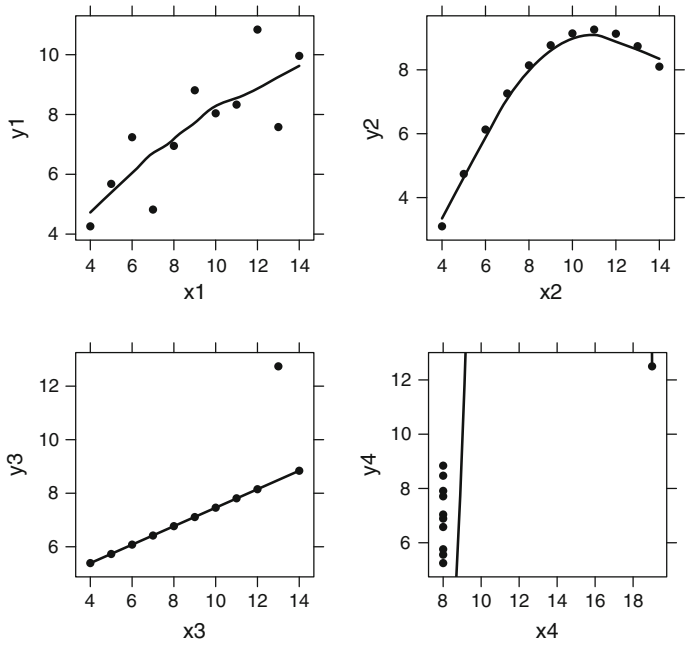

Fig. 9.2 Scatterplots and loess smoothers for the four datasets in Anscombe (1973)

What is also notable is that the smoother lines in Fig. 9.2 do not deceive us the way linear regression did.

9.5 Exploring Further

It is really worth reading Anscombe (1973) original paper.

Reference

Anscombe FJ (1973) Graphs in statistical analysis. Am Stat 27(1):17–21

Chapter 10
Carbon and Forests: Graphs and Regression

Abstract In this chapter we analyze data related to carbon storage and forest livelihoods. We plot histograms and conditional scatterplots. We then fit a multiple regression model with interactions, and use graphs to help us understand the model.

Keywords Carbon storage · Forest livelihoods · Interaction · Multiple regression

10.1 Introduction

Forests help store carbon, and, in developing countries, are a source of livelihoods for rural people. We will examine data on carbon storage and forest livelihoods used by Chhatre and Agrawal (2009). This dataset was put together by the International Forestry Resources and Institutions (IFRI) network.

10.2 Graphs

We first install and then load the mosaic package (Pruim et al. 2014).

```
> library(mosaic)
```

We then load the dataset that can be downloaded from the web as we discussed in an earlier chapter. It is in a file with a comma separated variable (csv) format.

```
> ifri <- read.csv("~/Documents/R/RMa2/chhatre/ifri_car_liv.csv")
```

We rename the two key variables to make the results easier to understand; zbio is renamed carbon, and zliv is renamed liveli:

```
> ifri$carbon <- ifri$zbio
> ifri$liveli <- ifri$zliv
```

On examining the data object in R Studio, we found that there were some rows of missing values that had crept in beyond the 80th observation (there are 80

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_10

observations in the original dataset). We can easily subset the data by selecting the first 80 rows:

```
> # for some reason some na rows come in, so need to
> # select and choose rows without na i.e. missing
> # data
> ifri <- ifri[1:80, ]
```

We create 'factors' from the ownership and rulematch variables, so that the graphs are more informative.

```
> # viewing the top 6 rows
> ifri$f_own <- factor(ifri$ownstate, labels = c("Community",
+     "State"))
> ifri$f_rule <- factor(ifri$rulematch, labels = c("Low",
+     "High"))
```

The variable liveli is an index of forest livelihoods (based on fractions of subsistence needs like firewood, fodder etc. met from the forest). We examine its histogram (Fig. 10.1):

```
> histogram(~liveli, data = ifri, type = "percent")
```

The variable carbon represents carbon storage (standardized), and is based on basal area of trees per hectare. We examine its histogram (Fig. 10.2):

```
> histogram(~carbon, data = ifri, type = "percent")
```

There is substantial variation in carbon storage and livelihoods index in the sample. We look at the scatterplot of carbon and liveli (Fig. 10.3):

```
> xyplot(carbon ~ liveli, data = ifri, type = c("p",
+     "smooth", "r"))
```

There is a very low correlation between liveli and carbon (Fig. 10.3); this is a key observation. Some forests have high levels of both carbon and liveli, some have low levels of both, and some have low or high levels of either. We are interested in seeing

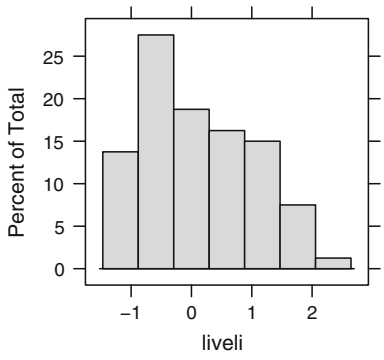

Fig. 10.1 Histogram of forest livelihoods

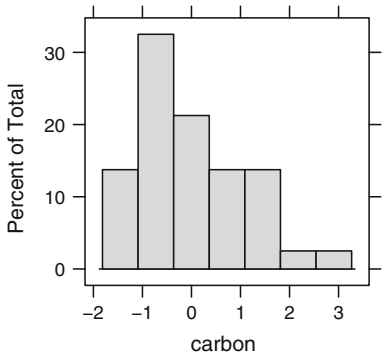

Fig. 10.2 Histogram of carbon storage

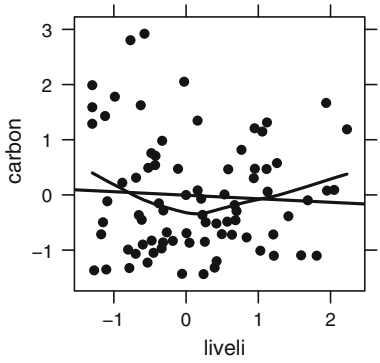

Fig. 10.3 Carbon versus liveli

how the levels of livelihoods and carbon vary among forests with different forest sizes, with ownership by community or state, and by perception of rules by forest users.

We use conditional plots and multiple regression to delve deeper into Fig. 10.3. We will see how the average level of carbon varies with livelihoods, conditional on other variables. Cleveland (1993) showed that conditional plots were a key technique to visualize data, and in R the mosaic package builds on the lattice package to provide this facility.

We examine the scatterplot of carbon versus liveli conditional on lnfsize; 'cutting' lnfsize into four groups with an equal number of observations (Fig. 10.4):

```
> xyplot(carbon ~ liveli | cut(lnfsize, 4), data = ifri,
+      type = c("p", "r", "smooth"), layout = c(4, 1),
+      span = 1)
```

In the code above we used layout to get a one row, four column layout, and span to change the amount of smoothing. We see that the average level of carbon conditional on liveli varies a little with the log of forest size, being higher with higher levels of the

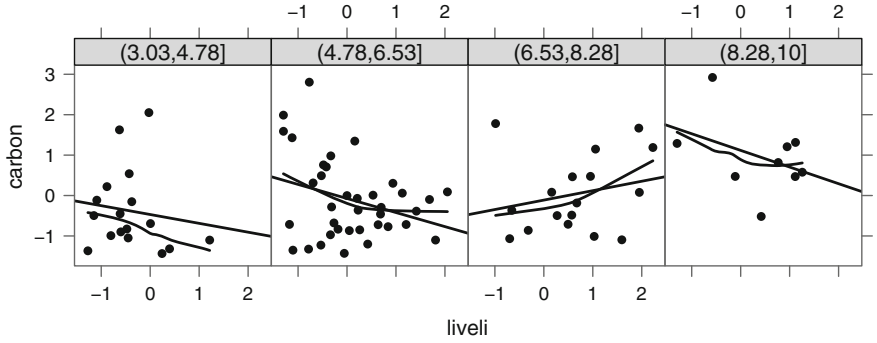

Fig. 10.4 Carbon versus liveli conditional on lnfsize

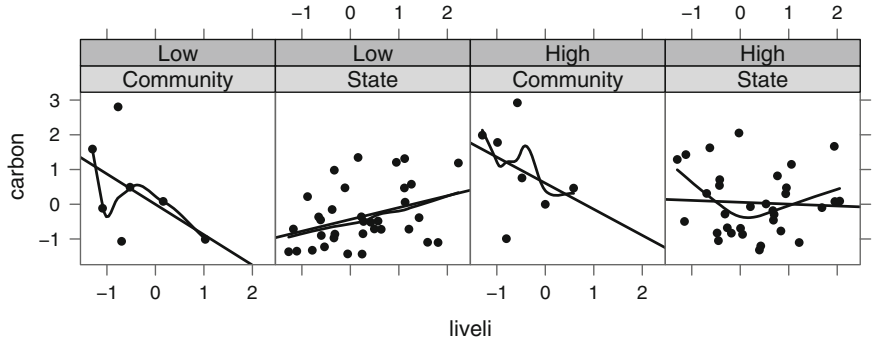

Fig. 10.5 Carbon versus liveli conditional on ownstate and rule match

log of forest size. We now examine the scatterplot of carbon versus liveli conditional on both ownstate and rulematch (Fig. 10.5):

```
> xyplot(carbon ~ liveli | f_own + f_rule, data = ifri,
+      type = c("p", "r", "smooth"), layout = c(4, 1),
+      span = 0.8)
```

The slope of average value of carbon versus liveli appears to be negative for community ownership and positive for state ownership in Fig. 10.5.

10.3 Multiple Regression

Figures 10.4 and 10.5 suggest that forest size, rules and ownership affect the average level of carbon conditional on livelihoods. Therefore, we now run a multiple regression to see how the average value of carbon varies with liveli when it is interacted with lnfsize, rulematch and ownstate.

