Using R for Introduction to Econometrics

Christoph Hanck, Martin Arnold, Alexander Gerber and Martin Schmelzer

2018-07-27

Contents

Chapter 1

Introduction

The interest in the freely available statistical programming language and software environment `R` (R Core Team, 2018) is soaring. By the time we wrote first drafts for this project, more than 11000 addons (many of them providing cutting-edge methods) were made available on the Comprehensive `R` Archive Network (CRAN), an extensive network of FTP servers around the world that store identical and up-to-date versions of `R` code and its documentation. `R` dominates other (commercial) software for statistical computing in most fields of research in applied statistics. The benefits of it being freely available, open source and having a large and constantly growing community of users that contribute to CRAN render `R` more and more appealing for empirical economists and econometricians as well.

A striking advantage of using `R` in econometrics courses is that it enables students to explicitly document their analysis step-by-step such that it is easy to update and to expand. This allows to re-use code for similar applications with different data. Furthermore, `R` programs are fully reproducible which makes it straightforward for others to comprehend and validate results.

Over the recent years, `R` has thus become an integral part of the curricula of econometrics classes we teach at University of Duisburg-Essen. In some sense, learning to code is comparable to learning a foreign language and continuous practice is essential for the learning success. Of course, presenting bare `R` code chunks

on slides has mostly a deterring effect for the students to engage with programming on their own. This is why we offer tutorials where both econometric theory and its applications using R are introduced, for some time now. As for accompanying literature, there are some excellent books that deal with R and its applications to econometrics like Kleiber and Zeileis (2008). However, we have found that these works are somewhat difficult to access, especially for undergraduate students in economics having little understanding of econometric methods and predominantly no experience in programming at all. Consequently, we have started to compile a collection of reproducible reports for use in class. These reports provide guidance on how to implement selected applications from the textbook *Introduction to Econometrics* (Stock and Watson, 2015) which serves as a basis for the lecture and the accompanying tutorials. The process is facilitated considerably by `knitr` (Xie, 2018) and `rmarkdown` (Allaire et al., 2018). In conjunction, both R packages provide powerful tools for dynamic report generation which allow to seamlessly combine pure text, LaTeX, R code and its output in a variety of formats, including PDF and HTML. Being inspired by *Using R for Introductory Econometrics* (Heiss, 2016)[1] and with this powerful toolkit at hand we decided to write up our own empirical companion to Stock and Watson (2015) which resulted in **U**sing **R** **f**or **I**ntroduction **t**o **E**conometrics (*URFITE*).

Similarly to the book by Heiss (2016) this project is neither a comprehensive econometrics textbook nor is it intended to be a general introduction R. *URFITE* is best described as an interactive script in the style of a reproducible research report which aims to provide students of economic sciences with a platform-independent e-learning arrangement by seamlessly intertwining theoretical core knowledge and empirical skills in undergraduate econometrics. Of course the focus is set on empirical applications with R; we leave out tedious derivations and formal proofs wherever we can. *URFITE* is closely aligned on Stock and Watson (2015) which does very well in motivating theory by real-world applications. However, we take it a step further and enable students not only to learn how results of case studies can be replicated with R but we also intend to strengthen their ability in using the newly acquired skills in other empirical applications — immediately within *URFITE*.

To realize this, each chapter contains interactive R programming exercises. These exercises are used as supplements to code chunks that display how previously discussed techniques can be implemented within R. They are generated using the DataCamp light widget and are backed by an R session which is maintained on DataCamp's servers. You may play around with the example exercise presented below.

This interactive application is only available in the HTML version.

As you can see above, the widget consists of two tabs. `script.R` mimics an `.R`-file, a file format that is commonly used for storing R code. Lines starting with a # are commented out, that is, they are not recognized as code. Furthermore, `script.R` works like an exercise sheet where you may write down the solution you come up with. If you hit the button *Run*, the code will be executed, submission correctness tests are run and you will be notified whether your approach is correct. If it is not correct, you will receive feedback suggesting improvements or hints. The other tab, `R Console`, is a fully functional R console that can be used for trying out solutions to exercises before submitting them. Of course you may submit (almost any) R code and use the console to play around and explore. Simply type a command and hit the enter key on your keyboard.

As an example, consider the following line of code presented in chunk below. It tells R to compute the number of packages available on `CRAN`. The code chunk is followed by the output produced.

```
# compute the number of packages available on CRAN
nrow(available.packages(repos = "http://cran.us.r-project.org"))
```

```
## [1] 12785
```

Each code chunk is equipped with a button on the outer right hand side which copies the code to your clipboard. This makes it convenient to work with larger code segments. In the widget above, you may click on `R Console` and type `nrow(available.packages())` (the command from the code chunk above)

[1] Heiss (2016) builds on the popular *Introductory Econometrics* by Wooldridge (2016) and demonstrates how to replicate the applications discussed therein using R.

and execute it by hitting *Enter* on your keyboard[2]

As you might have noticed, there are some out-commented lines in the widget that ask you to assign a numeric value to a variable and then to print the variable's content to the console. You may enter your solution approach to `script.R` and hit the button *Run* in order to get the feedback described further above. In case you do not know how to solve this sample exercise (don't panic, that is probably why you are reading this), a click on *Hint* will prompt you with some advice. If you still can't find a solution, a click on *solution* will provide you with another tab, `Solution.R` which contains sample solution code. It will often be the case that exercises can be solved in many different ways and `Solution.R` presents what we consider as comprehensible and idiomatic.

Conventions Used in this Book

- *Italic* text indicates new terms, names, buttons and alike.
- `Constant width text`, is generally used in paragraphs to refer to `R` code. This includes commands, variables, functions, data types, databases and file names.
- Constant width text on gray background is used to indicate `R` code that can be typed literally by you. It may appear in paragraphs for better distinguishability among executable and non-executable code statements but it will mostly be encountered in shape of large blocks of `R` code. These blocks are referred to as code chunks (see above).

Acknowledgements

We thank Alexander Blasberg and Kim Hermann for proofreading and their constructive criticism.

