Using R for Introduction to Econometrics

Christoph Hanck, Martin Arnold, Alexander Gerber and Martin Schmelzer

2018-04-26

Contents

Chapter 1

Preface

The interest in the freely available statistical programming language and software environment R is soaring. By the time we wrote first drafts for this project, more than 11000 add-ons (many of them cutting-edge) were made available on the Comprehensive R Archieve 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 but 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 at the same time.

A striking advantage of using R in econometrics courses is that it allows 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. More important for the purpose of education, R programs are fully reproducible which makes it straightforward for others to comprehend and validate the results.

Presenting bare R code chunks on lecture slides clearly has a deterring effect which is why we offer tutorials where we introduce students to both econometric theory and its applications using R for some time now. There are some excelent books that introduce R and its applications to econometrics like Kleiber & Zeilis (2008) and Hetekar (2010). However, we think that these works are somewhat difficult to access for under-

graduate 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 & Watson (2014) on which the lecture is based, using R. This process has been faciliated considerably with the release of knitr, an R package for dynamic report generation which allows to seamlessly combine pure text, Latex and R code in a variety of output formats, including PDF and HTML.

This book is an interactive script in the style of a reproducible research report. **U**sing **R** **f**or **I**ntroduction **t**o **E**conometrics (*URFITE*) aims to provide students of economic sciences with a platform-independent e-learning arrangement that seamlessly intertwines theoretical core knowledge and empirical skills in undergraduate econometrics. Thereby, the focus is set on empirical applications with R and we leave out tedious derivations and formal proofs wherever we can. We were inspired by *Using R for Introductory Econometrics* (Heiss, 2016) which builds on the popular *Introductory Econometrics* by Wooldridge (2015) and domonstrates how to replicate the applications discussed therin using R.

Similarly, *URFITE* is closely aligned on Stock & Watson (2014). While the book does very well in motivating theory by real-world applications, we want to take it a step further and would like to 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.

Put differently, this book is neither an introduction to R, nor is it intended to be a replacement for the textbook let alone a full course in undergraduate econometrics. Instead, we would like to encourage you to work through chapters of Stock and Watson (2014) and use *URFITE* as an empirically orientated companion to the book.

Throughout this script, interactive R programming exercises are used as supplements to code chunks that display how previously discussed techniques can be implemented within R. These programming exercises are realized using the DataCamp light widget and are backed by an R-session which is maintained on DataCamp's servers.

The widget consists of two tabs. script.R mimics a .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, submission correctness tests are run and you will be notified whether your approach is correct. If not, 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) arbitrary 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, the line of code presented in the code chunk below tells R to compute the number of packages available on CRAN.

```
nrow(available.packages())
```

You may click on R Console, enter and execute this command (the R session is initialized by clicking anywhere into the widget. This might take a few seconds.)

```
<code data-type="sample-code">
    # Create a variable a, equal to 5

    # Print out a

    # check the number of R packages available on CRAN

</code>

<code data-type="solution">
```

```
    # Create a variable a, equal to 5
    a <- 5

    # Print out a
    print(a)
</code>

<code data-type="sct">
    test_object("a")
    test_function("print")
    test_function("nrow")
    test_function("available.packages")
    success_msg("Great job!")
</code>

<div data-type="hint">
Use the assignment operator (<code> <- </code>) to create the variable <code>a</code>.
</div>
```

Chapter 2

Introduction

Chapter 3

Probability Theory

This chapter reviews some basic concepts of probability theory and demonstrates how they can be applied in R.

Most of the statistical functionalities in R's standard version are collected in the stats package. It provides simple functions which compute descriptive measures and faciliate calculus involving a variety of probability distributions but also holds more sophisticated routines that e.g. enable the user to estimate a large number of models based on the same data or help to conduct extensive simulation studies. Execute `library(help = "stats")` in the console to view the documentation and a complete list of all functions gathered in `stats`.

In what follows, we lay our focus on (some of) the probability distributions that are handled by R and show how to use the relevant functions to solve simple problems. Thereby we will repeat some core concepts of probability theory. Among other things, You will learn how to draw random numbers, how to compute densities, probabilities, quantiles and alike. As we shall see, it is very convenient to rely on these routines, especially when writing Your own functions.

3.1 Random Variables and Probability Distributions

For a start, let us briefly review some basic concepts in probability.

- The mutually exclusive results of a random process are called the *outcomes*. 'Mutually exclusive' means that only one of the possible outcomes is observed.
- We refer to the *probability* of an outcome as the proportion of the time that the outcome occurs in the long run, that is if the experiment is repeated very often.
- The set of all possible outcomes of a random variable is called the *sample space*.
- An *event* is a subset of the sample space and consists of one or more outcomes.

These indeas are unified in the concept of a *random variable* which is a numerical summary of random outcomes. Random variables can be *discrete* or *continuous*.

- Discrete random variables have discrete outcomes, e.g. 0 and 1.
- A continuous random variable takes on a continuum of possible values.

Probability Distributions of Discrete Random Variables

A typical example for a discrete random variable D is the result of a die roll: in terms of a random experiment this is nothing but randomly selecting a sample of size 1 from a set of numbers which are mutually exclusive outcomes. Here, the sample space is $\{1, 2, 3, 4, 5, 6\}$ and we can think of many different events, e.g. 'the observed outcome lies between 2 and 5'.

A basic function to draw random samples from a specified set of elements is the the function `sample()`, see `?sample`. We can use it to simulate the random outcome of a die roll. Let's role the die!

```
sample(1:6, 1)
```

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

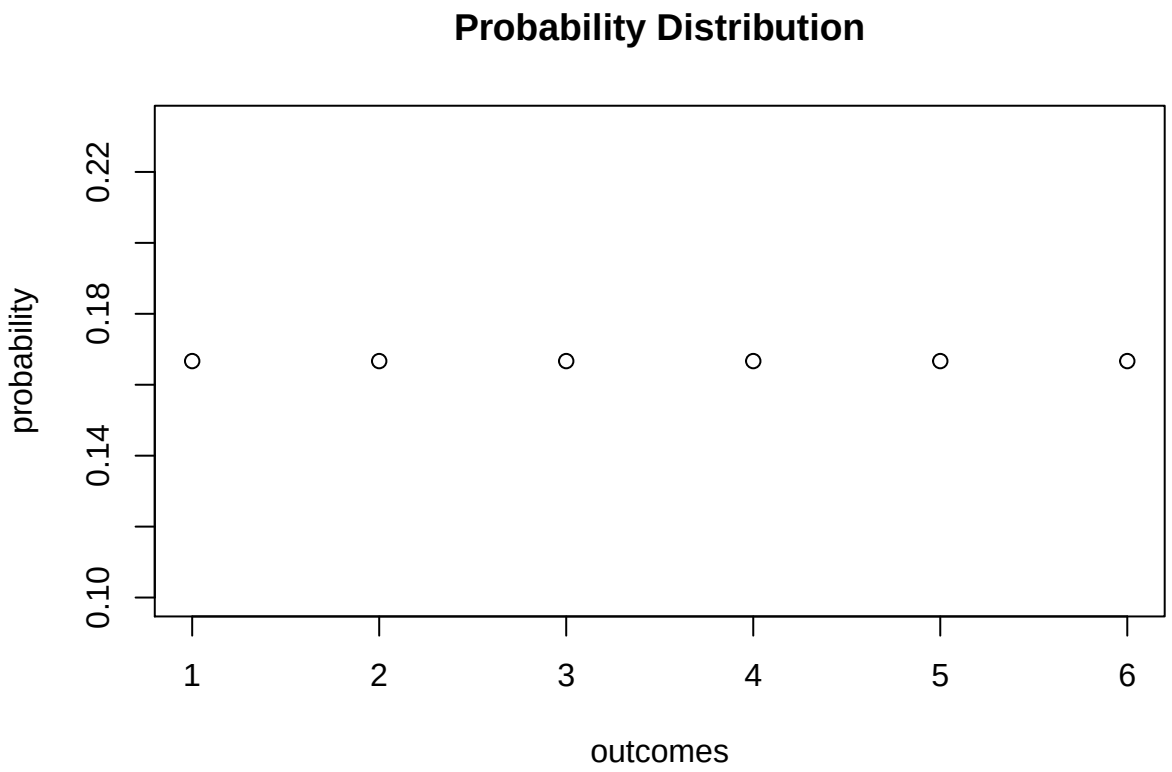
The probability distribution of a discrete random variable is the list of all possible values of the variable and thier probabilities which sum to 1. The cumulative probability distribution function states the probability that the random variable is less than or equal to a particular value.