We run the regression with the lm command, and use colons for interactions between variables:

```
> mod1 <- lm(carbon ~ liveli + ownstate + liveli:ownstate +
+      rulematch + lnfsize + lnfsize:liveli + liveli:rulematch,
+      data = ifri)
```

We use *display* to view the regression output:

```
> library(arm)
> display(mod1)

lm(formula = carbon ~ liveli + ownstate + liveli:ownstate + rulematch +
     lnfsize + lnfsize:liveli + liveli:rulematch, data = ifri)
                 coef.est coef.se
(Intercept)      -1.57     0.58
liveli           -1.63     0.59
ownstate         -0.27     0.32
rulematch         0.64     0.21
lnfsize           0.22     0.07
liveli:ownstate   1.07     0.38
liveli:lnfsize    0.09     0.07
liveli:rulematch -0.15     0.22
---
n = 80, k = 8
residual sd = 0.88, R-Squared = 0.34
```

We can get a figure that shows the coefficients and the confidence intervals (Fig. 10.6) using the mplot command in the mosaic package:

```
> mod.1 <- makeFun(mod1)
> mplot(mod1, which = 7)

[[1]]
```

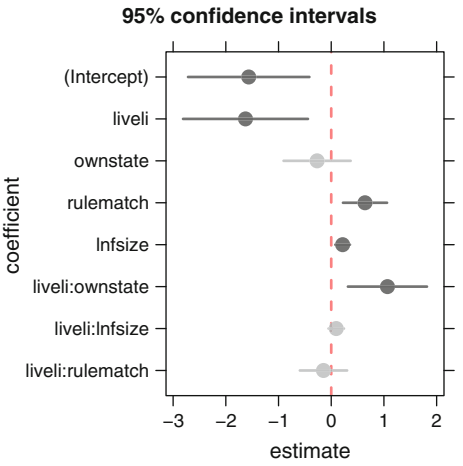

Fig. 10.6 Coefficients and confidence intervals of regression model mod1 (response is carbon)

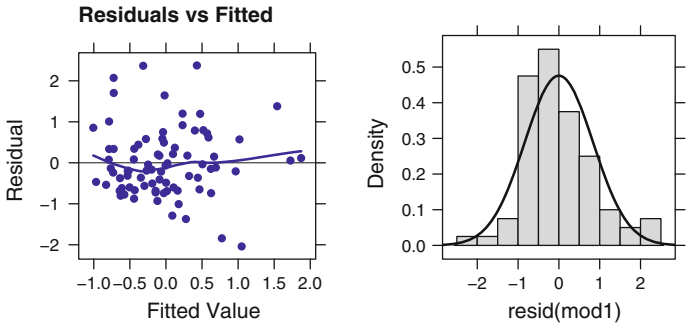

Fig. 10.7 Regression model mod1. Residuals versus fitted (*left*) and histogram of residuals (*right*)

Although the coefficient of ownstate is not statistically significant, the interaction of liveli and ownstate is statistically significant (Fig. 10.6). Although the interactions of liveli and lnfsize and liveli and rulematch are not statistically significant, the coefficients of liveli, rulematch and lnfsize are statistically significant.

We can plot the residuals versus fitted points with `mplot` and can also plot the histogram of residuals. We find there are no strong patterns in the residuals and the distribution of residuals is reasonable (Fig. 10.7).

```
> mplot(mod1, which = 1)

[[1]]

> histogram(~resid(mod1), breaks = 10, fit = "normal",
+     type = "density")
```

To better interpret the deterministic part of the model, which has several interactions, we should plot graphs (Gelman and Hill 2007). First, we use `xyplot` to plot the scatter of points of carbon versus liveli. Second, we use `plotFun` to plot the line of fit of mod1 of carbon versus liveli, with lnfsize $= 5$, ownstate $= 0.5$ and rulematch $= 0.5$. Third, we plot another line of fit, changing only lnfsize to 9. Fourth, we use `ladd` to label the lines (Fig. 10.8). This gives us the following chunk of code:

```
> xyplot(carbon ~ liveli, data = ifri, col = 'gray60')
> plotFun(mod.1(liveli, lnfsize = 5, ownstate = 0.5,
+       rulematch = 0.5) ~ liveli, add = TRUE, lty = 1)
> plotFun(mod.1(liveli, lnfsize = 9, ownstate = 0.5,
+       rulematch = 0.5) ~ liveli, add = TRUE, lty = 2)
> ladd(grid.text("high lnfsize",x=1,y=1,
+                default.units="native"))
> ladd(grid.text("low lnfsize",x=0,y=-1,
+                default.units="native"))
```

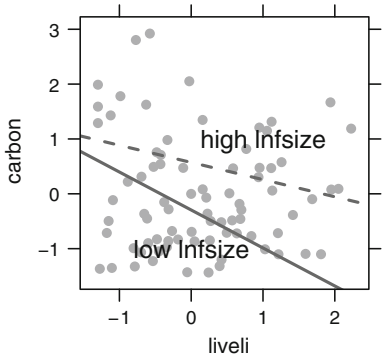

Fig. 10.8 Regression model mod1, carbon versus liveli, lnfsize changing

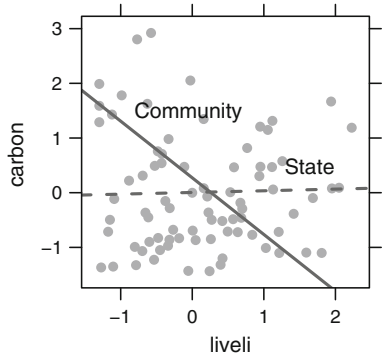

Fig. 10.9 Regression model mod1, carbon versus liveli, ownstate changing

In Fig. 10.8 we see that higher forest size is associated with higher carbon storage and livelihoods.

We now vary ownstate, and modify the code used above to produce Fig. 10.9.

```
> xyplot(carbon ~ liveli, data = ifri, col = 'gray60')
> plotFun(mod.1(liveli, lnfsize = 7, ownstate = 0,
+               rulematch = 0.5) ~ liveli, add = TRUE, lty = 1)
> plotFun(mod.1(liveli, lnfsize = 7, ownstate = 1,
+               rulematch = 0.5) ~ liveli, add = TRUE, lty = 2)
> ladd(grid.text("State", x = 1.7, y = 0.5,
+                default.units = "native"))
> ladd(grid.text("Community", x = 0, y = 1.5,
+                default.units="native"))
```

In Fig. 10.9 we see that in community owned forests, there is a negative association between carbon storage and livelihoods, while there is virtually no association between them in state owned forests. Some community owned forests have relatively high levels of carbon storage and low levels of livelihoods.

We now vary rulematch, and modify the code used above to produce Fig. 10.10.

```
> xyplot(carbon ~ liveli, data = ifri, col = 'gray60')
> plotFun(mod.1(liveli, lnfsize = 7, ownstate = 0.5,
```

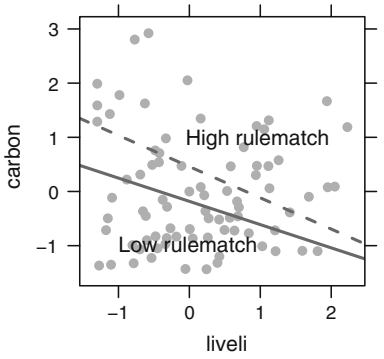

Fig. 10.10 Regression model mod1, carbon versus liveli, rulematch changing

```
+                rulematch = 0) ~ liveli, add = TRUE, lty = 1)
> plotFun(mod.1(liveli, lnfsize = 7, ownstate = 0.5,
+                rulematch = 1) ~ liveli, add = TRUE, lty = 2)
> ladd(grid.text("High rulematch", x = 1, y = 1,
+                 default.units="native"))
> ladd(grid.text("Low rulematch", x = 0, y = -1,
+                 default.units="native"))
```

In Fig. 10.10 we see that a higher level of rule match (local users perceive rules to be appropriate) is associated with higher levels of carbon storage and livelihoods.

10.4 Exploring Further

This chapter has been influenced by the lovely book by Gelman and Hill (2007), which also explains how to use the Cleveland (1993) book on Visualizing Data is testimony to the power of using graphs along with data analysis; Cleveland designed the trellis graphic system which was brought to R by Sarkar (2008). There is much to explore about regression and associated graphics in the car package that accompanies the book by Fox and Weisberg (2011).

References

Chhatre A, Agrawal A (2009) Trade-offs and synergies between carbon storage and livelihood benefits from forest commons. PNAS 106(42):17667–17670

Cleveland WS (1993) Visualizing data. Hobart Press, New Jersey

Fox J, Weisberg S (2011) An R companion to applied regression, 2nd edn. Sage, Thousand Oaks. http://socserv.socsci.mcmaster.ca/jfox/Books/Companion

Gelman A, Hill J (2007) Data analysis using regression and multilevel/hierarchical models. Cambridge University Press, New York

Pruim R, Kaplan D, Horton N (2014) Mosaic: project MOSAIC (mosaic-web.org) statistics and mathematics teaching utilities. R package version 0.9.1-3. http://CRAN.R-project.org/package=mosaic

Sarkar D (2008) Lattice: multivariate data visualization with R. Springer, New York

Chapter 11
Evaluating Training

Abstract We see how matching helps us compare treatment with control groups to evaluate whether a training programme increases earnings. We examine the Lalonde dataset which is included in the MatchIt package in R. We use the MatchIt package to match the treated and control units. We then compare treatment and control groups.

Keywords Lalonde · MatchIt · Lattice · Matching · Evaluation · Experiments

11.1 Introduction

A running debate in statistics is whether we can reach causal conclusions with observational data; or is it that only experiments can help us reach causal conclusions? The Neyman Rubin causal framework has been very influential, and has helped clarify these issues.

In the Neyman Rubin causal framework, we see the effect of a treatment by comparing it to a control. A unit of analysis, for example, a person, could either receive a treatment or not. If the person receives the treatment, we need to compare the resulting outcome with what the same person would have experienced had the person not received the treatment. This is counterfactual, and is not observed. However, if on average those receiving the treatment are similar to those in the control group, the average effect of the treatment can be calculated. A randomized experiment helps ensure the similarity of the treatment and control groups in respects other than the treatment. With observational data we could use a set of relevant covariates to match the treatment and control groups so that we are comparing like with like.