1.1 A Very Short Introduction to `R` and *RStudio*

`R` Basics

As mentioned before, this book is not intended to be an introduction to `R` but as a guide on how to use its capabilities for applications commonly encountered in undergraduate econometrics. Those having basic knowledge in `R` programming will feel comfortable starting with Chapter **??**. This section, however, is meant for those who have not worked with `R` or *RStudio* before. If you at least know how to create objects and call functions, you can skip it. If you would like to refresh your memories or get a feeling for how to work with *RStudio*, keep reading.

First of all start *RStudio* and create a new `R` script by selecting *File*, *New File*, *R Script*. In the editor pane, type

```
1 + 1
```

and click on the button labeled *Run* in the top right corner of the editor. By doing so, your line of code is send to the console and the result of this operation should be displayed right underneath it. As you can see, `R` works just like a calculator. You can do all the arithmetic calculations by using the corresponding operator (`+`, `-`, `*`, `/` or `^`). If you are not sure what the last operator does, try it out and check the results.

[2] The `R` session is initialized by clicking anywhere into the widget. This might take a few seconds. Just wait for the indicator next to the button *Run* to turn green.

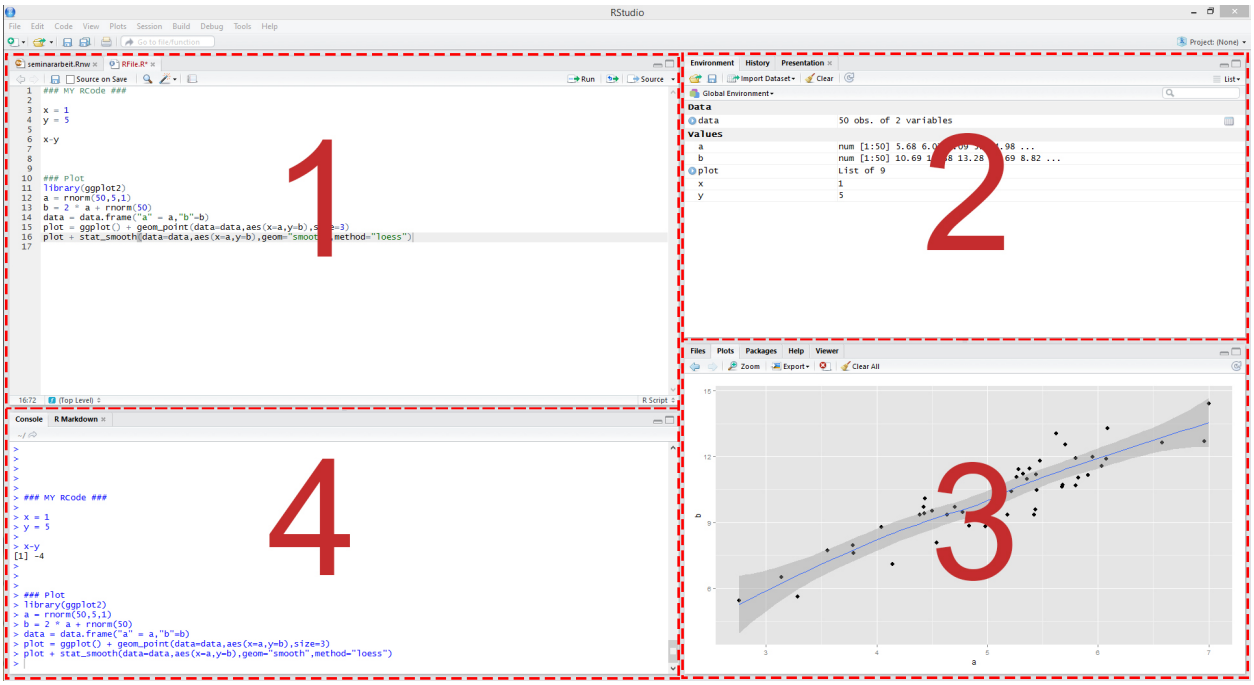


Figure 1.1: RStudio: the four panes

Vectors

`R` is of course more sophisticated than that. We can work with variables or more generally objects. Objects are defined by using the assignment operator `<-`. To create a variable named `x` which contains the value `10` type `x <- 10` and click the button *Run* yet again. The new variable should have appeared in the environment pane on the top right. The console however did not show any results, because our line of code did not contain any call that creates output. When you now type `x` in the console and hit return, you ask `R` to show you the value of `x` and the corresponding value should be printed in the console.

`x` is a scalar, a vector of length 1. You can easily create longer vectors by using the function `c()` (*c* for "concatenate" or "combine"). To create a vector `y` containing the numbers 1 to 5 and print it, do the following.

```
y <- c(1, 2, 3, 4, 5)
y
```

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

You can also create a vector of letters or words. For now just remember that characters have to be surrounded by quotes, else wise they will be parsed as object names.

```
hello <- c("Hello", "World")
```

Here we have created a vector of length 2 containing the words `Hello` and `World`.

Do not forget to save your script! To do so, select *File*, *Save*.

Functions

You have seen the function `c()` that can be used to combine objects. In general, function calls look all the same, a function name is always followed by round parentheses. Sometimes, the parentheses include

arguments

Here are two simple examples.

```
z <- seq(from = 1, to = 5, by = 1)

mean(x = z)
```

```
## [1] 3
```

In the first line we use a function called `seq()` to create the exact same vector as we did in the previous section but naming it `z`. The function takes on the arguments `from`, `to` and `by` which should be self-explaining. The function `mean()` computes the arithmetic mean of its argument `x`. Since we pass the vector `z` as the argument `x`, the result is `3`!

If you are not sure which arguments a function expects, you may consult the function's documentation. Let's say we are not sure how the arguments required for `seq()` work. Then we may type `?seq` in the console and by hitting return the documentation page for that function pops up in the lower right pane of *RStudio*. In there, the section *Arguments* holds the information we seek.

On the bottom of almost every help page you find examples on how to use the corresponding functions. This is very helpful for beginners and we recommend to look out for those.