For the die roll, this is straightforward to set up

Outcome	1	2	3	4	5	6
Probability distribution	1/6	1/6	1/6	1/6	1/6	1/6
Cumulative probability distribution	1/6	2/6	3/6	4/6	5/6	1

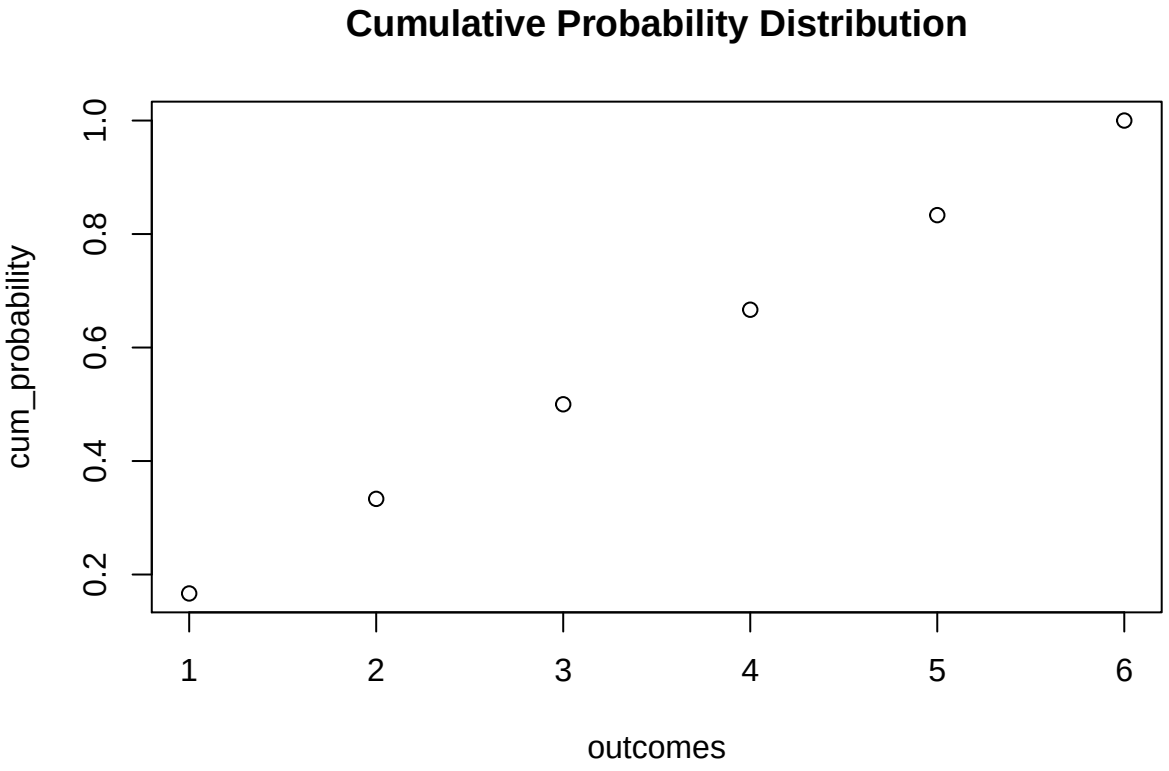
We can easily plot both functions using R. Since the probability equals 1/6 for each outcome, we set up the vector `probability` by using the `rep()` function which replicates a given value a specified number of times.

```
# generate the vector of probabilities
probability <- rep(1/6,6)

# plot the probabilites
plot(probability, xlab = "outcomes",
     main = "Probability Distribution"
     )
```



For the cumulative probability distribution we need the cumulative probabilities i.e. we need the cumulative sums of the vector `probability`. These sums can be computed using `cumsum()`.

```
#generate the vector of cumulative probabilities
cum_probability <- cumsum(probability)

# plot the probabilites
plot(cum_probability,
     xlab = "outcomes",
     main = "Cumulative Probability Distribution"
     )
```



Bernoulli Trials

The set of elements `sample()` draws from does not have to consist of numbers only. We might as well simulate coin tossing with outcomes H (head) and T (tail).

```
sample(c("H","T"),1)
```

```
## [1] "T"
```

The result of a coin toss is a Bernoulli distributed random variable i.e. a variable with to possible distinct outcomes.

Imagine you are about to toss a coin 10 times in a row and wonder how likely it is to end up with a sequence of outcomes like

$$H\ H\ T\ T\ T\ H\ T\ T\ H\ H.$$

This is a typical example of a Bernoulli experiment as it consists of $n = 10$ Bernoulli trials that are independent of each other and we are interested in the likelihood of observing $k = 5$ successes H that occur with probability $p = 0.5$ (assuming a fair coin) in each trial.

It is a well known result that the number of successes k follows a binomial distribution

$$k \sim B(n, p).$$

The probability of observing k successes in the experiment $B(n, p)$ is hence given by

$$f(k) = P(k) = \binom{n}{k} \cdot p^k \cdot q^{n-k} = \frac{n!}{k!(n-k)!} \cdot p^k \cdot q^{n-k}$$

where $\binom{n}{k}$ is a binomial coefficient.

In R, we can solve the problem stated above by means of the function `dbinom()` which calculates the probability of the binomial distribution for parameters `x`, `size`, and `prob`, see `?binom`.

```
dbinom(x = 5,
       size = 10,
       prob = 0.5
       )
```

```
## [1] 0.2460938
```

We conclude that the probability of observing Head $k = 5$ times when tossing the coin $n = 10$ times is about 24.6%.