In a key paper, Lalonde (1986) had shown that the experimental evaluation of a training programme and the econometric evaluation with observational data reached different conclusions. However, Dehejia and Wahba (1999) later used *matching* to show that an observational study using matching could reach similar conclusions to the experimental evaluation. In matching, we choose from our observations so that the treatment and control groups are like each other with respect to the covariates.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_11

11.2 Lalonde Dataset

We shall look at a subset of the data used by Lalonde (1986). We load the MatchIt package, and then the dataset called lalonde that it includes.

```
> library(MatchIt)
> library(mosaic)
> data(lalonde)
> trellis.par.set(theme.mosaic(bw = TRUE))
```

We can look at the help for the dataset lalonde with `?lalonde`. We examine the structure of the lalonde dataset.

```
> str(lalonde)
```

```
'data.frame':        614 obs. of  10 variables:
 $ treat   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ age     : int  37 22 30 27 33 22 23 32 22 33 ...
 $ educ    : int  11 9 12 11 8 9 12 11 16 12 ...
 $ black   : int  1 0 1 1 1 1 1 1 1 0 ...
 $ hispan  : int  0 1 0 0 0 0 0 0 0 0 ...
 $ married : int  1 0 0 0 0 0 0 0 0 1 ...
 $ nodegree: int  1 1 0 1 1 1 0 1 0 0 ...
 $ re74    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ re75    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ re78    : num  9930 3596 24909 7506 290 ...
```

There are 614 observations. We want to know whether the treatment (treat) increased the earnings in 1978 (re78)? The treatment was a labour training programme that provided work experience for 6 to 18 months to people with economic and social problems.

We now use a conditional density plot (smoothed histogram, Fig. 11.1) to see the distribution of age in the treatment and control group. We see that the treatment and control groups differ in their balance and do not overlap completely. The balance of the distribution with respect to whether the race of the person is black is also different in the treatment and control groups (Fig. 11.1).

```
> densityplot(~age, groups=factor(treat),data=lalonde,col=c("grey30","black"),
   auto.key=TRUE)
> histogram(~factor(black)|factor(treat),data=lalonde,type='percent')
```

We can also use the `favstats` command in the mosaic package to get our favourite descriptive statistics for the treatment and control group.

```
> favstats(age, data = lalonde, groups = treat)
```

```
  .group min Q1 median Q3 max  mean     sd   n
1      0  16 19     25 35  55 28.03 10.787 429
2      1  17 20     25 29  48 25.82  7.155 185
```

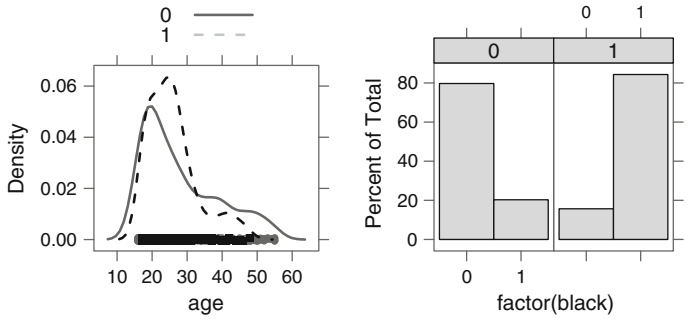

Fig. 11.1 Conditional distributions of age (*left*) and race (*right*) by treatment (*1*) and control group (*0*)

```
  missing
1       0
2       0

> favstats(black, groups = treat, data = lalonde)

  .group min Q1 median Q3 max   mean     sd   n
1      0   0  0      0  0   1 0.2028 0.4026 429
2      1   0  1      1  1   1 0.8432 0.3646 185
  missing
1       0
2       0
```

The maximum age in the treatment group is 48; in the control group it is 55. About 84 % of the treatment group is black compared to only 20 % of the control group.

11.3 Matching Treatment and Control

We will now match the data, using functions in the matchit package. We will not go into the details of matching; we can think of it intuitively. Let an observation in the treatment group be of a person who is a 35 year old black. Let there be 3 control observations we could match the treatment observation to: (1) a 20 year old black, (2) a 32 year old white, and (3) a 32 year old black. We would match the treatment observation to the third control observation who is closest in terms of age and race. Matching has to account for a number of covariates. There are different ways of matching, and in our example we will use the coarsened exact matching technique suggested by Ho et al. (2011), who are also the authors of the MatchIt package.

We now use the *matchit* function to match the data using the coarsened exact matching method.

```
> m.out <- matchit(treat ~ age + educ + black + hispan +
+     married + nodegree + re74 + re75, data = lalonde,
+     method = "cem")

Using 'treat'='1' as baseline group
```

We look at m.out:

```
> m.out


Call:
matchit(formula = treat ~ age + educ + black + hispan + married +
    nodegree + re74 + re75, data = lalonde, method = "cem")

Sample sizes:
          Control Treated
All           429     185
Matched        78      68
Unmatched     351     117
Discarded       0       0
```

The original data has 429 observations in the control group and 185 observations in the treatment group; after matching there are 78 observations in the control group and 68 observations in the treated group.

Since we have many covariates that we are looking at, we can use the propensity score to summarize the probability of being in the treatment group as a function of covariates. We can plot histograms that show how the propensity score balance has improved (Fig. 11.2). This is a function written into the MatchIt package.

```
> plot(m.out, type = "hist")
```

We can recover the data:

```
> m.data <- match.data(m.out)
```

We can also see that the balance has improved among the treatment and control groups with respect to age and race (Fig. 11.3, compare with Fig. 11.1).

```
> densityplot(~age, groups=factor(treat),data=m.data,col=c("grey30","black"),
   auto.key=TRUE)
> histogram(~factor(black)|factor(treat),data=m.data,type='percent')
> favstats(age,data=m.data,groups=treat)

  .group min    Q1 median    Q3 max  mean    sd
1      0  16 16.25     18 20.00  51 19.63 5.367
2      1  17 18.00     20 23.25  48 21.54 5.382
   n missing
1 78       0
2 68       0

> favstats(black,groups=treat,data=m.data)
```

```
  .group min Q1 median Q3 max   mean     sd  n
1      0   0  0      1  1   1 0.6026 0.4925 78
2      1   0  1      1  1   1 0.8676 0.3414 68
  missing
1       0
2       0
```

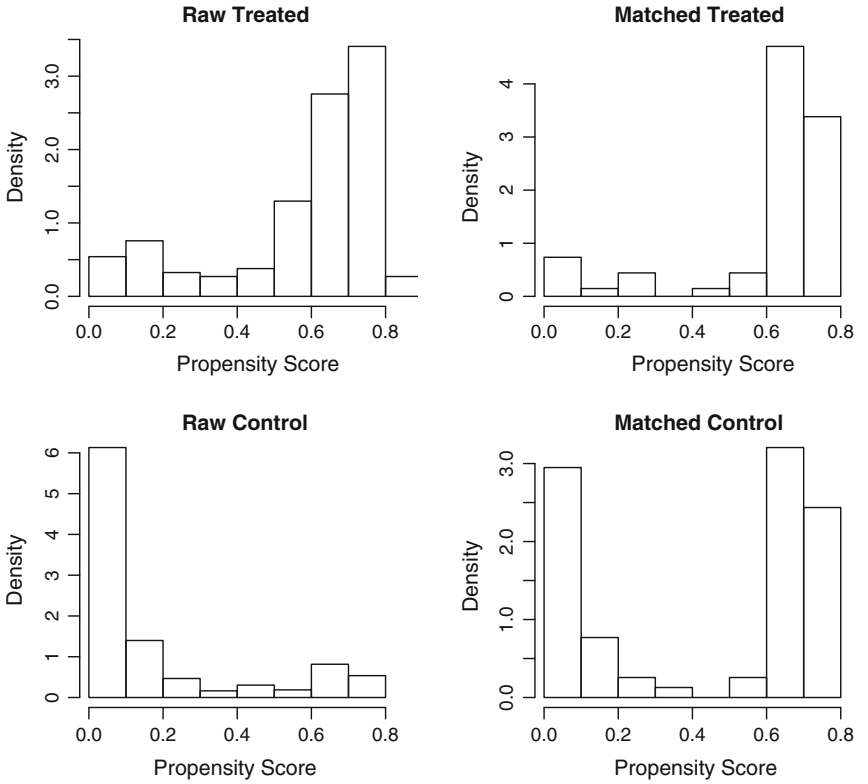

Fig. 11.2 Histograms showing improvement in balance after matching

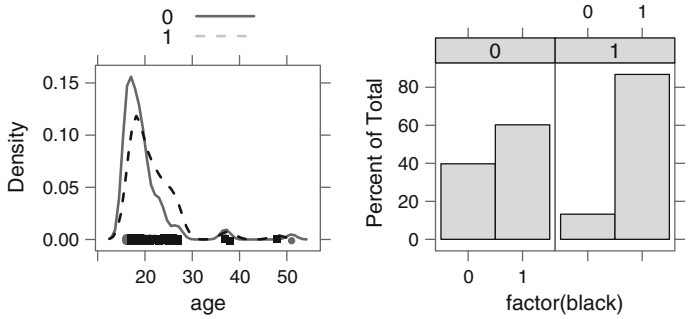

Fig. 11.3 Conditional distributions of age (*left*) and race (*right*) by treatment and control group after matching

11.4 Comparing Treatment and Control

One advantage of matching is that we do not look at the outcome variable while matching; we only check if we have achieved better balance and reduced overlap. We will now compare the difference in earnings between the treatment and control group, before and after matching. We first use *favstats*:

```
> favstats(re78, data = lalonde, groups = treat)

  .group min    Q1 median    Q3   max mean   sd
1      0   0 220.2   4976 11689 25565 6984 7294
2      1   0 485.2   4232  9643 60308 6349 7867
    n missing
1 429       0
2 185       0

> favstats(re78, data = m.data, groups = treat)

  .group min  Q1 median   Q3   max mean   sd  n
1      0   0 140   2505 7065 20243 4874 5667 78
2      1   0   0   4144 9292 34099 5875 6514 68
  missing
1       0
2       0
```

Before matching the control group has higher earnings, whereas after matching the treatment group has higher earnings. We run simple regressions, and plot the results (Figs. 11.4 and 11.5):

```
> fit1 <- lm(re78~treat,data=lalonde)
> library(arm)
> display(fit1)
```

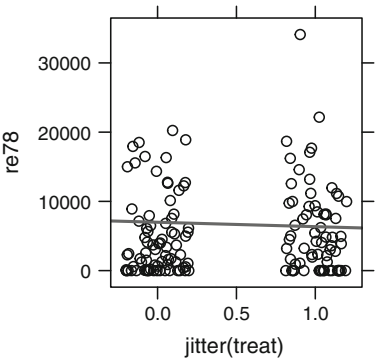

Fig. 11.4 Scatter plot and simple regression fit of earnings on treatment, before matching