Chapter 2

Hypothesis Tests and Confidence Intervals in Multiple Regression

This chapter discusses methods that allow to quantify the sampling uncertainty in the OLS estimator of the coefficients in multiple regression models. The basis for this are hypothesis tests and confidence intervals which, just as for the simple linear regression model, can be computed using basic `R` functions. We will also tackle the issue of testing joint hypotheses on these coefficients.

Make sure the packages `AER` (**?**) and `stargazer` (**?**) are installed before you go ahead and replicate the examples. The safest way to do so is by checking whether the following code chunk executes without any issues.

```
library(AER)
library(stargazer)
```

2.1 Hypothesis Tests and Confidence Intervals for a Single Coefficient

We first discuss how to compute standard errors, how to test hypotheses and how to construct confidence intervals for a single regression coefficient β_j in a multiple regression model. The basic idea is summarized in Key Concept 7.1.

Key Concept 7.1
Testing the Hypothesis $\beta_j = \beta_{j,0}$ Against the Alternative $\beta_j \neq \beta_{j,0}$

1. Compute the standard error of $\hat{\beta}_j$
2. Compute the t-statistic,
$$t^{act} = \frac{\hat{\beta}_j - \beta_{j,0}}{SE(\hat{\beta}_j)}$$
3. Compute the p-value,
$$p\text{-value} = 2\Phi(-|t^{act}|)$$
where t^{act} is the value of the t-statistic actually computed. Reject the hypothesis at the 5% significance level if the p-value is less than 0.05 or, equivalently, if $|t^{act}| > 1.96$.

The standard error and (typically) the t-statistic and the corresponding p-value for testing $\beta_j = 0$ are computed automatically by suitable `R` functions, e.g., by `summary()`.

Testing a single hypothesis about the significance of a coefficient in the multiple regression model proceeds as in in the simple regression model.

You can easily see this by inspecting the coefficient summary of the regression model

$$TestScore = \beta_0 + \beta_1 \times size\beta_2 \times english + u$$

already discussed in Chapter **??**. Let us review this:

```
model <- lm(score ~ size + english, data = CASchools)
coeftest(model, vcov. = vcovHC(model, type = "HC1"))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 686.032245   8.728225  78.5993  < 2e-16 ***
## size         -1.101296   0.432847  -2.5443  0.01131 *
## english      -0.649777   0.031032 -20.9391  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You may check that these quantities are computed as in the simple regression model by computing the t-statistics or p-values by hand using the output above and `R` as a calculator.

For example, using the definition of the p-value for a two-sided test as given in Key Concept 7.1, we can confirm the p-value for a test of the hypothesis that the coefficient β_1, the coefficient on `size`, to be approximately zero.

```
2 * (1 - pt(
            abs(
              coeftest(model, vcov. = vcovHC(model, type = "HC1"))[2, 3]),
            df = model$df.residual))
```

```
## [1] 0.01130921
```

Key Concept 7.2
Confidence Intervals for a Single Coefficient in Multiple Regression

A 95% two-sided confidence interval for the coefficient β_j is an interval that contains the true value of β_j with a 95% probability; that is, it contains the true value of β_j in 95% of repeated samples. Equivalently, it is the set of values of β_j that cannot be rejected by a 5% two-sided hypothesis test. When the sample size is large, the 95% confidence interval for β_j is

$$\left[\hat{\beta}_j - 1.96 \times SE(\hat{\beta}_j), \hat{\beta}_j + 1.96 \times SE(\hat{\beta}_j)\right].$$

2.2 An Application to Test Scores and the Student-Teacher Ratio

Let us take a look at the regression from Section **??** again.

Computing confidence intervals for individual coefficients in the multiple regression model proceeds as in the simple regression model using the function `confint()`.

```
model <- lm(score ~ size + english, data = CASchools)
confint(model)
```

```
##                   2.5 %      97.5 %
## (Intercept) 671.4640580 700.6004311
## size         -1.8487969  -0.3537944
## english      -0.7271113  -0.5724424
```

To obtain confidence intervals at another level, say 90%, just set the argument `level` in our call of `confint()` accordingly.

```
confint(model, level = 0.9)
```

```
##                    5 %        95 %
## (Intercept) 673.8145793 698.2499098
## size         -1.7281904  -0.4744009
## english      -0.7146336  -0.5849200
```

The output now reports the desired 90% confidence intervals for all coefficients.

A disadvantage of `confint()` is that is does not use robust standard errors to compute the confidence interval. For large-sample confidence intervals, this is quickly done manually as follows.

```
# compute robust standard errors
rob_se <- diag(vcovHC(model, type = "HC1"))^0.5

# compute robust 95% confidence intervals
rbind(
  "lower" = coef(model) - qnorm(0.975) * rob_se,
  "upper" = coef(model) + qnorm(0.975) * rob_se
)
```

```
##       (Intercept)       size    english
## lower    668.9252 -1.9496606 -0.7105980
## upper    703.1393 -0.2529307 -0.5889557
```

```
# compute robust 90% confidence intervals

rbind(
  "lower" = coef(model) - qnorm(0.95) * rob_se,
```

```
  "upper" = coef(model) + qnorm(0.95) * rob_se
)
```

```
##       (Intercept)       size    english
## lower    671.6756 -1.8132659 -0.7008195
## upper    700.3889 -0.3893254 -0.5987341
```

Knowing how to use `R` to make inference about the coefficients in multiple regression models, you can now answer the following question:

Can the null hypothesis that a change in the student-teacher ratio, `size`, has no significant influence on test scores, `scores`, — if we control for the percentage of students learning English in the district, `english`, — be rejected at the 10% and the 5% level of significance?

The output above shows that zero is not an element of the confidence interval for the coefficient on `size` such that we can reject the null hypothesis at significance levels of 5% and 10%. The same conclusion can be made via the p-value for `size`: $0.00398 < 0.05 = \alpha$.

Note that rejection at the 5%-level implies rejection at the 10% level (why?).

Recall from Chapter **??** the 95% confidence interval computed above *does not* tell us that a one-unit decrease in the student-teacher ratio has an effect on test scores that lies in the interval with a lower bound of -1.9497 and an upper bound of -0.2529. Once a confidence interval has been computed, a probabilistic statement like this is wrong: either the interval contains the true parameter or it does not. We do not know which is true.

Another Augmentation of the Model

What is the average effect on test scores of reducing the student-teacher ratio when the expenditures per pupil and the percentage of english learning pupils are held constant?

Let us augment our model by an additional regressor that is a measure for expenditure per pupil. Using `?CASchools` we find that `CASchools` contains the variable `expenditure`, which provides expenditure per student.

Our model now is

$$TestScore = \beta_0 + \beta_1 \times size + \beta_2 \times english + \beta_3 \times expenditure + u$$

with *expenditure* the total amount of expenditure per pupil in the district (thousands of dollars).

Let us now estimate the model:

```
# scale expenditure to thousands of dollars
CASchools$expenditure <- CASchools$expenditure/1000

# estimate the model
model <- lm(score ~ size + english + expenditure, data = CASchools)
coeftest(model, vcov. = vcovHC(model, type = "HC1"))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 649.577947  15.458344  42.0212  < 2e-16 ***
## size         -0.286399   0.482073  -0.5941  0.55277
## english      -0.656023   0.031784 -20.6398  < 2e-16 ***
## expenditure   3.867901   1.580722   2.4469  0.01482 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated effect of a one unit change in the student-teacher ratio on test scores with expenditure and the share of english learning pupils held constant is -0.29, which is rather small. What is more, the coefficient on *size* is not significantly different from zero anymore even at 10% since p-value $= 0.55$. Can you come up with an interpretation for these findings (see Chapter 7.1 of the book)? The insignificance of $\hat{\beta}_1$ could be due to a larger standard error of $\hat{\beta}_1$ resulting from adding *expenditure* to the model so that we estimate the coefficient on *size* less precisely. This illustrates the issue of strongly correlated regressors (imperfect multicollinearity). The correlation between *size* and *expenditure* can be computed using `cor()`.

```
# compute the sample correlation between 'size' and 'expenditure'
cor(CASchools$size, CASchools$expenditure)
```

```
## [1] -0.6199822
```

Altogether, we conclude that the new model provides no evidence that changing the student-teacher ratio, e.g., by hiring new teachers, has any effect on the test scores while keeping expenditures per student and the share of English learners constant.

2.3 Joint Hypothesis Testing Using the F-Statistic

The estimated model is

$$\widehat{TestScore} = \underset{(15.21)}{649.58} - \underset{(0.48)}{0.29} \times size - \underset{(0.04)}{0.66} \times english + \underset{(1.41)}{3.87} \times expenditure.$$

Now, can we reject the hypothesis that the coefficient on *size and* the coefficient on *expenditure* are zero? To answer this, we have to resort to joint hypothesis tests. A joint hypothesis imposes restrictions on multiple regression coefficients. This is different from conducting individual t-tests where a restriction is imposed on a single coefficient. Chapter 7.2 of the book explains why testing hypotheses about the model coefficients one at a time is different from testing them jointly.

The homoskedasticity-only F-Statistic is given by

$$F = \frac{(SSR_{restricted} - SSR_{unrestricted})/q}{SSR_{unrestricted}/(n-k-1)}$$

with $SSR_{restricted}$ being the sum of squared residuals from the restricted regression, i.e., the regression where we impose the restriction. $SSR_{unrestricted}$ is the sum of squared residuals from the full model, q is the number of restrictions under the null and k is the number of regressors in the unrestricted regression.

It is fairly easy to conduct F-tests in `R`. We can use the function `linearHypothesis()`contained in the package `car`.

```
# estimate the multiple regression model
model <- lm(score ~ size + english + expenditure, data = CASchools)

# execute the function on the model object and provide both linear restrictions
# to be tested as strings
linearHypothesis(model, c("size=0", "expenditure=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## size = 0
```

```
## expenditure = 0
##
## Model 1: restricted model
## Model 2: score ~ size + english + expenditure
##
##   Res.Df   RSS Df Sum of Sq      F   Pr(>F)
## 1    418 89000
## 2    416 85700  2    3300.3 8.0101 0.000386 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output reveals that the F-statistic for this joint hypothesis test is about 8.01 and the corresponding p-value is 0.0004. Thus, we can reject the null hypothesis that both coefficients are zero at any level of significance commonly used in practice.

A heteroskedasticity-robust version of this F-test (which leads to the same conclusion) can be conducted as follows.

```
# heteroskedasticity-robust F-test
linearHypothesis(model, c("size=0", "expenditure=0"), white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## size = 0
## expenditure = 0
##
## Model 1: restricted model
## Model 2: score ~ size + english + expenditure
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1    418
## 2    416  2 5.4337 0.004682 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The standard output of a model summary also reports an F-statistic and the corresponding p-value. The null hypothesis belonging to this F-test is that *all* of the population coefficients in the model except for the intercept are zero, so the hypotheses are

$$H_0 : \beta_1 = 0, \ \beta_2 = 0, \ \beta_3 = 0 \quad \text{vs.} \quad H_1 : \beta_j \neq 0 \text{ for at least one } j = 1, 2, 3.$$

This is also called the *overall regression F-statistic* and the null hypothesis is obviously different from testing if only β_1 and β_3 are zero.

We now check whether the F-statistic belonging to the p-value listed in the model's summary coincides with the result reported by `linearHypothesis()`.

```
# execute the function on the model object and provide the restrictions
# to be tested as a character vector
linearHypothesis(model, c("size=0", "english=0", "expenditure=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## size = 0
```

```
## english = 0
## expenditure = 0
##
## Model 1: restricted model
## Model 2: score ~ size + english + expenditure
##
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    419 152110
## 2    416  85700  3     66410 107.45 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Access the overall F-statistic from the model's summary
summary(model)$fstatistic
```

```
##    value    numdf    dendf
## 107.4547   3.0000 416.0000
```

The entry `value` is the overall F-statistics and it equals the result of `linearHypothesis()`. The F-test rejects the null hypothesis that the model has no power in explaining test scores. It is important to know that the F-statistic reported by `summary` is *not* robust to heteroskedasticity!