Now assume we are interested in $P(4 \leq k \leq 7)$ i.e. the probability of observing 4, 5, 6 or 7 successes for $B(10, 0.5)$. This is easily computed by providing a vector as the `x` argument in our call of `dbinom()` and summing up using `sum()`.

```
sum(
  dbinom(x = 4:7,
         size = 10,
         prob = 0.5
         )
  )
```

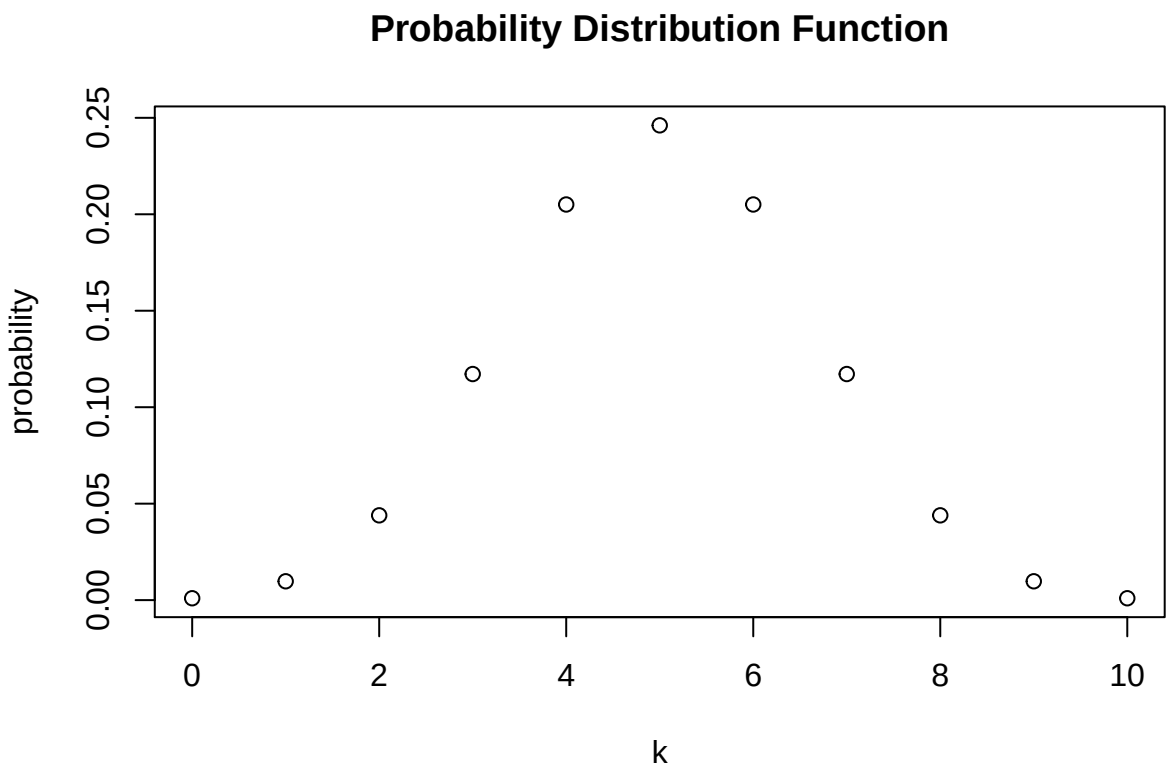
```
## [1] 0.7734375
```

The Probability distribution of a discrete random variable is nothing but a list of all possible outcomes that can occur and their respective probabilities. In our coin tossing example, we face 11 possible outcomes for k

```
# set up vector of possible outcomes
k <- 0:10
```

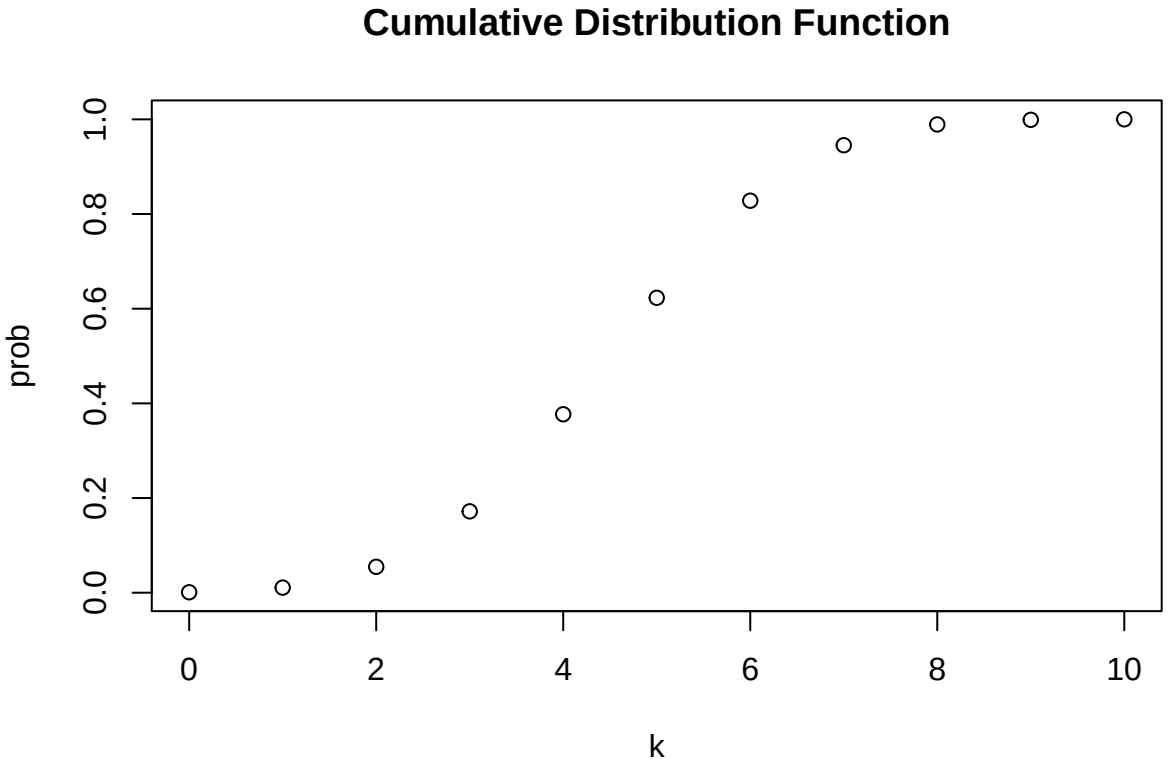
To visualize the probability distribution function of k we may therefore simply call

```
# assign probabilities
probability <- dbinom(x = k,
                      size = 10,
                      prob = 0.5
                      )

# plot outcomes against probabilities
plot(x = k,
     y = probability,
     main = "Probability Distribution Function")
```



In a similar fashion we may plot the cumulative distribution function of k by executing the following code chunk:

```
prob <- cumsum(
              dbinom(x =0:10,
                     size = 10,
                     prob = 0.5
                     )
              )

k <- 0:10
plot(x = k,
     y = prob,
     main = "Cumulative Distribution Function")
```



Expected Values, Mean and Variance

The expected value of a random variable is the long-run average value of the random variable over many repeated trials. For a discrete random variable, the expected value is computed as a weighted average of its possible outcomes whereby the weights are the related probabilities. This is formalized in Key Concept 2.1.

Key Concept 2.1

Expected Value and the Mean

Suppose the random variable Y takes on k possible values, $y_1, \ldots, y_k$, where y_1 denotes the first value, y_2 denotes the second value, and so forth, and that the probability that Y takes on y_1 is p_1, the probability that Y takes on y_2 is p_2 and so forth. The expected value of Y, $E(Y)$ is defined as

$$E(Y) = y_1 p_1 + y_2 p_2 + \cdots + y_k p_k = \sum_{i=1}^{k} y_i p_i$$

where the notation $\sum_{i=1}^{k} y_i p_i$ means "the sum of y_i p_i for i running from 1 to k". The expected value of Y is also called the mean of Y or the expectation of Y and is denoted by μ_y.