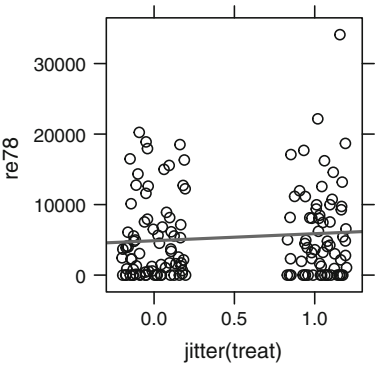

Fig. 11.5 Scatter plot and simple regression fit of earnings on treatment, after matching the data

```
lm(formula = re78 ~ treat, data = lalonde)
            coef.est coef.se
(Intercept) 6984.17   360.71
treat       -635.03   657.14
---
n = 614, k = 2
residual sd = 7471.13, R-Squared = 0.00
```

```
> fit.1 <- makeFun(fit1)
> xyplot(re78 ~ jitter(treat),data=m.data, pch =1 )
> plotFun(fit.1(treat)~treat,add=TRUE)


> fit2 <- lm(re78~treat,data=m.data)
> display(fit2)
```

```
lm(formula = re78 ~ treat, data = m.data)
            coef.est coef.se
(Intercept) 4874.19   687.95
treat       1000.68  1008.05
```

```
---
n = 146, k = 2
residual sd = 6075.83, R-Squared = 0.01
```

```
> fit.2 <- makeFun(fit2)
> xyplot(re78~jitter(treat),data=m.data,pch=1)
> plotFun(fit.2(treat)~treat,add=TRUE)
```

Ho et al. (2011) see matching as a form of preprocessing of the data, not as a form of analysis. To make the comparison doubly robust, they advocate both matching before analysis, and adjusting for covariates after matching.

```
> fit3 <- lm(re78 ~ treat + age + educ + black + hispan +
+     married + nodegree + re74 + re75, data = m.data)
> display(fit3)
```

```
lm(formula = re78 ~ treat + age + educ + black + hispan + married +
    nodegree + re74 + re75, data = m.data)
            coef.est coef.se
(Intercept)   900.74  6689.30
treat        1011.95  1062.07
age            31.50   104.71
educ          351.87   443.18
black        -590.73  1392.13
hispan       3744.69  2110.97
married     -3890.39  2192.12
nodegree     -641.66  2044.51
re74            0.29     0.55
re75            1.11     0.75
---
n = 146, k = 10
residual sd = 5948.15, R-Squared = 0.10
```

After matching the data, we ran a simple regression of earnings on treatment (fit2), and another regression of earnings on treatment with other covariates (fit3). We see that there is not much difference between the estimate of treatment effects in the models fit2 (1001) and fit3 (1012). As Ho et al. (2011) point out, with matching we need to worry less about whether we have specified the model correctly. The crucial issue is whether the covariates we have matched for and adjusted for, are the right ones.

11.5 Exploring Further

Gelman and Hill (2007) have a brief and intuitive but uniquely insightful exposition of matching. Ho et al. (2011) have written a wonderful paper that despite describing the details and most recent developments in matching is surprisingly not difficult to understand; also, the paper is worth reading for a perspective on econometric practice and analysis more generally. Ho et al. (2011) have a very well written and easy to follow paper on the MatchIt package. There are a number of packages

related to matching in R. Angrist and Pischke (2009) have written an entertaining and enlightening book on econometrics, and discuss the Lalonde study at length. The papers by Lalonde (1986) and Dehejia and Wahba (1999) have been very influential.

References

Angrist JD, Pischke JS (2009) Mostly harmless econometrics: an empiricist's companion. Princeton University Press, Princeton

Dehejia RH, Wahba S (1999) Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. J Am Stat Assoc 94(448):1053–1062

Gelman A, Hill J (2007) Data analysis using regression and multilevel/hierarchical models. Cambridge University Press, New York

Ho DE, Imai K, King G, Stuart EA (2011) MatchIt: nonparametric preprocessing for parametric causal inference. J Stat Softw 42(8):1–28. http://www.jstatsoft.org/v42/i08/

Lalonde RJ (1986) Evaluating the econometric evaluations of training programs with experimental data. Am Econ Rev 76(4):604–620

Chapter 12
The Solow Growth Model

Abstract We use the mosaic package to visualize the Solow model and also to compute and plot the values of capital stock over time. We will explore a dataset on long term economic growth, available online at the Maddison Project website. We see time series data of per capita GDP for several leading economies, going back to 1900! We then see how the distribution of income among countries of the world has changed in more recent decades.

Keywords Mosaic package · Solow growth model · Differential equations

12.1 Introduction

The Solow model is usually regarded as a benchmark model by growth economists. According to Jones (2013, p.2), "The modern examination of this question by macroeconomists dates to the 1950s and the publication of two famous papers by Robert Solow of the Massachusetts Institute of Technology. Solow's theories helped to clarify the role of technological progress as the ultimate driving force behind sustained economic growth. "In this chapter we will follow Jones's (2013, p. 3) suggestion to look at growth in terms of the "interplay between observation and theory".

12.2 The Solow Model

We will use the mosaic package to visualize the model. Income (Y) is a function of capital (K) and labour (L). The Cobb-Douglas production function is used. Thus,

$Y = AK^a L^{1-a}$

To simplify, we keep L at 200. We load the mosaic package, and use `makeFun` to make the function Y, giving R values for L, A and a.

```
> library(mosaic)

> Y <- makeFun(A * (K^a) * (L^(1 - a)) ~ K, a = 0.5,
+      A = 2, L = 200)
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_12

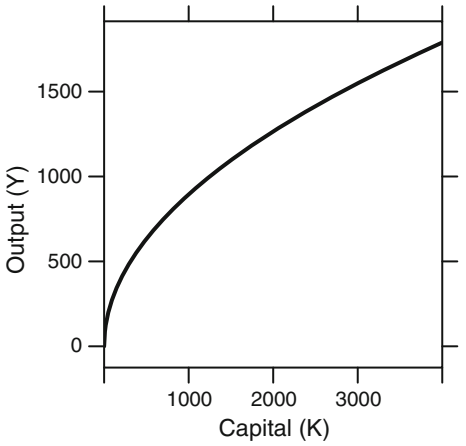

Fig. 12.1 Output and capital

We plot Y with `plotFun` (Fig. 12.1).

```
> plotFun(Y, xlim = range(0, 4000), xlab = "Capital(K)",
+     ylab = "Output(Y)")
```

Savings is a function of income, which in turn is a function of capital.
$S = sY$.
As with Y, we make and plot S (Fig. 12.2).

```
> S <- makeFun(s * A * (K^a) * (L^(1 - a)) ~ K, a = 0.5,
+     A = 2, L = 200, s = 0.2)
> plotFun(S, xlim = range(0, 4000), xlab = "Capital(K)",
+     ylab = "Savings(S)")
```

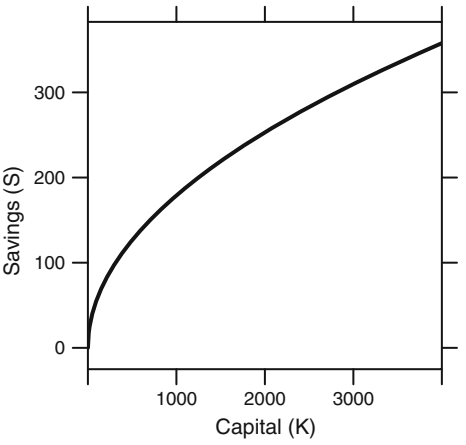

Fig. 12.2 Savings and capital

Depreciation is a certain fraction of the capital stock.

$Dep = dK$

We make Dep now. Note that when giving variables names we should be careful to avoid using the name of an R function; had we used D for depreciation R would have thought we were referring to the D function (used for derivative). We plot S and Dep in one figure:

```
> Dep <- makeFun(d * K ~ K, d = 0.1)
> # warning if we use D instead of Dep causes
> # confusion; D is command
> plotFun(S, xlim = range(0, 4000), xlab = "Capital(K)",
+     ylab = "Savings(S)")
> plotFun(Dep, d = 0.1, xlim = range(0, 4000), xlab = "Capital(K)",
+     ylab = "Depreciation(D)", add = TRUE)
```

The rate of change of capital is savings minus depreciation.

$dK/dt = S - dep$

This is a differential equation; an equation describing how K changes. We can see where K does not change. The point of intersection between savings and depreciation gives us the steady state capital stock, i.e. where $dk/dt = 0$ (Fig. 12.3). We use the `findZeros` command.

```
> Steady.state.K <- findZeros((s * A * (K^a) * (L^(1 -
+     a)) - d * K) ~ K, a = 0.5, A = 2, L = 200, s = 0.2,
+     d = 0.1)
> Steady.state.K

     K
1    0
2 3200
```

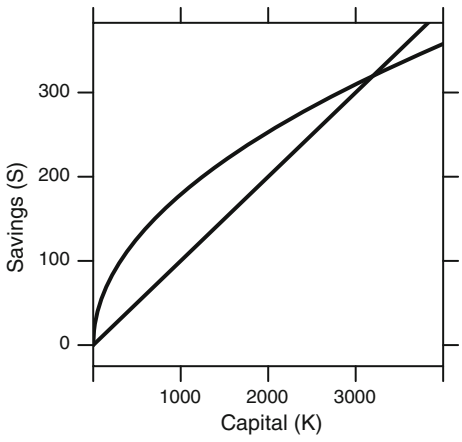

Fig. 12.3 Savings and depreciation versus capital

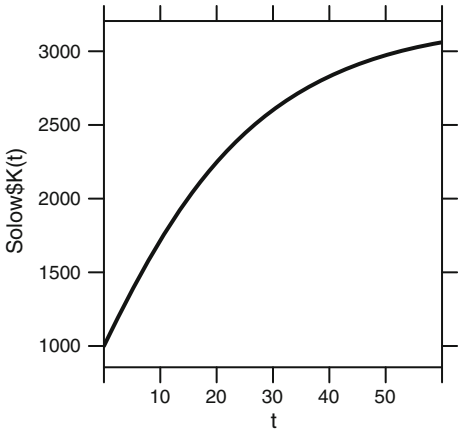

Fig. 12.4 Capital over time

We can solve for capital over time by integrating numerically with the *integrateODE* function:

```
> Solow <- integrateODE(dK ~ -dep * K + (s * A * (K^a) *
+      (L^(1 - a))), K = 1000, a = 0.5, A = 2, L = 200,
+      s = 0.2, dep = 0.1, tdur = list(from = 0, to = 60))
```

The vector of capital values is stored in Solow$K, which we can plot (Fig. 12.4):

```
> plotFun(Solow$K(t) ~ t, t.lim = range(0, 60))
```

Capital, and as a result, output, growth tapers off after a while (Fig. 12.4). For growth to be rekindled, we need a boost to A, i.e. technology.

12.3 Growth Time Series

In this section we explore long run growth data on a few economies. The data can be downloaded from the Maddison Project website (2015). We read in the data and change the file name to something shorter and convenient.