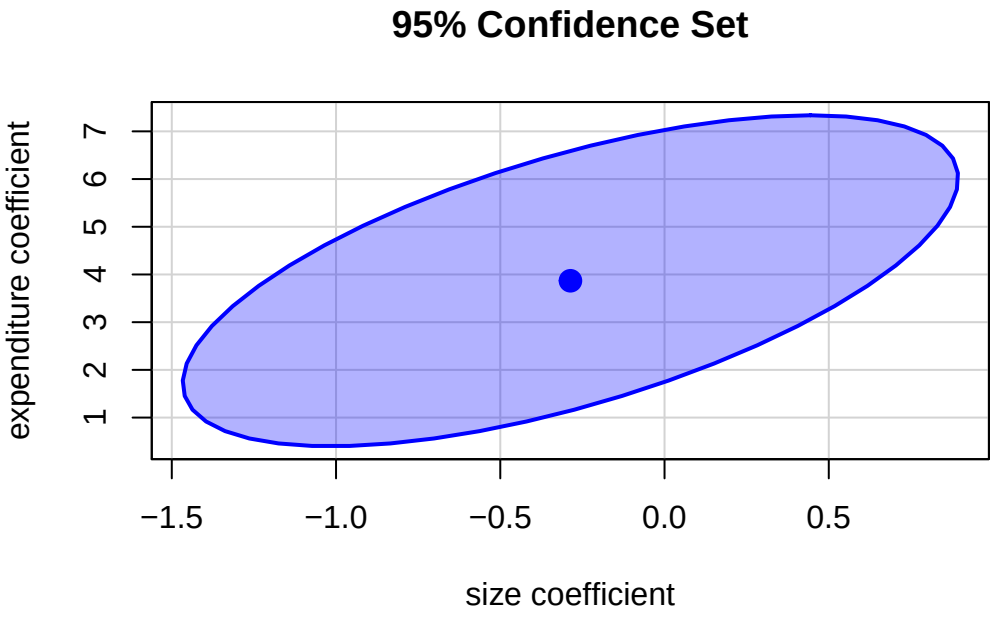
2.4 Confidence Sets for Multiple Coefficients

Based on the F-statistic that we have previously encountered, we can specify confidence sets. Confidence sets are analogous to confidence intervals for single coefficients. As such, confidence sets consist of *combinations* of coefficients that contain the true combination of coefficients in, say, 95% of all cases if we could repeatedly draw random samples, just like in the univariate case. Put differently, a confidence set is the set of all coefficient combinations for which we cannot reject the corresponding joint null hypothesis tested using an F-test.

The confidence set for two coefficients an ellipse which is centered around the point defined by both coefficient estimates. Again, there is a very convenient way to plot the confidence set for two coefficients of model objects, namely the function `confidenceEllipse()` from the `car` package.

We now plot the 95% confidence ellipse for the coefficients on `size` and `expenditure` from the regression conducted above. By specifying the additional argument `fill`, the confidence set is colored.

```
# Draw the 95% confidence set for coefficients on size and expenditure
confidenceEllipse(model,
                  fill = T,
                  which.coef = c("size", "expenditure"),
                  main = "95% Confidence Set")
```

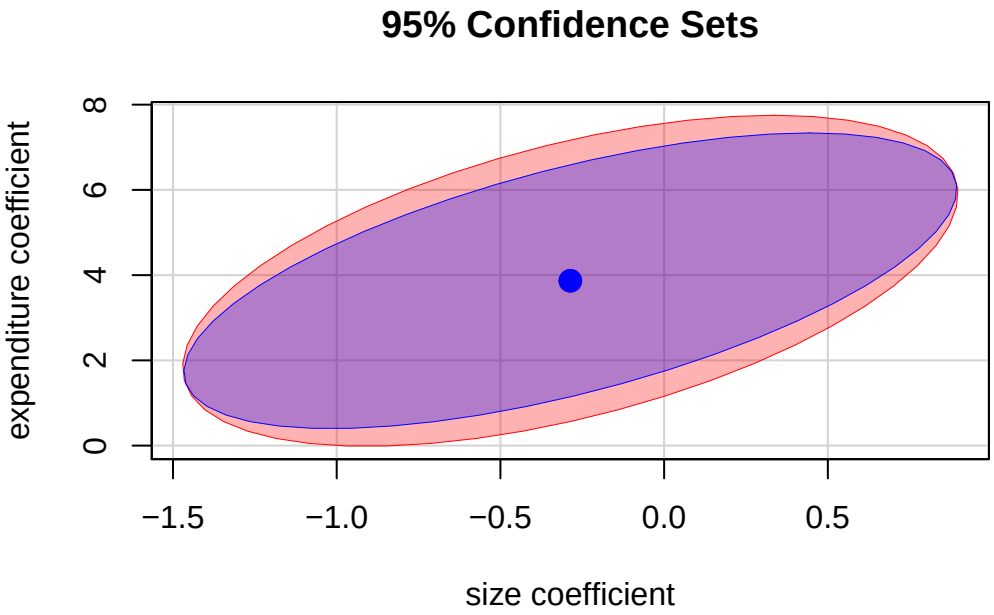


We see that the ellipse is centered around $(-0.29, 3.87)$, the pair of coefficients estimates on *size* and *expenditure*. What is more, $(0, 0)$ is not element of the 95% confidence set so that we can reject H_0 : $\beta_1 = 0,\ \beta_3 = 0$.

By default, `confidenceEllipse()` uses homoskedasticity-only standard errors. The following code chunk shows how compute a robust confidence ellipse and how to overlay it with the previous plot.

```
# Draw the 95% confidence set for coefficients on size and expenditure (robust)
confidenceEllipse(model,
                  fill = T,
                  lwd = 0,
                  which.coef = c("size", "expenditure"),
                  main = "95% Confidence Sets",
                  vcov. = vcovHC(model, type = "HC1"),
                  col = "red")

# Draw the 95% confidence set for coefficients on size and expenditure
confidenceEllipse(model,
                  fill = T,
                  lwd = 0,
                  which.coef = c("size", "expenditure"),
                  add = T)
```



As the robust standard errors are slightly larger than those valid under homoskedasticity only in this case, the robust confidence set is slightly larger. This is analogous to the confidence intervals for the individual coefficients.