In the die example, the random variable, D say, takes on 6 possible values $d_1 = 1, d_2 = 2, \ldots, d_6 = 6$. Assuming a fair die, each of the 6 outcomes occurs with a probability of 1/6. It is therefore easy to calculate the exact value of $E(D)$ by hand:

$$E(D) = 1/6 \sum_{i=1}^{6} d_i = 3.5$$

Here, this is simply the average of the natural numbers from 1 to 6 since all wights p_i are 1/6. Convince Yourself that this can be easily calculated using the function `mean()` which computes the arithmetic mean of a numeric vector.

```
mean(1:6)
```

```
## [1] 3.5
```

An example of sampling with replacement is rolling a die three times in a row.

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

# rolling a die three times in a row
sample(1:6, 3, replace = T)
```

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

Of course we could also consider a much bigger number of trials, 10000 say. Doing so, it would be pointless to simply print the results to the console: by default R displays up to 1000 entries of large vectors and omitts the remainder (give it a go). Eyeballing the numbers does not reveal too much. Instead let us calculate the sample average of the outcomes using `mean()` and see if the result comes close to the expected value $E(D) = 3.5$.

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

# compute the sample mean of 10000 die rolls
mean(
    sample(1:6,
           10000,
           replace = T
           )
    )
```

```
## [1] 3.5039
```

We find the sample mean to be fairly close to the expected value. (ref to WLLN)

Other frequently encountered measures are the variance and the standard deviation. Both are measures of the *dispersion* of a random variable.

Key Concept 2.2

Variance and Standard Deviation

The Variance of the discrete *random variable* Y, denoted σ_Y^2, is

$$\sigma_Y^2 = \text{Var}(Y) = E\left[(Y - \mu_y)^2\right] = \sum_{i=1}^{k}(y_i - \mu_y)^2 p_i$$

The standard deviation of Y is σ_Y, the square root of the variance. The units of the standard deviation are the same as the units of Y.

The variance as defined in Key Concept 2.2 *is not* implemented as a function in R. Instead we have the function `var()` which computes the *sample variance*

$$s_Y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2.$$

Remember that s_Y^2 is different from the so called *population variance* of Y,

$$\text{Var}(Y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_Y)^2,$$

since it measures how the data is dispersed around the sample average $\overline{y}$ instead of the population mean μ_Y. This becomes clear when we look at our die rolling example. For D we have

$$\text{Var}(D) = 1/6\sum_{i=1}^{6}(d_i - 3.5)^2 = 2.92$$

which is obviously different from the result of s^2 as computed by `var()`.

```
var(1:6)
```

```
## [1] 3.5
```

3.2 Probability Distributions of Continuous Random Variables

Since a continuous random variable takes on a continuum of possible values, we cannot use the concept of a probability distribution as used for discrete random variables. Instead, the probability distribution of a continuous random variable is summarized by its *probability density function* (PDF).

The cumulative probability distribution function (CDF) for a continuous random variable is defined just as in the discrete case. Hence, the cumulative probability distribution of a continuous random variables states the probability that the random variable is less than or equal to a particular value.

For completeness, we present revisions of Key Concepts 2.1 and 2.2 for the continuous case.

Key Concept 2.3

Probabilities, Expected Value and Variance of a Continuous Random Variable

Let $f_Y(y)$ denote the probability density function of Y. Because probabilities cannot be negative, we have $f_Y \geq 0$ for all y. The Probability that Y falls between a and b, $a < b$ is

$$P(a \leq Y \leq b) = \int_a^b f_Y(y)\text{d}y.$$

We further have that $P(-\infty \leq Y \leq \infty) = 1$ and therefore $\int_{-\infty}^{\infty} f_Y(y)\text{d}y = 1$.

As for the discrete case, the expected value of Y is the probability weighted average of its values. Due to continuity, we use intergrals instead of sums.

The expected value of Y is

$$E(Y) = \mu_Y = \int y f_Y(y)\text{d}y.$$

The variance is the expected value of $(Y - \mu_Y)^2$. We thus have

$$\text{Var}(Y) = \sigma_Y^2 = \int (y - \mu_Y)^2 f_Y(y)\text{d}y.$$

Let us discuss an example:

Consider the continuous random variable X with propability density function

$$f_X(x) = \frac{3}{x^4}, x > 1.$$

- We can show analytically that the integral of $f_X(x)$ over the real line equals 1.

$$\begin{aligned}
\int f_X(x)\mathrm{d}x &= \int_1^\infty \frac{3}{x^4}\mathrm{d}x && (3.1)\\
&= \lim_{t\to\infty} \int_1^t \frac{3}{x^4}\mathrm{d}x && (3.2)\\
&= \lim_{t\to\infty} -x^{-3}|_{x=1}^t && (3.3)\\
&= -\left(\lim_{t\to\infty} \frac{1}{t^3} - 1\right) && (3.4)\\
&= 1 && (3.5)
\end{aligned}$$

- The expectation of X can be computed as follows:

$$\begin{aligned}
E(X) = \int x \cdot f_X(x)\mathrm{d}x &= \int_1^\infty x \cdot \frac{3}{x^4}\mathrm{d}x && (3.6)\\
&= -\frac{3}{2}x^{-2}|_{x=1}^\infty && (3.7)\\
&= -\frac{3}{2}\left(\lim_{t\to\infty} \frac{1}{t^2} - 1\right) && (3.8)\\
&= \frac{3}{2} && (3.9)
\end{aligned}$$

- Note that the variance of X can be expressed as $\mathrm{Var}(X) = E(X^2) - E(X)^2$. Since $E(X)$ has been computed in the previous step, we seek $E(X^2)$:

$$\begin{aligned}
E(X^2) = \int x^2 \cdot f_X(x)\mathrm{d}x &= \int_1^\infty x^2 \cdot \frac{3}{x^4}\mathrm{d}x && (3.10)\\
&= -3x^{-1}|_{x=1}^\infty && (3.11)\\
&= -3\left(\lim_{t\to\infty} \frac{1}{t} - 1\right) && (3.12)\\
&= 3 && (3.13)
\end{aligned}$$

So we have shown that the area under the curve equals one, that the expectation is $E(X) = \frac{3}{2}$ and we found the variance to be $\mathrm{Var}(X) = \frac{3}{4}$. However, this was quite tedious and, as we shall see soon, an analytic approach is not applicable for some probability density functions e.g. if integrals have no closed form solutions.