```
> #need to change the format below and path as needed
> mpd_extract2_subset_US_japan <-
+   read.csv
   ("~/Documents/R/RMa3/growth_data_explore/mpd_extract2_subset_US_japan.csv")
> myd <- mpd_extract2_subset_US_japan
```

Since we are looking at long run growth, we compute the ratio of per capita GDP in 2010 (in row 111) to that in 1900 (in row 1):

```
> ratio.2010.1900 <- myd[111, 2:6]/myd[1, 2:6]
> ratio.2010.1900
```

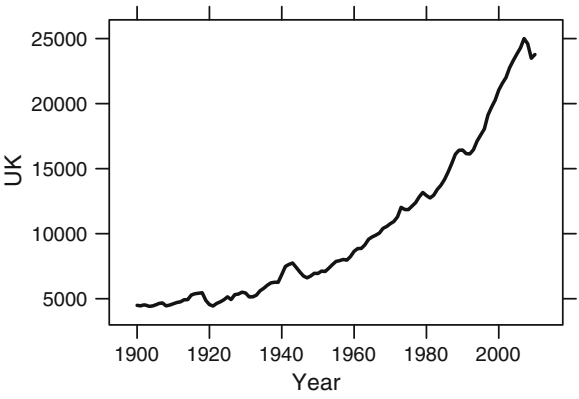

Fig. 12.5 Per capita GDP in 1990 GK dollars for the UK

```
       UK   USA Argentina India Japan
111 5.293 7.453     3.567 5.629 18.59
```

Between 2010 and 1900 the economies of the UK, US, Argentina, India and Japan, are estimated to have grown about 5, 7, 3, 6, and 19 times respectively.

We can plot the per capita GDP in 1990 GK dollars for the UK (Fig. 12.5).

```
> xyplot(UK ~ Year, data = myd, type = "l")
```

We can also plot the log of the series for the UK. This makes it linear. We prefer using the log because the slope gives us the growth rate (Fig. 12.6).

```
> xyplot(log(UK) ~ Year, data = myd, type = "l")
```

We now make comparisons for how different economies grew. When we use `xyplot`, we use the + sign to tell R that we want to plot a number of time series on the same graph. We use the `ladd` function to add text to the graph plotted by `xyplot`, giving the coordinates of where we want the text.

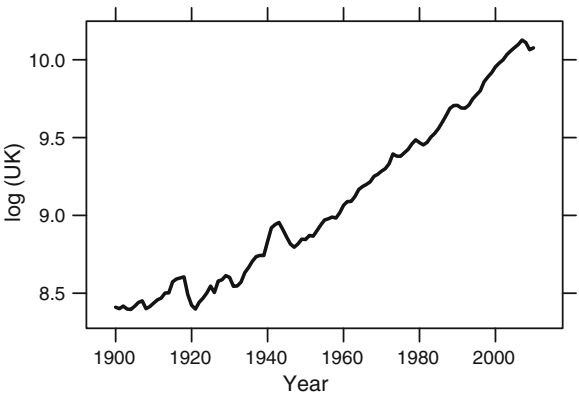

Fig. 12.6 Logarithm of per capita GDP in 1990 GK dollars for the UK

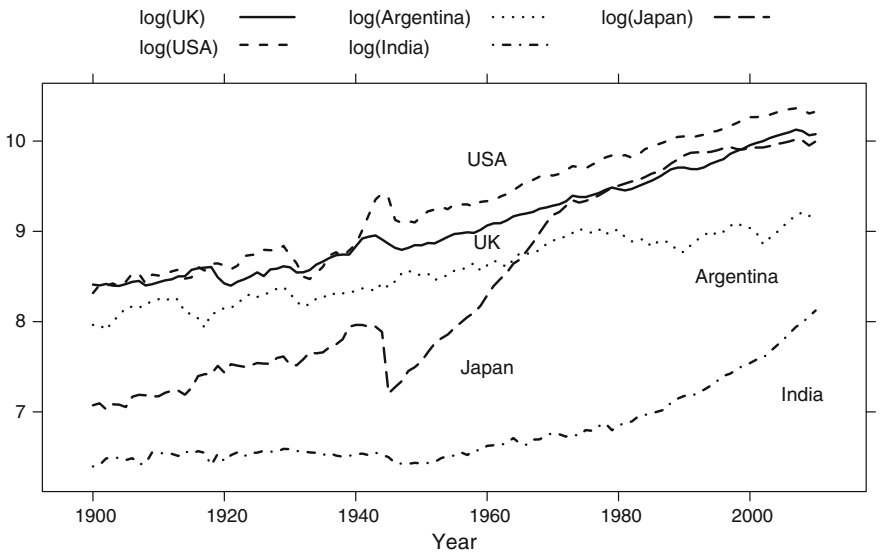

Fig. 12.7 Logarithm of per capita GDP for the UK, USA, Japan, Argentina, and India

```
> xyplot(log(UK) + log(USA) + log(Argentina) + log(India) +
+     log(Japan) ~ Year, data = myd, type = "l", col = "black",
+     ylab = "", auto.key = list(lines = TRUE, points = FALSE,
+         columns = 3))
> ladd(grid.text("USA", x = 1960, y = 9.8, default.units = "native"))
> ladd(grid.text("India", x = 2008, y = 7.2, default.units = "native"))
> ladd(grid.text("Argentina", x = 1998, y = 8.5, default.units = "native"))
> ladd(grid.text("Japan", x = 1960, y = 7.5, default.units = "native"))
> ladd(grid.text("UK", x = 1960, y = 8.9, default.units = "native"))
```

We can see that over the long run, the US grew steadily, with the UK lagging just a bit (Fig. 12.7). Japan grew really fast after the second World War. Argentina started as a relatively prosperous country, but the gap between it and the US grew larger. Towards the end of the 20th century Indian growth picked up.

12.4 Distribution Over Time

We will now use World Development Indicators (World Bank 2014) data to examine income distribution across the countries of the world.

```
> gdp_pc_time <- read.csv
   ("~/Documents/R/RMa2/growth_data_explore/growth_data_explore/gdp_pc_time.csv")
> gdp <- gdp_pc_time
```

A histogram shows that the distribution of GDP per capita (constant 2005 US$) across countries is very positively skewed (Fig. 12.8).

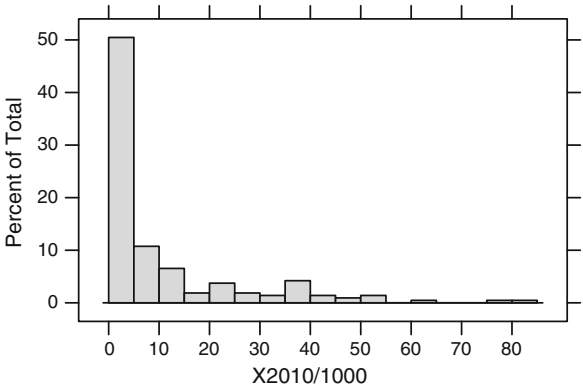

Fig. 12.8 Histogram of GDP per capita (in thousands) of countries in 2010

```
> histogram(~X2010/1000, breaks = 20, type = "percent",
+     data = gdp, scales = list(x = list(at = seq(0,
+         80, 10))))
```

We calculate the ratio of GDP per capita in 2010 to the GDP per capita in 1980.

```
> gdp$ratio <- gdp$X2010/gdp$X1980
> favstats(ratio, data = gdp)
```

```
    min     Q1 median     Q3    max  mean     sd   n
 0.2955 1.128  1.513 2.011 13.02 1.771 1.321 132
 missing
      82
```

The median value of the GDP per capita in 2010 to the GDP per capita in 1980 across countries is 1.5 and the mean value is 1.8. We list the countries for which this ratio is more than 4 and less than 0.3.

```
> subset(gdp, subset = ratio > 4 | ratio < 0.3, select = c(Country.Name,
+     X1980, X2010, ratio))
```

```
             Country.Name   X1980 X2010   ratio
23                 Bhutan   325.1  1795  5.5222
35             Cape Verde   557.6  3144  5.6384
41                  China   220.4  2869  13.0152
102          Korea, Rep.  4270.4 20625  4.8297
202 United Arab Emirates 81947.2 24219  0.2955
```

Bhutan and Cape Verde grew more than 5 times, China grew 13 times, while the United Arab Emirates grew smaller. We do a scatterplot of ratio of GDP per capita of countries in 2010 to GDP per capita in 1980 versus GDP per capita in 1980 (Fig. 12.9). China is an outlier in Fig. 12.9. We can also see that generally growth was evenly spread across rich and poor countries (Fig. 12.9).