2.5 Model Specification for Multiple Regression

Choosing a regression specification, i.e. selecting, the variables to be included in a regression model, is a difficult task. However, there are some guidelines on how to proceed. The goal is clear: obtaining an unbiased and precise estimate of the causal effect of interest. As a starting point, think about omitted variables, that is, to avoid possible bias by using suitable control variables. Omitted variables bias in the context of multiple regression is explained in Key Concept 7.3. A second step could be to compare different specifications by measures of fit. However, as we shall see one should not rely solely on $\overline{R^2}$.

Key Concept 7.3
Omitted Variable Bias in Multiple Regression

Omitted variable bias is the bias in the OLS estimator that arises when regressors correlate with an omitted variable. For omitted variable bias to arise, two things must be true:

1. At least one of the included regressors must be correlated with the omitted variable.
2. The omitted variable must be a determinant of the dependent variable, Y.

We now discuss an example were we face a potential omitted variable bias in a multiple regression model:

Consider again the estimated regression equation

$$\widehat{TestScore} = \underset{(8.7)}{686.0} - \underset{(0.43)}{1.10} \times size - \underset{(0.031)}{0.650} \times english.$$

We are interested in estimating the causal effect of class size on test score. There might be a bias due to omitting "outside learning opportunities" from our regression since such a measure could be a determinant of the students' test scores and could also be correlated with both regressors already included in the model (so

that both conditions of Key Concept 7.3 are fulfilled). "Outside learning opportunities" are a complicated concept that is difficult to quantify. A surrogate we can consider instead is the students' economic background which likely are strongly related to outside learning opportunities: think of wealthy parents that are able to provide time and/or money for private tuition of their children. We thus augment the model with the variable `lunch`, the percentage of students that qualify for a free or subsidized lunch in school due to family incomes below a certain threshold, and reestimate the model.

```
# estimate the model and print the summary to console
model <- lm(score ~ size + english + lunch, data = CASchools)
coeftest(model, vcov. = vcovHC(model, type = "HC1"))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 700.149957   5.568453 125.7351 < 2.2e-16 ***
## size         -0.998309   0.270080  -3.6963 0.0002480 ***
## english      -0.121573   0.032832  -3.7029 0.0002418 ***
## lunch        -0.547345   0.024107 -22.7046 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, the estimated regression line is

$$\widehat{TestScore} = \underset{(5.56)}{700.15} - \underset{(0.27)}{1.00} \times size - \underset{(0.03)}{0.12} \times english - \underset{(0.02)}{0.55} \times lunch.$$

We observe no substantial changes in the conclusion about the effect of *size* on *TestScore*: the coefficient on *size* changes by only 0.1 and retains its significance.

Although the difference in estimated coefficients is not big in this case, it is useful to keep `lunch` to make the assumption of conditional mean independence more credible (see Chapter 7.5 of the book).

Model Specification in Theory and in Practice

Key Concept 7.4 lists some common pitfalls when using R^2 and $\overline{R^2}$ to evaluate the predictive ability of regression models.

Key Concept 7.4
R^2 and $\overline{R^2}$: What They Tell You — and What They Do not

The R^2 and $\overline{R^2}$ tell you whether the regressors are good at explaining the variation of the independent variable in the sample. If the R^2 (or $\overline{R^2}$) is nearly 1, then the regressors produce a good prediction of the dependent variable in that sample, in the sense that the variance of OLS residuals is small compared to the variance of the dependent variable. If the R^2 (or $\overline{R^2}$) is nearly 0, the opposite is true.

The R^2 and $\overline{R^2}$ do *not* tell you whether:

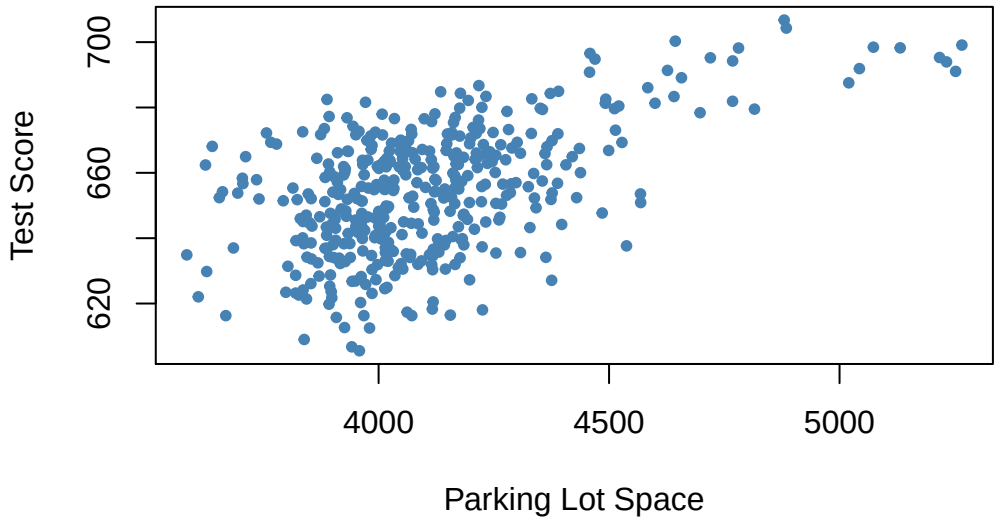
1. An included variable is statistically significant.
2. The regressors are the true cause of the movements in the dependent variable.
3. There is omitted variable bias.
4. You have chosen the most appropriate set of regressors.

For example, think of regressing $TestScore$ on PLS which measures the available parking lot space in thousand square feet. You are likely to observe a significant coefficient of reasonable magnitude and moderate to high values for R^2 and $\overline{R^2}$. The reason for this is that parking lot space is correlated with many determinants of the test score like location, class size, financial endowment and so on. Although we do not have observations on PLS, we can use R to generate some relatively realistic data.