Luckily, R enables us to find the results derived above in an instant. The tool we use for this is the function `integrate()`. First, we have to define the functions we want to calculate integrals for as R functions, i.e. the PDF $f_X(x)$ as well as the expressions $x \cdot f_X(x)$ and $x^2 \cdot f_X(x)$.

```
# define functions
f <- function(x) 3/x^4
g <- function(x) x*f(x)
h <- function(x) x^2*f(x)
```

Next, we use `integrate()` and set lower and upper limits of integration to 1 and ∞ using arguments `lower` and `upper`. By default, `integrate()` prints the result along with an estimate of the calculation error to the console. However, the outcome is not a numeric value one can do further calculation with readily. In order to get only a numeric value of the integral, we need to use the `$` operator in conjunction with `value`.

```
# calculate area under curve
AUC <- integrate(f,
                 lower = 1,
                 upper = Inf
                 )
AUC
```

```
## 1 with absolute error < 1.1e-14
```

```
# calculate E(X)
EX <- integrate(g,
                lower = 1,
                upper = Inf)
EX
```

```
## 1.5 with absolute error < 1.7e-14
```

```
# calculate Var(X)
VarX <- integrate(h,
                  lower = 1,
                  upper = Inf
                  )$value - EX$value^2
VarX
```

```
## [1] 0.75
```

Although there is a wide variety of distributions, the ones most often encountered in econometrics are the normal, chi-squared, Student t and F distributions. Therefore we will discuss some core R functions that allow to do calculations involving densities, probabilities and quantiles of these distributions.

Every probability distribution that R handles has four basic functions whose names consist of a prefix followed by a root name. As an example, take the normal distribution. The root name of all four functions associated with the normal distribution is norm. The four prefixes are

- d for "density" - probability function / probability density function
- p for "probability" - cumulative distribution function
- q for "quantile" - quantile function (inverse cumulative distribution function)
- r for "random" - random number generator

Thus, for the normal distribution we have the R functions `dnorm()`, `pnorm()`, `qnorm()` and `rnorm()`.

The Normal Distribution

The probably most important probability distribution considered here is the normal distribution. This is not least due to the special role of the standard normal distribution and the Central Limit Theorem which is treated shortly during the course of this section. Distributions of the normal family have a familiar symmetric, bell-shaped probability density. A normal distribution is characterized by its mean μ and its standard deviation σ what is concisely expressed by $N(\mu, \sigma^2)$. The normal distribution has the PDF

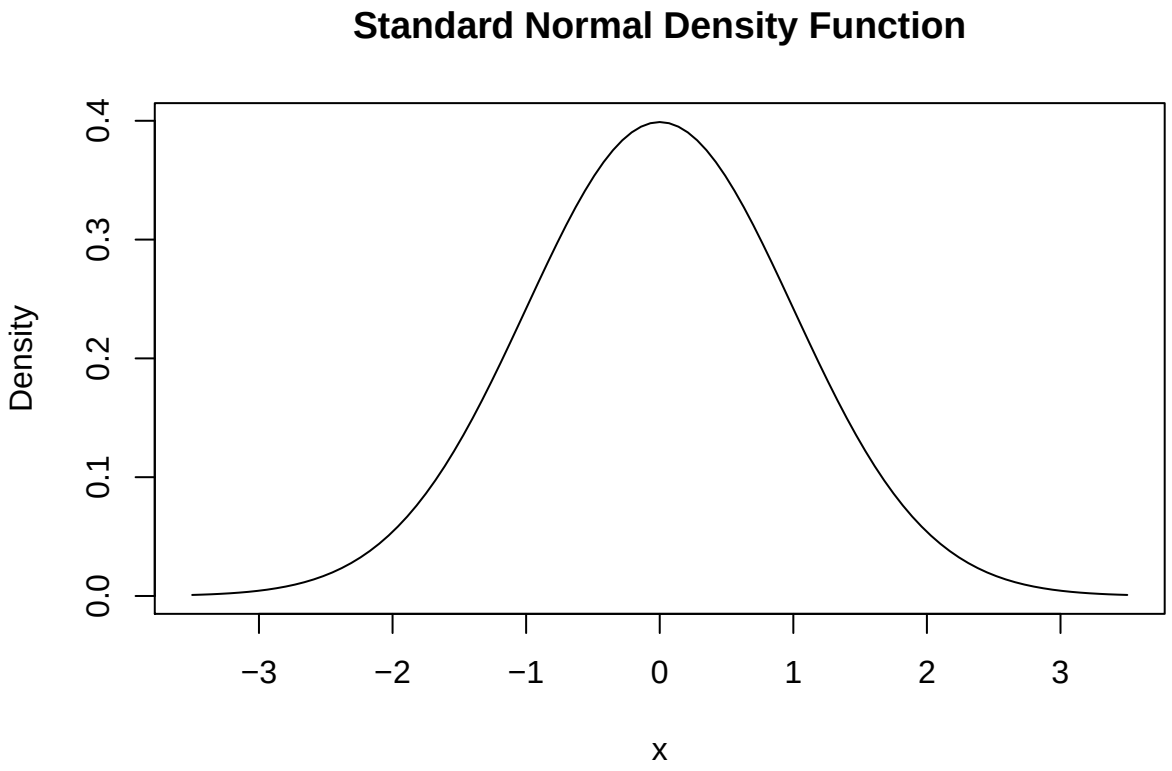
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-)^2/(2^2)}.$$

For the standard normal distribution we have $\mu = 0$ and $\sigma = 1$. Standard normal variates are often denoted Z. Usually, the standard normal PDF is denoted by ϕ and the standard normal CDF is denoted by Φ. Hence,

$$\phi(c) = \Phi'(c) \quad , \quad \Phi(c) = P(Z \leq c) \quad , \quad Z \sim N(0,1).$$

In R, we can conveniently obtain density values of normal distributions using the function `dnorm()`. Let us draw a plot of the standard normal density function using `curve()` and `dnorm()`.

```
# draw a plot of the N(0,1) pdf
curve(dnorm(x),
      xlim=c(-3.5, 3.5),
      ylab = "Density",
      main = "Standard Normal Density Function"
      )
```



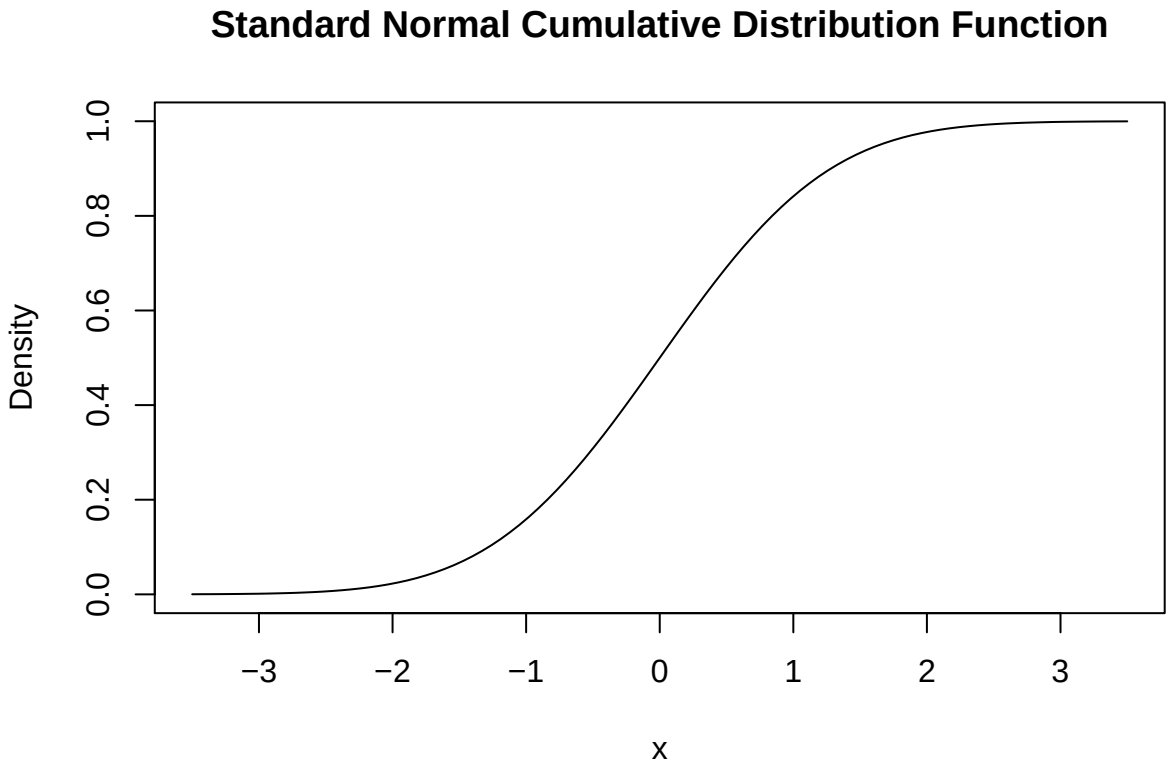
We can obtain the density at different positions by passing a vector of quantiles to `dnorm()`.

```
# compute denstiy at x=-1.96, x=0 and x=1.96
dnorm(x = c(-1.96, 0, 1.96))
```

```
## [1] 0.05844094 0.39894228 0.05844094
```