```
> xyplot(ratio ~ log10(X1980), data = gdp, type = c("p",
+     "smooth"))
```

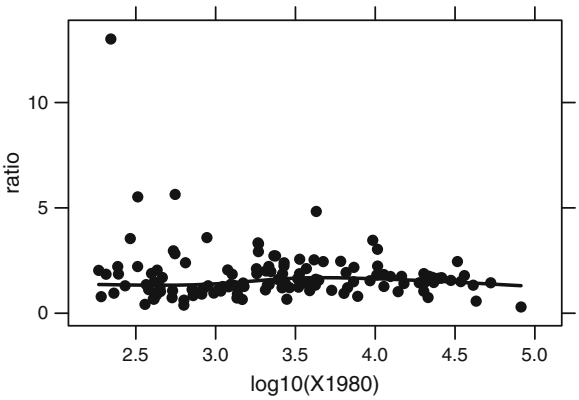

Fig. 12.9 Scatterplot of ratio of GDP per capita of countries in 2010 to GDP per capita in 1980 versus GDP per capita in 1980

12.5 Exploring Further

Jones (2013) and Weil (2012) cover growth theory and explore relevant data. There is a lot to explore at the Maddison Project website (2015).

References

Jones CI (2013) Introduction to economic growth, Indian edn. Viva, New Delhi in arrangement with Norton

Maddison Project (2015). http://www.ggdc.net/maddison/maddison-project/home.htm. Accessed 2 Mar 2015

Weil DN (2012) Economic growth, 3rd edn. Prentice Hall

World Bank (2014) World development indicators http://databank.worldbank.org/data/home.aspx. Accessed 7 Jan 2014

Chapter 13
Simulating Random Walks and Fishing Cycles

Abstract We simulate simple deterministic difference equations with loops. We then simulate white noise and a random walk. Finally, we simulate fish growth and harvest.

Keywords Difference equation · Simulation · White noise · Random walk · Logistic growth · Fish

13.1 Introduction

Difference equations are like differential equations with discrete adjustment. In time series econometrics difference equations with stochastic elements are used. Difference equation versions of the logistic growth equations are commonly used in ecology.

13.2 Difference Equations

In a difference equation we have a relationship between the value of a variable in the current period and the value in the previous period. Let us take the example of a difference equation for a variable x.

$x_t = ax_{t-1} + b$

This can also be written in the following form by subtracting x_{t-1} from both sides:

$\Delta x_t = x_t - x_{t-1} = (a-1)x_{t-1} + b$

If $x_t = x_{t-1}$, then x is said to be in equilibrium or a steady state. Denoting the steady state by x^*, we can see that

$x^* = b/(1-a)$

If we know x_{t-1} we can calculate x_t; if we know the initial value of x we can go on calculating the next value. This is something computers can do very well, and in R we can use a loop that repeats the calculation.

V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_13

We examine a simple difference equation first; x in this period is simply 0.5 times x in the last period plus two.

$x_t = 0.5\, x_{t-1} + 2$

We first create a vector x with 30 elements. We give the first element of x (the initial value) a value of 100.

```
> x = numeric(30)
> x[1] = 100
```

We then make a loop; t (which stands for time) goes from 2 to 30. Each time its value increases by one, we multiply the previous value of x by 0.5 and add 2.

```
> for (t in 2:30) {
+     # t stands for time +
      x[t] <- 0.5 * x[t - 1] + 2  # the difference equation
+ }
```

```
> library(mosaic)
```

To plot the values of x against time with `xyplot` we make a vector time which has the same values as t, going from 1 to 30.

```
> time <- 1:30
> xyplot(x ~ time, type = "l")
```

In Fig. 13.1 we see that x drops from a value of 100 to 4. We can print the values of x from time $= 1$ to 10, and time $= 20$ to 30:

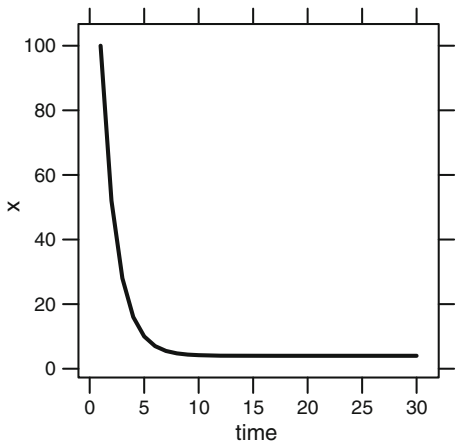

Fig. 13.1 Plot of x for equation x in time t $= 0.5*$ x in previous time $+2$

```
> x[1:10]

 [1] 100.000  52.000  28.000  16.000  10.000
 [6]   7.000   5.500   4.750   4.375   4.188

> x[20:30]

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

13.3 Stochastic Elements

We will now add stochastic elements to the difference equations; first we examine the plot of white noise. We draw a random sample of size 100 from a normal distribution using the function *rnorm*.

```
> white <- rnorm(80)
> time <- 1:80
```

Autocorrelation is the correlation between lagged values of a time series, for example, between x_t and x_{t-1} or x_t and x_{t-2}. The acf is the graph of such correlations. We use the acf of white noise as a benchmark.

```
> plot(white, type = "l", las = 1)
> acf(white, las = 1, main = "")
```

In Fig. 13.2 (left) we see a plot of white noise—there are random movements up and down. The acf of white noise (Fig. 13.2 right) shows negligible correlation between lags.

We now add white noise to our difference equations, so we have $x_t = 0.5\, x_{t-1} + 2 + w$. We simulate this and plot it (Fig. 13.3):

```
> x = numeric(80)
> x[1] = 100
> w <- 10 * rnorm(80)
> for (t in 2:80) {
```

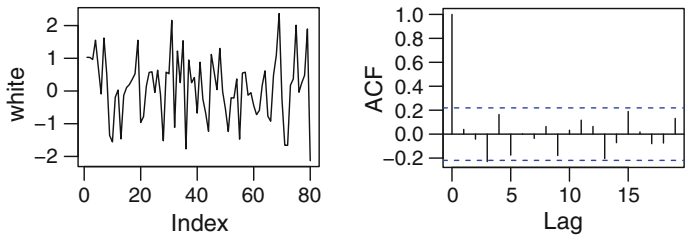

Fig. 13.2 White noise (*left*) and acf of white noise (*right*)

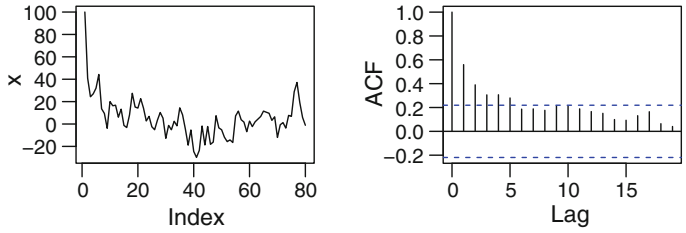

Fig. 13.3 Difference equation with white noise

```
+      x[t] <- 0.5 * x[t - 1] + 2 + w[t]
+ }
> plot(x, type = "l", las = 1)
> acf(x, main = "", las = 1)
```

We can compare Fig. 13.3 with Figs. 13.1 and 13.2. Initially x drops and then it fluctuates. The acf shows one significant lag.

13.4 Random Walk

We now look at a random walk.

$$x_t = x_{t-1} + w$$

```
> x = numeric(300)
> x[1] = 200
> w <- 10 * rnorm(300)
> for (t in 2:300) {
+      x[t] <- x[t - 1] + w[t]  #random walk
+ }
```

We plot the random walk (Fig. 13.4 left) and its acf (Fig. 13.4 right).

```
> plot(x, type = "l", las = 1)
> acf(x, main = "")
```

We can compare Fig. 13.4 with Fig. 13.2. The acf in Fig. 13.4 shows slowly decaying lags. In a random walk, if there is a large random shock, it persists and influences future values of x.

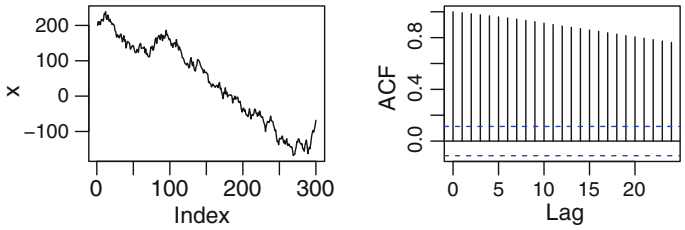

Fig. 13.4 Random walk (*left*) and acf (*right*)

13.5 Fishing

Bjorndal and Conrad (1987) modelled the open access exploitation of North Sea herring during the period 1963–1977. Fish populations are often modelled with a logistic equation.

Denoting the fish stock with S, the intrinsic rate of growth by r, and its carrying capacity by L, we have: $S_t = S_{t-1} + rS_{t-1}(1 - (S_{t-1}/L))$.

If $S_{t-1} = 0$ or L, the growth is zero. Growth in fish depends on the stock of fish. We first see how S changes over time if the fish are left alone, i.e. there is no harvest.

We need to give R the formula, the values for r and L, and the value for initial S.

```
> S = numeric(15)
> S[1] = 2325000
> r = 0.8
> L = 3200000
```

We use a loop as before, within the loop we use the formula for logistic growth to calculate the stock S in each period.

```
> for (t in 2:15) {
+     S[t] <- S[t - 1] + (S[t - 1] * r) * (1 - S[t -
+         1]/L)
+ }
> Time <- 1:15
```

We now plot S against Time (Fig. 13.5):

```
> xyplot(S/10^6 ~ Time, type = "p")
```

The fish stock increases until it is equal to the carrying capacity (Fig. 13.5).

We now examine the fish stock with fishing. Fishing harvest depends on the fish stock S2 and fishing capital, Bjorndal and Conrad (1987) use a Cobb-Douglas production function. We will not go into the details of the derivation, but they arrive at the following dynamic system that they use for a simulation:

$K_{t+1} = K_t + n(aK_t^{b-1}S_t^g - c_t/p_t)$

$S_{t+1} = S_t + rS_t(1 - S_t/L) - aK_t^b S_t^g$

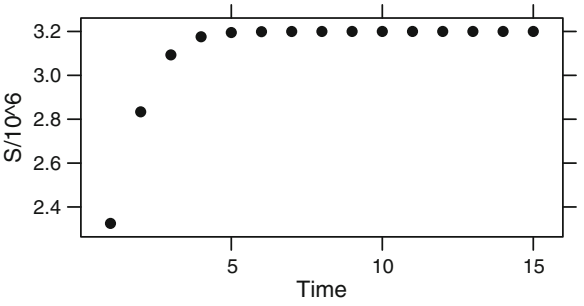

Fig. 13.5 Fishing stock with no harvest

The equation on the top represents the adjustment of capital to profit—higher profits lead to an expansion of capital. We had used the lower equation earlier; it represents the biological growth and harvest of fish. We now have more parameters that we have to provide numerical values for:

```
> S2 = numeric(15)
> K = numeric(15)
> S2[1] = 2325000
> K[1] = 120
> r = 0.8
> L = 3200000
> a = 0.06157
> b = 1.356
> g = 0.562
> n = 0.1
> c <- c(190380, 195840, 198760, 201060, 204880, 206880,
+     215220, 277820, 382880, 455340, 565780, 686240,
+     556580,721640, 857000)
> p <- c(232, 203, 205, 214, 141, 128, 185, 262, 244,
+     214, 384, 498, 735, 853, 1415)
```