```
# set seed for reproducibility
set.seed(1)

# generate observations for parking lot space
CASchools$PLS <- c(22 * CASchools$income
                   - 15 * CASchools$size
                   + 0.2 * CASchools$expenditure
                   + rnorm(nrow(CASchools), sd = 80) + 3000)
```

```
# plot parking lot space against test score
plot(CASchools$PLS,
     CASchools$score,
     xlab = "Parking Lot Space",
     ylab = "Test Score",
     pch = 20,
     col = "steelblue")
```




```
# regress test score on PLS
summary(lm(score ~ PLS, data = CASchools))
```

```
##
## Call:
## lm(formula = score ~ PLS, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.608 -11.049   0.342  12.558  37.105
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.897e+02  1.227e+01   39.90   <2e-16 ***
## PLS         4.002e-02  2.981e-03   13.43   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.95 on 418 degrees of freedom
## Multiple R-squared:  0.3013, Adjusted R-squared:  0.2996
## F-statistic: 180.2 on 1 and 418 DF,  p-value: < 2.2e-16
```

PLS is generated as a linear function of *expenditure*, *income*, *size* and a random disturbance. Therefore the data suggest that there is some positive relationship between parking lot space and test score. In fact, when estimating the model

$$TestScore = \beta_0 + \beta_1 \times PLS + u \tag{2.1}$$

using `lm()` we find that the coefficient on PLS is positive and significantly different from zero. Also R^2 and $\overline{R^2}$ are about 0.3 which is a lot more than the roughly 0.05 observed when regressing the test scores on the class sizes only. This suggests that increasing the parking lot space boosts a school's test scores and that model (2.1) does even better in explaining heterogeneity in the dependent variable than a model with *size* as the only regressor. Keeping in mind how PLS is constructed this comes as no surprise. It is evident that the high R^2 cannot be used to the conclude that the estimated relation between parking lot space and test scores is causal: the (relatively) high R^2 is due to correlation between PLS and other determinants and/or control variables. Increasing parking lot space is *not* an appropriate measure to generate more learning success!

2.6 Analysis of the Test Score Data Set

Chapter **??** and some of the previous sections have stressed that it is important to include control variables in regression models if it is plausible that there are omitted factors. In our example of test scores we want to estimate the causal effect of a change in the student-teacher ratio on test scores. We now provide an example how to use multiple regression in order to alleviate omitted variable bias and demonstrate how to report results using `R`.

So far we have considered two variables that control for unobservable student characteristics which correlate with the student-teacher ratio *and* are assumed to have an impact on test scores:

- *English*, the percentage of English learning students
- *lunch*, the share of students that qualify for a subsidized or even a free lunch at school

Another new variable provided with `CASchools` is `calworks`, the percentage of students that qualify for the *CalWorks* income assistance program. Students eligible for *CalWorks* live in families with a total income below the threshold for the subsidized lunch program so both variables are indicators for the share of economically disadvantaged children. Both indicators are highly correlated:

```
# estimate the correlation between 'calworks' and 'lunch'
cor(CASchools$calworks, CASchools$lunch)
```

```
## [1] 0.7394218
```

There is no unambiguous way to proceed when deciding which variable to use. In any case it may not a good idea to use both variables as regressors in view of collinearity. Therefore, we also consider alternative model specifications.

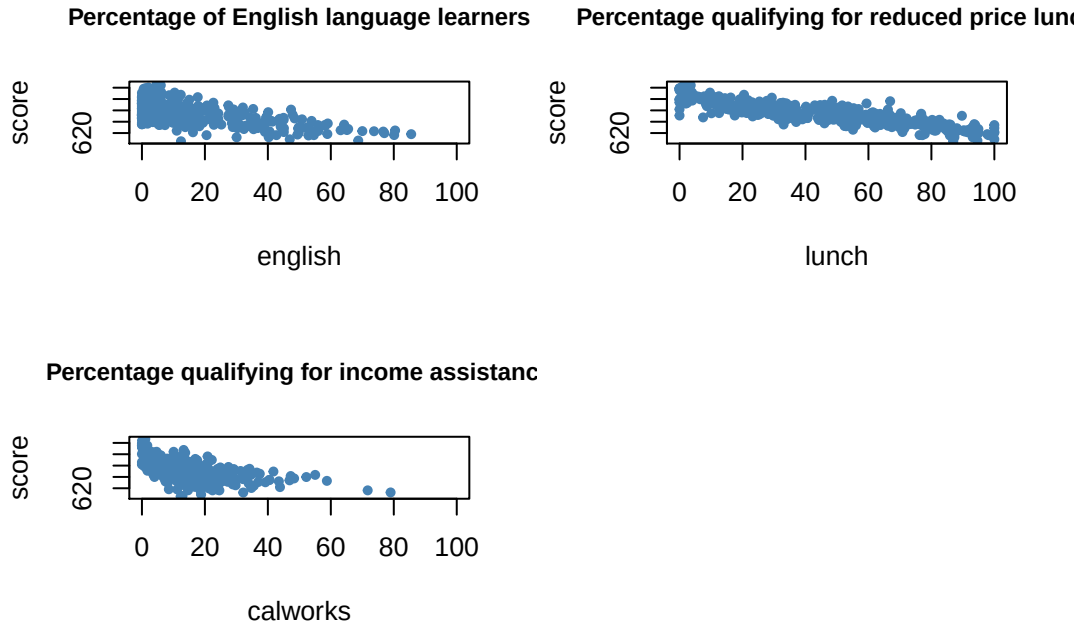
For a start, we plot student characteristics against test scores.

```
# set up arrangement of plots
m <- rbind(c(1, 2), c(3, 0))
layout(mat = m)
```

```
# scatterplots
plot(score ~ english,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     xlim = c(0, 100),
     cex.main = 0.9,
     main = "Percentage of English language learners")

plot(score ~ lunch,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     cex.main = 0.9,
     main = "Percentage qualifying for reduced price lunch")

plot(score ~ calworks,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     xlim = c(0, 100),
     cex.main = 0.9,
     main = "Percentage qualifying for income assistance")
```



We divide the plotting area up using `layout()`. The matrix `m` specifies the locaion of the plots, see `?layout`.

We see that all relationships are negative. Here are the correlation coefficients.