Similary as for the PDF, we can plot the standard normal CDF using `curve()` and `pnorm()`.

```
# plot the standard normal CDF
curve(pnorm(x),
      xlim=c(-3.5, 3.5),
      ylab = "Density",
      main = "Standard Normal Cumulative Distribution Function"
      )
```



We can also use R to calculate the probability of events associated with a standard normal random variate.

Let us say we are interested in $P(Z \leq 1.337)$. For some general continuous random variable Z on $[-\infty, \infty]$ with density function $g(x)$ we would have to determine $G(x)$, the antiderivative of $g(x)$ since

$$P(Z \leq 1,337) = G(1,337) = \int_{-\infty}^{1,337} g(x)\mathrm{d}x.$$

However, if $Z \sim N(0,1)$, we have $g(x) = \phi(x)$ so there is no analytic solution to the integral above and it is cumbersome to come up with an approximation. However, we may circumvent this using R in different ways. The first approach makes use of the function `integrate()` which allows to solve one-dimensional integration problems using a numerical method. For this, we first define the function we want to compute the integral of as a R function `f`. In our example, `f` needs to be the standard normal density function and hence takes a single argument `x`. Following the definition of $\phi(x)$ we define `f` as

```
# define the standard normal PDF as a R function
f <- function(x) {
  1/(sqrt(2 * pi)) * exp(-0.5 * x^2)
}
```

Let us check if this function enables us to compute standard normal density values by passing it a vector of quantiles.

```
# define vector of quantiles
quants <- c(-1.96,0,1.96)

# compute density values
f(quants)
```

```
## [1] 0.05844094 0.39894228 0.05844094
```

```
# compare to results produced by dnorm()
f(quants) == dnorm(quants)
```

```
## [1] TRUE TRUE TRUE
```

Notice that the results produces by `f()` are indeed equivalent to those given by `dnorm()`.

Next, we call `integrate()` on `f()` and further specify the arguments `lower` and `upper`, the lower and upper limits of integration.

```
# integrate f()
integrate(f,
          lower = -Inf,
          upper = 1.337
          )
```

```
## 0.9093887 with absolute error < 1.7e-07
```

We find that the probability of observing $Z \leq 1,337$ is about 0.9094%.

A second and much more convenient way is to use the function `pnorm()` which also allows calculus involving the standard normal cumulative distribution function.

```
# compute a probability using pnorm()
pnorm(1.337)
```

```
## [1] 0.9093887
```

The result matches the outcome of the approach using `integrate()`.

Let us discuss some further examples:

A commonly known result is that 95% probability mass of a standard normal lies in the intervall $[-1.96, 1.96]$, that is in a distance of about 2 standard deviations to the mean. We can easily confirm this by calculating

$$P(-1.96 \leq Z \leq 1.96) = 1 - 2 \times P(Z \leq -1.96)$$

due to symmetry of the standard normal PDF. Thanks to R, we can abondon the table of the standard normal CDF again and instead solve this by means of the function `pnorm()`.

```
# compute the probability
1 - 2 * (pnorm(-1.96))
```

```
## [1] 0.9500042
```

Now consider a random variable Y with $Y \sim N(5, 25)$. As You should already know from Your statistics courses it is not possible to make any statement of probability without prior standardizing as shown in Key Concept 2.4.

Key Concept 2.4

Computing Probabilities Involving Normal Random Variables

Suppose Y is normally distributed with mean μ and variance σ^2:

$$Y \sim N(\mu, \sigma^2)$$

Then Y is standardized by substracting its mean and dividing by its standard deviation:

$$Z = \frac{Y - \mu}{\sigma}$$

Let c_1 and c_2 denote two numbers whereby $c_1 < c_2$ and further $d_1 = (c_1 - \mu)/\sigma$ and $d_2 = (c_2 - \mu)/\sigma$. Then

$$P(Y \leq c_2) = P(Z \leq d_2) = \Phi(d_2) \tag{3.14}$$
$$P(Y \geq c_1) = P(Z \geq d_1) = 1 - \Phi(d_1) \tag{3.15}$$
$$P(c_1 \leq Y \leq c_2) = P(d_1 \leq Z \leq d_2) = \Phi(d_2) - \Phi(d_1) \tag{3.16}$$

R functions that handle the normal distribution can perform this standardization. If we are interested in $P(3 \leq Y \leq 4)$ we can use `pnorm()` and adjust for a mean and/or a standard deviation that deviate from $\mu = 0$ and $\sigma = 1$ by specifying the arguments `mean` and `sd` accordingly. **Attention**: `pnorm()` requires the argument `sd` which is the standard deviation, not the variance!

```
pnorm(4, mean = 5, sd = 5) - pnorm(3, mean = 5, sd = 5)
```

```
## [1] 0.07616203
```

An extension of the normal distribution in a univariate setting is the multivariate normal distribution. The PDF of two random normal variables X and Y is given by

$$g_{X,Y}(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho_{XY}^2}} \tag{3.17}$$
$$\cdot \exp\left\{\frac{1}{-2(1-\rho_{XY}^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho_{XY}\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right) + \left(\frac{y-\mu_Y}{\sigma_Y}\right)^2\right]\right\}. \tag{3.18}$$

Equation (3.18) contains the bivariate normal PDF. Admittedly, it is hard to gain insights from this complicated expression. Instead, let us consider the special case where X and Y are uncorrelated standard normal random variables with density functions $f_X(x)$ and $f_Y(y)$ and we assume that they have a joint normal distribution. We then have the parameters $\sigma_X = \sigma_Y = 1$, $\mu_X = \mu_Y = 0$ (due to marginal standard normality) and $\rho_{XY} = 0$ (due to uncorrelatedness). The joint probability density function of X and Y then becomes

$$g_{X,Y}(x,y) = f_X(x)f_Y(y) = \frac{1}{2\pi} \cdot \exp\left\{-\frac{1}{2}\left[x^2 + y^2\right]\right\}, \tag{2.2}$$

the PDF of the bivariate standard normal distribution. The next plot provides an interactive three dimensional plot of (2.2). By moving the mouse curser over the plot You can see that the density is rotationally invariant.