We put the two equations into a loop:

```
> for (t in 2:15) {
+     S2[t] <- S2[t - 1] + (S2[t - 1] * r) * (1 - S2[t -
+         1]/L) - a * K[t - 1]^b * S2[t - 1]^g
+     K[t] <- K[t - 1] + (n * (a * (K[t - 1]^(b - 1)) *
+         (S2[t - 1]^g) - c[t - 1]/p[t - 1]))
+ }
```

We plot K against S (Fig. 13.6):

```
> xyplot(K ~ S2/10^6, type = "l")
```

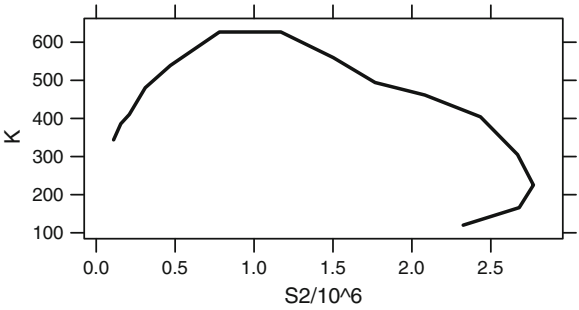

Fig. 13.6 Capital versus fishing stock, with open access harvest

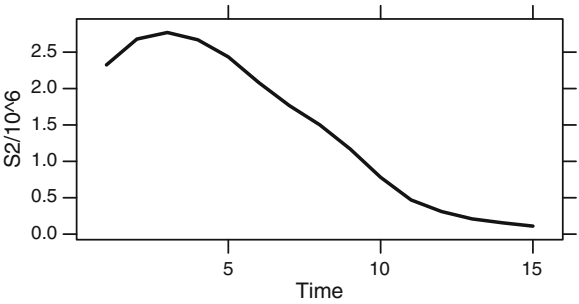

Fig. 13.7 Fishing stock with open access harvest

and S against Time (Fig. 13.7):

```
> xyplot(S2/10^6 ~ Time, type = "l")
```

We see that with open access fishing the dynamics of the fishery is dramatically changed from the case of no fishing. In Fig. 13.6 we see that initially fishing leads to an expansion of capital which over time leads to a reduction in the fishing stock. When capital is around 600, it starts falling but the reduction in capital is a case of too little, too late. We can contrast Fig. 13.7 with Fig. 13.5; in Fig. 13.7 the fish die out.

In their conclusions Bjorndal and Conrad (1987, pp. 83–84) write:

> In the empirical analysis of open access systems it is important to note that non-linear difference equations, with or without longer lags, are capable of more complex dynamic behaviour than their continuous-time (differential equation) analogues. The lag in adjustment by both the exploited species and the harvesters themselves is often a more accurate depiction of dynamics, and the differential equation systems are best viewed as theoretical approximations. ... If adjustment in an open access system is discrete, there is a greater likelihood of overshoot, severe depletion, and possibly extinction. ... the North Sea herring fishery ... was saved from more severe overfishing and possibly extinction by the closure of the fishery at the end of the 1977 season.

13.6 Exploring Further

Cowpertwait and Metcalfe (2009) provide a delightful and accessible introduction to time series. They use simulation as a tool for understanding.

The Conrad (2010) book uses numerical examples in Excel to make it easier to understand difficult mathematical models in resource economics.

References

Bjorndal T, Conrad JM (1987) The dynamics of an open access fishery. Can J Econ 20(1):74–85
Conrad JM (2010) Resource economics. Cambridge University Press, New York
Cowpertwait PSP, Metcalfe AV (2009) Introductory time series with R. Springer, Dordrecht

Chapter 14
Basic Time Series

Abstract We graph different time series in this chapter and also do a few computations and a bit of modeling. We begin by looking at some Air Passenger data, and find that it has both seasonal and trend components. We look at scatterplots of inflation versus unemployment in the US before and after 1970—and find two different curves. We graph inflation data and examine its stationarity. We use a very simple model to forecast inflation and graph the forecast. Finally, we explore the volatility in the stock market by calculating standard deviations for different time periods.

Keywords Time series · Forecasting · Autoregressive model · Volatility · Inflation

14.1 Introduction

Time series econometrics has grown enormously in the last few decades. Time series models tackle several complications that arise in time series. A good starting point for time series is graphing and simple computations. Time series come in different shapes and sizes and this variety in time series can be best appreciated through graphs. Graphs can guide models used in forecasting, and then help us see the uncertainty in the forecast alongside past movement in the time series.

14.2 Air Passengers

We begin this chapter with some air passenger data described by Cowpertwait and Metcalfe (2009; p. 4), as "The number of international passenger bookings (in thousands) per month on an airline (Pan Am) in the United States …obtained from the Federal Aviation Administration for the period 1949–1960".

The data is already in R, AirPassengers, we load it and rename it.

```
> data(AirPassengers)
> APass <- AirPassengers
```


V. Dayal, *An Introduction to R for Quantitative Economics*,
SpringerBriefs in Economics, DOI 10.1007/978-81-322-2340-5_14

We use the `class` function to find out what the nature of APass is.

```
> class(APass)

[1]"ts"

> head(APass)

[1] 112 118 132 129 121 135
```

APass is a time series (ts) object.

We plot APass (Fig. 14.1), using the las option so the y-axis labels are horizontal.

```
> plot(APass, las = 1)
```

There is a steady increase in the number of Air Passengers, and there is a seasonal element. We plot the first year (first twelve months) of data and the last year (last twelve months) in one figure (Fig. 14.2):

```
> APfirst <- ts(APass[1:12])
> APlast <- ts(APass[133:144])
> ts.plot(APlast, APfirst, las = 1, lty = c(1, 2))
```

Figure 14.2 extracts the key elements of Air Passengers: the upward shift in APass, and the pronounced seasonal effect. We can also use the following figure which puts together the data for different years by month:

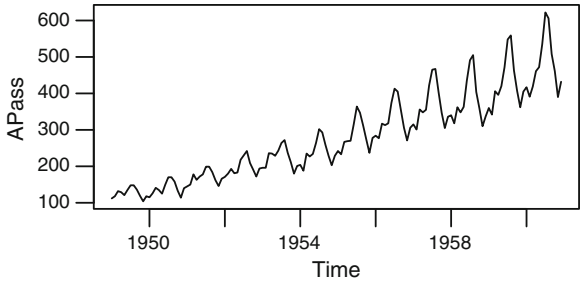

Fig. 14.1 Air passengers in thousands per month

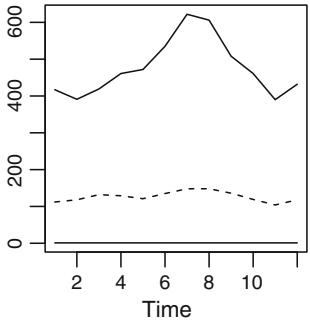

Fig. 14.2 Air passengers over time, last 12 months (1960) on *top*, first twelve months (1949) *below* (*dashed line*)

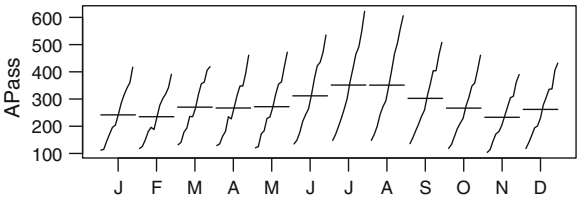

Fig. 14.3 Air passengers per month for different years arranged by month

```
> monthplot(APass, las = 1)
```

July and August are the months with the highest number of air passengers, and there is an increase in every month over the years (Fig. 14.3).

14.3 The Phillips Curve

Kleiber and Zeileis (2008) have written a fine book called Applied Econometrics with R. This has an accompanying R package called AER with some key datasets. This package includes data and relevant code for examples in the textbook by Stock and Watson (2011).

We can get the data into R with:

```
> data("USMacroSW", package = "AER")
```

We can calculate the rate of inflation and bind it to the existing columns, ts.intersect helps us avoid blank rows since we are using differences. When we take the difference of the log of the consumer price index (cpi) we get the rate of inflation. We multiply by 100 to get percentages, and since the data is quarterly we multiply by 4 to get rates of inflation on a quarterly basis.

```
> usm <- ts.intersect(USMacroSW, 4 * 100 * diff(log(USMacroSW[,
+     "cpi"])))
```

We add in infl (for inflation) to the names of columns and print.

```
> colnames(usm) <- c((colnames(USMacroSW)), "infl")
> colnames(usm)
```

```
[1] "unemp"  "cpi"    "ffrate" "tbill"  "tbond"
[6] "gbpusd" "gdpjp"  "infl"
```

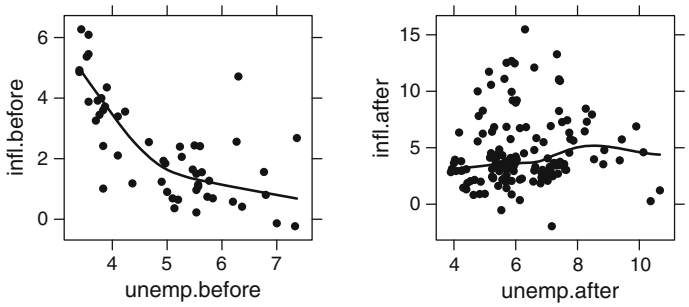

Fig. 14.4 Inflation versus unemployment in the USA before 1970 (*left*) and after 1970 (*right*)

Is there a tradeoff between inflation and unemployment? Following Leamer (2010), we use graphs to see how the Phillips curve changed over time (Fig. 14.4). We divide the series into two—before and after 1970:

```
> unemp1 <- usm[, "unemp"]
> unemp.before <- unemp1[time(unemp1) < 1970]
> unemp.after <- unemp1[time(unemp1) >= 1970]
> infl1 <- usm[, "infl"]
> infl.before <- infl1[time(infl1) < 1970]
> infl.after <- infl1[time(infl1) >= 1970]
```

We now plot inflation versus unemployment before and after 1970 (Fig. 14.4):

```
> library(mosaic)

> xyplot(infl.before ~ unemp.before, type = c("p", "smooth"))
> xyplot(infl.after ~ unemp.after, type = c("p", "smooth"))
```

We quote Leamer (2010, p. 4), who tells us what the implications of Fig. 14.4 are:

> You can see the Phillips Curve clearly in the figure at the left ... The Phillips curve offered governments some very nasty medicine for unwanted inflation: increase unemployment. The good news for workers around the globe is that the story of the Phillips curve was never very compelling and the subsequent movements in US data made the curve much more difficult to see (see the figure at the right).