```
# estimate correlation between student characteristics and test scores
cor(CASchools$score, CASchools$english)
```

```
## [1] -0.6441238
```

```
cor(CASchools$score, CASchools$lunch)
```

```
## [1] -0.868772
```

```
cor(CASchools$score, CASchools$calworks)
```

```
## [1] -0.6268533
```

We shall consider five different model equations:

$$
\begin{aligned}
(I) \quad & TestScore = \beta_0 + \beta_1 \times size + u, \\
(II) \quad & TestScore = \beta_0 + \beta_1 \times size + \beta_2 \times english + u, \\
(III) \quad & TestScore = \beta_0 + \beta_1 \times size + \beta_2 \times english + \beta_3 \times lunch + u, \\
(IV) \quad & TestScore = \beta_0 + \beta_1 \times size + \beta_2 \times english + \beta_4 \times calworks + u, \\
(V) \quad & TestScore = \beta_0 + \beta_1 \times size + \beta_2 \times english + \beta_3 \times lunch + \beta_4 \times calworks + u
\end{aligned}
$$

The best way to communicate regression results is in a table. The `stargazer` package is very convenient for this purpose. It provides a function that generates professionally looking HTML and LaTeX tables that satisfy scientific standards. One simply has to provide one or multiple object(s) of class `lm`. The rest is done by the function `stargazer()`.

```
# load the stargazer library
library(stargazer)

# estimate different model specifications
spec1 <- lm(score ~ size, data = CASchools)
spec2 <- lm(score ~ size + english, data = CASchools)
spec3 <- lm(score ~ size + english + lunch, data = CASchools)
spec4 <- lm(score ~ size + english + calworks, data = CASchools)
spec5 <- lm(score ~ size + english + lunch + calworks, data = CASchools)

# gather robust standard errors in a list
rob_se <- list(
  sqrt(diag(vcovHC(spec1, type = "HC1"))),
  sqrt(diag(vcovHC(spec2, type = "HC1"))),
  sqrt(diag(vcovHC(spec3, type = "HC1"))),
  sqrt(diag(vcovHC(spec4, type = "HC1"))),
  sqrt(diag(vcovHC(spec5, type = "HC1")))
)

# generate a LaTeX table using stargazer
stargazer(spec1, spec2, spec3, spec4, spec5,
          se = rob_se,
          digits = 3,
          header = F,
          column.labels = c("(I)", "(II)", "(III)", "(IV)", "(V)")
          )
```

Table 2.1 states that *scores* is the dependent variable and that we consider five models. We see that the columns of Table 2.1 contain most of the information provided by `coeftest()` and `summary()` for the regression models under consideration: the coefficients estimates equipped with significance codes (the asterisks) and standard errors in parentheses below. Although there are no t-statistics, it is straightforward for the reader to compute them simply by dividing a coefficient estimate by the corresponding standard error. The bottom of the table reports summary statistics for each model and a legend. For an in-depth discussion of the tabular presentation of regression results, see Chapter 7.6 of the book.

What can we conclude from the model comparison?

Table 2.1: Regressions of Test Scores on the Student-Teacher Ratio and Control Variables

	Dependent Variable: Test Score				
			score		
	(I)	(II)	(III)	(IV)	(V)
	spec1	spec2	spec3	spec4	spec5
size	-2.280^{***}	-1.101^{**}	-0.998^{***}	-1.308^{***}	-1.014^{***}
	(0.519)	(0.433)	(0.270)	(0.339)	(0.269)
english		-0.650^{***}	-0.122^{***}	-0.488^{***}	-0.130^{***}
		(0.031)	(0.033)	(0.030)	(0.036)
lunch			-0.547^{***}		-0.529^{***}
			(0.024)		(0.038)
calworks				-0.790^{***}	-0.048
				(0.068)	(0.059)
Constant	698.933^{***}	686.032^{***}	700.150^{***}	697.999^{***}	700.392^{***}
	(10.364)	(8.728)	(5.568)	(6.920)	(5.537)
Observations	420	420	420	420	420
R^2	0.051	0.426	0.775	0.629	0.775
Adjusted R^2	0.049	0.424	0.773	0.626	0.773
Residual Std. Error	18.581 (df = 418)	14.464 (df = 417)	9.080 (df = 416)	11.654 (df = 416)	9.084 (df = 415)
F Statistic	22.575^{***} (df = 1; 418)	155.014^{***} (df = 2; 417)	476.306^{***} (df = 3; 416)	234.638^{***} (df = 3; 416)	357.054^{***} (df = 4; 415)

Note: *p<0.1; **p<0.05; ***p<0.01

1. We see that adding control variables roughly halves the coefficient on `size`. Also, the estimate is not very sensitive to the set of control variables used. The conclusion is that decreasing the student-teacher ratio ceteris paribus by one unit leads to an estimated average increase in test scores of about 1 point.
2. Adding student characteristics as controls boosts R^2 and $\overline{R^2}$ from 0.049 (`spec1`) up to 0.773 (`spec3` and `spec5`), so we can consider these variables as suitable predictors for test scores. Moreover, the estimated coefficients on all control variables are consistent with the impressions gained from Figure 7.2 of the book.
3. We see that the control variables are not statistically significant in all models. For example in `spec5` we see that the coefficient on *calworks* is not significantly different from zero at the level of 5% since $|-0.048/0.059| = 0.81 < 1.64$. We also observe that the effect on the estimate (and its standard error) of the coefficient on *size* of adding *calworks* to the base specification `spec3` is negligible. We can therefore consider `calworks` as a redundant control variable in this model.

2.7 Exercises

This interactive part of URFITE is only available in the HTML version.

Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., and Chang, W. (2018). *rmarkdown: Dynamic Documents for R*. R package version 1.9.

Heiss, F. (2016). *Using R for Introductory Econometrics*. CreateSpace Independent Publishing Platform.

Kleiber, C. and Zeileis, A. (2008). *Applied econometrics with R*. Springer.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Stock, J. and Watson, M. (2015). *Introduction to Econometrics, Third Update, Global Edition*. Pearson Education Limited.

Wooldridge, J. (2016). *Introductory econometrics*. Cengage Learning, sixth edition.

Xie, Y. (2018). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.20.