The Chi-Squared Distribution

Another distribution relevant in econometric day-to-day work is the chi-squared distribution. It is often needed when testing special types of hypotheses frequently ecountered when dealing with regression models.

The sum of M squared independent standard normal distributed random variables follows a chi-squared distribution with M degrees of freedom.

$$Z_1^2 + \cdots + Z_M^2 = \sum_{m=1}^{M} Z_m^2 \sim \chi_M^2 \quad \text{with} \quad Z_m \overset{i.i.d.}{\sim} N(0,1)$$

A χ^2 distributed random variable with M degrees of freedom has expectation M, mode at $M-2$ for $n \geq 2$ and variance $2 \cdot M$.

For example, if we have

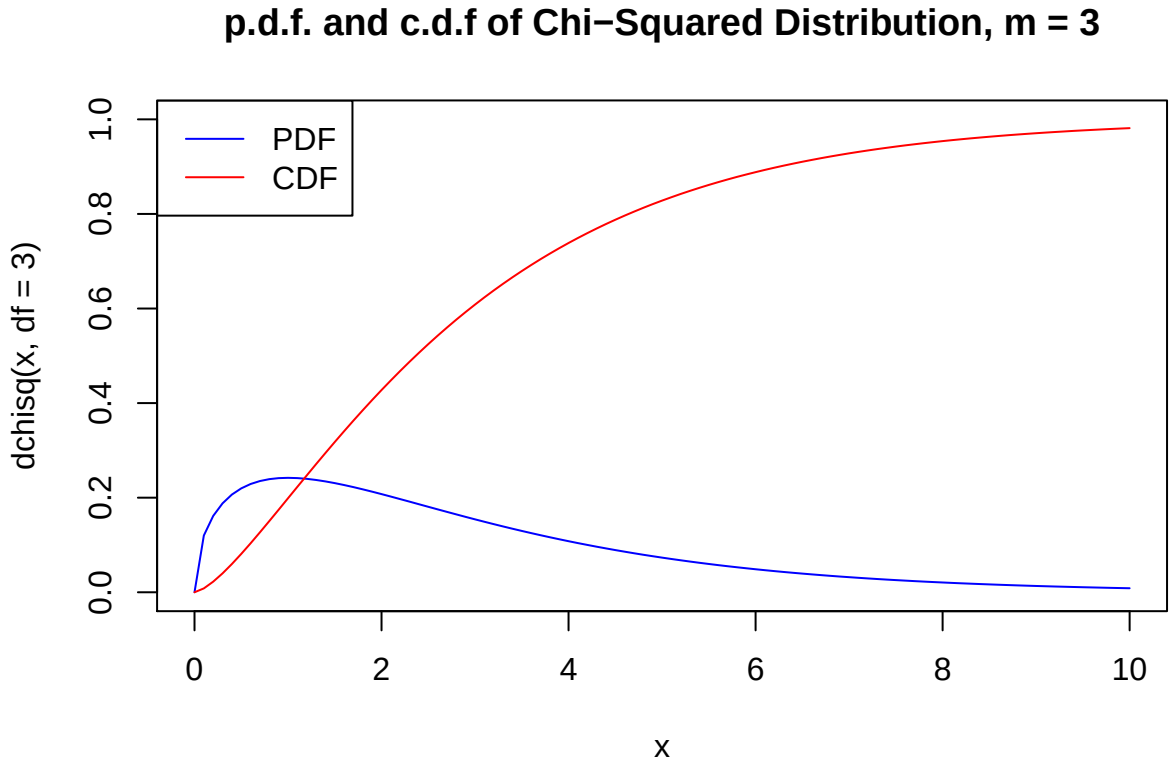
$$Z_1, Z_2, Z_3 \overset{i.i.d.}{\sim} N(0,1)$$

it holds that

$$Z_1^2 + Z_2^2 + Z_3^3 \sim \chi_3^2. \tag{2.3}$$

By means of the code below, we can display the PDF and the CDF of a χ_3^2 random variable in a single plot. This is achieved by setting the argument `add = TRUE` in the second call of `curve()`. Further we adjust limits of both axes using `xlim` and `ylim` and choose different colors to make both functions better distinguishable. The plot is completed by adding a legend with help of the function `legend()`.

```
# plot the PDF
curve(dchisq(x, df=3),
      xlim=c(0,10),
      ylim = c(0,1),
      col="blue",
      main="p.d.f. and c.d.f of Chi-Squared Distribution, m = 3"
      )

# add the CDF to the plot
curve(pchisq(x, df=3),
      xlim=c(0,10),
      add = TRUE,
      col="red"
      )

# add a legend to the plot
legend("topleft",
       c("PDF","CDF"),
       col = c("blue","red"),
       lty = c(1,1)
       )
```



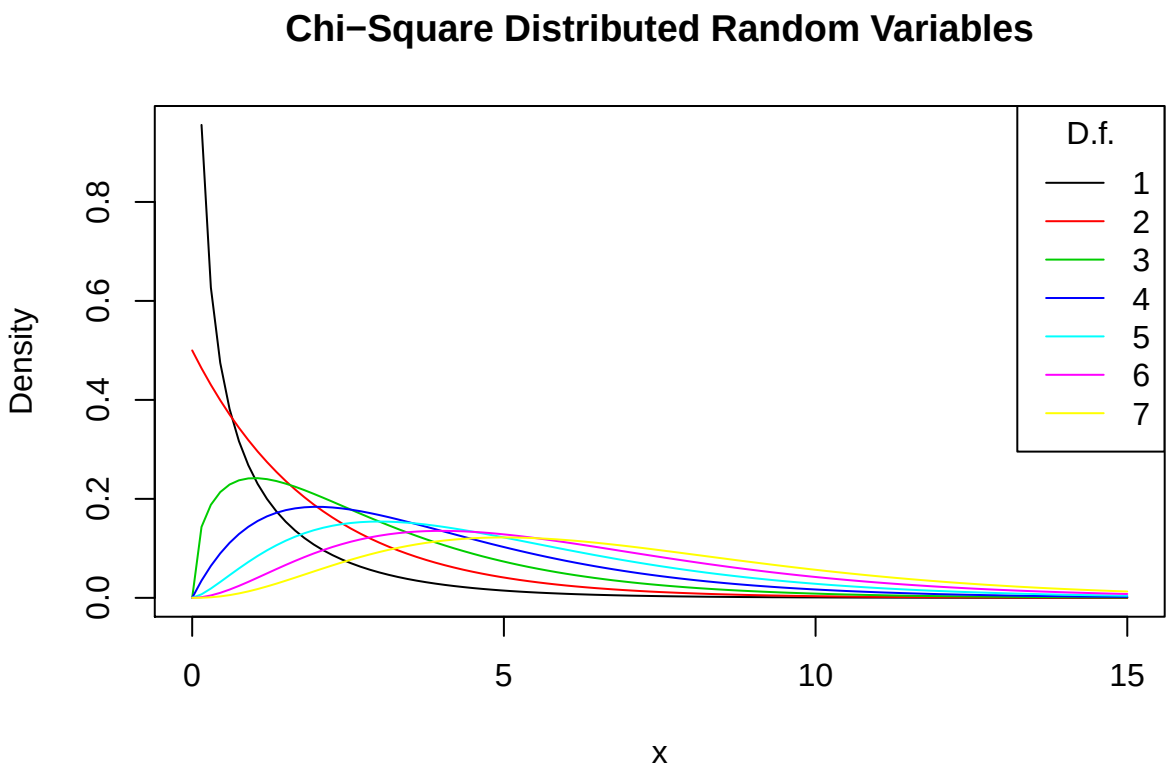
Notice that, since the outcomes of a χ^2_M distributed random variable are always positive, the domain of the related PDF and CDF is $\mathbb{R}_{\geq 0}$.