14.4 Forecasting Inflation

We try to forecast inflation into the future from its past values. Before we do that, we examine it to see whether it is stationary.

We can plot inflation (Fig. 14.5 left) and its acf (Fig. 14.5 right):

```
> infl.1 <- usm[, "infl"]
> plot(infl.1, las = 1)
> acf(infl.1, main = "")
```

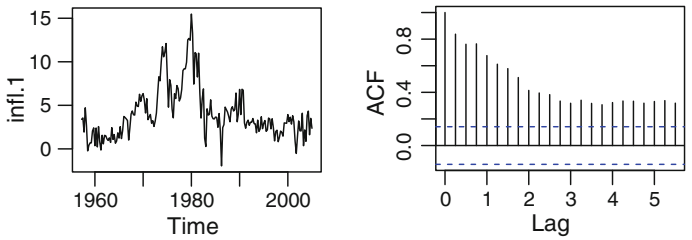

Fig. 14.5 Inflation (*left*) and acf of inflation (*right*)

The acf indicates non-stationarity as it decays very slowly. We can use a formal test. In the Dickey Fuller test, we basically test whether $\beta_1 = 1$ in the equation $Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t$.

It is better to use the Augmented Dickey-Fuller test, which augments the Dickey-Fuller test by lags of the difference of Y.

We install and then load the package tseries; then use the function *adf.test*.

```
> # install.packages('tseries')
> library(tseries)
> adf.test(usm[, "infl"])

    Augmented Dickey-Fuller Test

data:  usm[, "infl"]
Dickey-Fuller = -2.572, Lag order = 5,
p-value = 0.3366
alternative hypothesis: stationary
```

Because our null hypothesis that a unit root is present was not rejected, we difference inflation to achieve stationarity, and then check graphically (Fig. 14.6) and with a formal test:

```
> diff.infl <- diff(usm[, "infl"])
> plot(diff.infl)
> acf(diff.infl, main = "")

> adf.test(diff((usm[, "infl"])))

Warning: p-value smaller than printed p-value


    Augmented Dickey-Fuller Test

data:  diff((usm[, "infl"]))
Dickey-Fuller = -6.112, Lag order = 5,
p-value = 0.01
alternative hypothesis: stationary
```

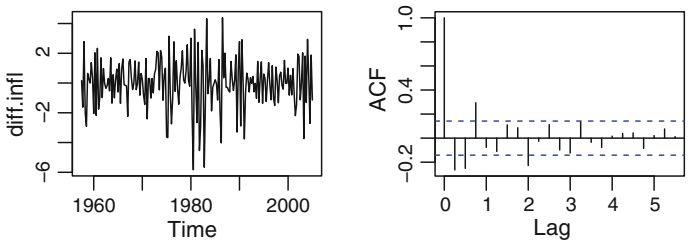

Fig. 14.6 Difference of inflation (*left*) and acf (*right*)

The null hypothesis of a unit root is now rejected.

A simple way to forecast a series Z is to use its past values. For example, if we estimate a first order autoregressive model,

$Z_t = \beta_0 + \beta_1 Z_{t-1} + u_t$

we can get a forecast for Z for a period ahead with

$Z_{T+1} = \beta_0^{EST} + \beta_1^{EST} Z_T$,

where EST denotes estimate.

We might find that our forecasts improve with more lags than one. Stock and Watson (2011) suggest using a criterion such as the Bayes information criterion to choose the number of lags. In this case, two lags minimize the BIC.

We use the forecast package (Hyndman et al. 2014) to estimate the forecast and plot the result:

```
> # install.packages('forecast')
> library(forecast)
```

We use the `Arima` function, we choose AR of order 2, integrated of order 1, and no MA terms.

```
> fit_ar4 <- Arima(infl1, order = c(2, 1, 0))
```

We print the forecast

```
> forecast(fit_ar4, level = 95)
```

```
         Point Forecast   Lo 95 Hi 95
2005 Q2          2.119 -0.8937 5.132
2005 Q3          2.601 -0.9854 6.187
2005 Q4          2.515 -1.2931 6.323
2006 Q1          2.379 -1.8707 6.629
2006 Q2          2.457 -2.1987 7.112
2006 Q3          2.476 -2.4754 7.428
2006 Q4          2.443 -2.8119 7.697
2007 Q1          2.448 -3.1086 8.004
```

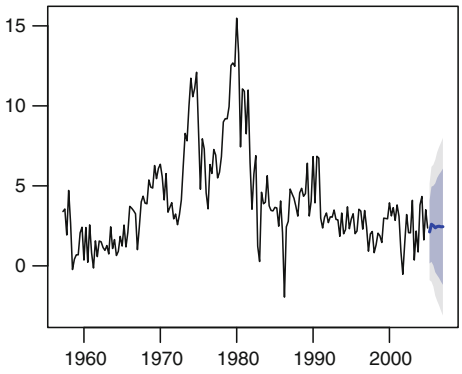

Fig. 14.7 Forecast for inflation from 2005 Q2 to 2007 Q1

and then plot the forecast

```
> plot(forecast(fit_ar4), las = 1, main = "")
```

The plot of the forecast (Fig. 14.7) helps place the numerical forecast in the context of the time series and how it changed historically. We can also use the `auto.arima` function in the forecast package (see Hyndman and Athanasopoulos 2014). This is based on an algorithm and outperforms attempts by beginners.

14.5 Volatility in the Stock Market

We now examine volatility in the stock market graphically. This is another example from Stock and Watson (2011), with the data contained in the AER package in R. Stock and Watson use this example to exposit autoregressive conditional heteroskedasticity (ARCH) and generalized ARCH (GARCH) models; we will be content to examine the standard deviation of the time series for different sub-periods. The time series is the daily New York Stock Exchange stock price index from 1990 to 2005.

```
> library(AER)
> data("NYSESW", package = "AER")
```

We calculate daily percentage changes (dpc):

```
> dpc <- 100 * diff(log(NYSESW))
> str(dpc)

'zoo' series from 1990-01-03 to 2005-11-11
  Data: num [1:4002] -0.101 -0.766 -0.844 0.354 -1.019 ...
  Index:  Date[1:4002], format: "1990-01-03" "1990-01-04" ...

> length(dpc)
```

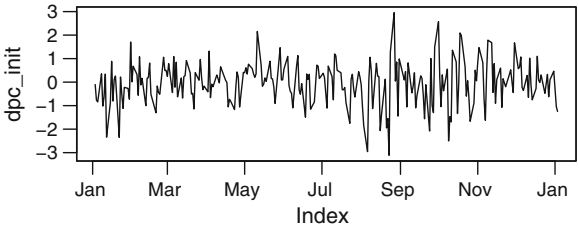

Fig. 14.8 Daily percentage changes, 1990 Jan–1991 Jan

```
[1] 4002
```

There are a lot of observations! We plot the data for the first year (Fig. 14.8):

```
> dpc_init <- window(dpc, start = as.Date("1990-01-03"),
+     end = as.Date("1991-01-03"))
> plot(dpc_init, las = 1)
```

In the stock markets, a month is a long time. We estimate the standard deviation at monthly intervals and then plot the result (Fig. 14.9). We use the xts package (Ryan and Ulrich 2014) as follows:

```
> library(xts)
> dpc5 <- as.xts(dpc)
> dpc_sd_monthly <- apply.monthly(dpc5, sd)
> plot(dpc_sd_monthly, las = 1)
```

In Fig. 14.9 we get a glimpse of periods of high and low volatility. In 2002, for example, the standard deviation was high, in the range of about 1–2.5, whereas in 2004 it was low, in the range of about 0.5–1.

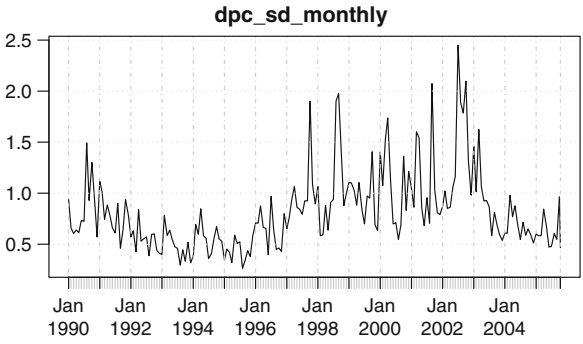

Fig. 14.9 Monthly standard deviation of daily percentage changes in the New York Stock Exchange Index

14.6 Exploring Further

Hyndman and Athanasopoulos (2014) have written a must-read free online text on forecasting; it uses R, and is both simply written and state of the art. Cowpertwait and Metcalfe's (2009) introduction to time series uses R. Teetor's (2011) R Cookbook has a chapter on Time Series Analysis. Kleiber and Zeileis (2008) have written a wonderful book with an accompanying R package, Applied Econometrics with R. This has code for the textbook by Stock and Watson (2011), which is a fine text to get a basic acquaintance with time series econometrics.

References

Cowpertwait PSP, Metcalfe AV (2009) Introductory time series with R. Springer, Dordrecht
Hyndman RJ, Athanasopoulos G (2014) Forecasting: principles and practice. https://www.otexts.org/fpp. Accessed on 12 Sep 2014
Hyndman RJ, Athanasopoulos G, Razbash S, Schmidt D, Zhou Z, Khan Y, Bergmeir C, Wang E (2014) Forecast: forecasting functions for time series and linear models. R package version 5.5. http://CRAN.R-project.org/package=forecast
Kleiber C, Zeileis A (2008) Applied econometrics with R. Springer, Dordrecht
Ryan JA, Ulrich JM (2014) xts: eXtensible Time Series. R package version 0.9-7. http://CRAN.R-project.org/package=xts
Stock JH, Watson MW (2011) Introduction to econometrics. Addison-Wesley, Boston
Teetor P (2011) R Cookbook. O'Reilly Cookbooks