As expectation and variance depend (solely) on the degrees of freedom, the distribution's shape changes drastically if we vary the number of squared standard normals that are summed up. This relation is often depicted by overlaying densities for different M, see e.g. the Wikipedia Article.

Of course, one can easily reproduce such a plot using R. Again we start by plotting the density of the χ^2_1 distribution on the intervall $[0, 15]$ with `curve()`. In the next step, we loop over degrees of freedom $m = 2, ..., 7$ and add a density curve for each m to the plot. We also adjust the line color for each iteration of the loop by setting `col = m`. At last, we add a legend that displays degrees of freedom and the associated colors.

```
# plot the density for m=1
curve(dchisq(x, df=1),
      xlim=c(0,15),
      xlab = "x",
      ylab = "Density",
      main="Chi-Square Distributed Random Variables"
      )

# add densities for m=2,...,7 to the plot using a for loop
for (m in 2:7) {
  curve(dchisq(x, df = m),
        xlim = c(0,15),
        add = T,
        col = m
        )
}

# add a legend
```

```
legend("topright",
       as.character(1:7),
       col = 1:7 ,
       lty = 1,
       title = "D.f."
       )
```



It is evident that increasing the degrees of freedom shifts the distribution to the right (the modus becomes larger) and increases its dispersion (the distribution's variance grows).

The Student t Distribution

Let Z be a standard normal variate, W a random variable that follows a χ^2_M distribution with M degrees of freedom and further assume that Z and W are independently distributed. Then it holds that

$$\frac{Z}{\sqrt{W/M}} =: X \sim t_M$$

and we say that X follows a student t distribution (or simple t distribution) with M degrees of freedom.

As for the χ^2_M distribution, a t distribution depends on the degrees of freedom M. t distributions are symmetric, bell-shaped and look very similar to a normal distribution, especially when M is large. This is not a coincidence: for a sufficient large M, a t_M distribution can be approximated by the standard normal distribution. This approximation works reasonably well for $M \geq 30$. As we will show later by means of a small simulation study, the t_∞ distribution *is* the standard normal distribution.

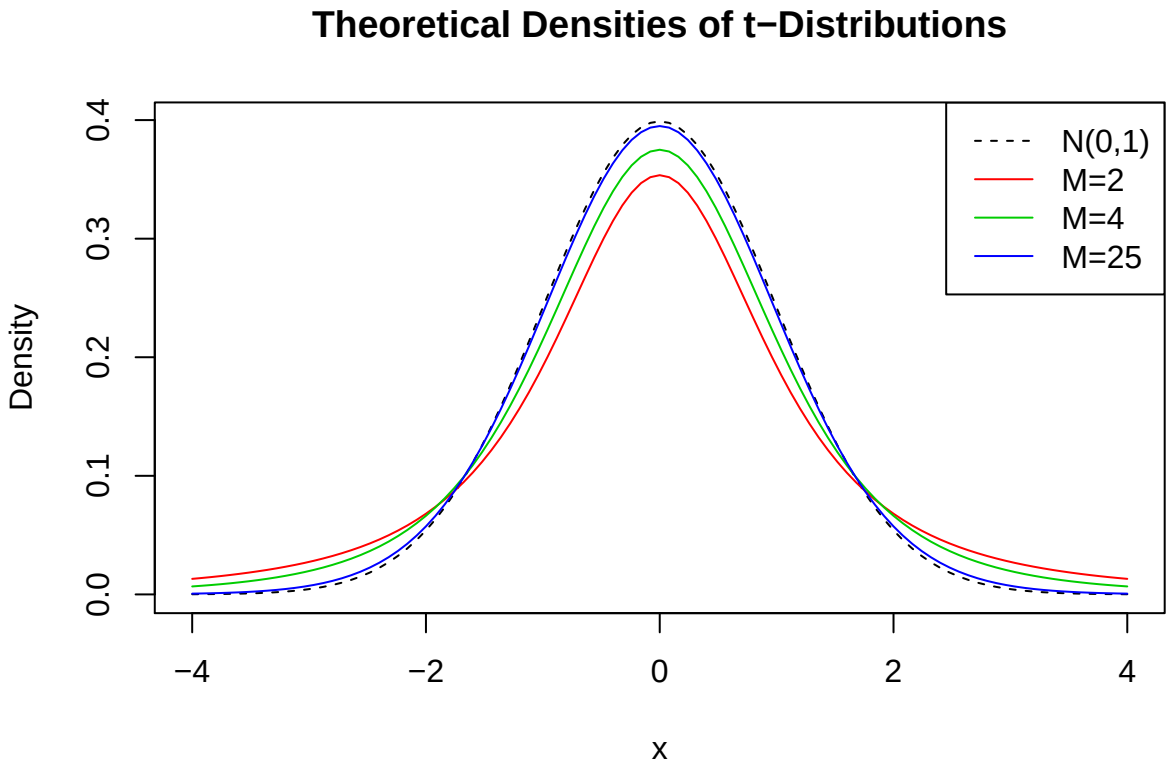
A t_M distributed random variable has an expectation value if $M > 1$ and a variance if $n > 2$.

$$E(X) = 0 \ , \ M > 1 \tag{3.19}$$

$$\text{Var}(X) = \frac{M}{M-2} \ , \ M > 2 \tag{3.20}$$

Let us graph some t distributions with different M and compare them with the standard normal distribution.

```
# plot the standard normal density
curve(dnorm(x),
      xlim=c(-4,4),
      xlab = "x",
      lty=2,
      ylab = "Density",
      main="Theoretical Densities of t-Distributions"
      )

# plot the t density for m=2
curve(dt(x, df=2),
      xlim=c(-4,4),
      col=2,
      add = T
      )

# plot the t density for m=4
curve(dt(x, df=4),
      xlim=c(-4,4),
      col=3,
      add=T
      )

# plot the t density for m=25
curve(dt(x, df=25),
      xlim=c(-4,4),
      col=4,
      add=T
      )

# add a legend
legend("topright",
       c("N(0,1)","M=2","M=4","M=25"),
       col = 1:4,
       lty = c(2,1,1,1)
       )
```



The plot indicates what has been claimed in the previous paragraph: as the degrees of freedom increase, the shape of the t distribution comes closer to that of a standard normal bell. Already for $M = 25$ we find little difference to the dashed line which belongs to the standard normal density curve. If M is small, we find the distribution to have slightly havier tails than a standard normal, i.e. it has a "fatter" bell shape.

The F Distribution

Another ratio of random variables important to econometricians is the ratio of two indpendently χ^2 distributed random variables that are divided by their degrees of freedom. Such a quantity follows a F distribution with numerator degrees of freedom M and denominator degrees of freedom n, denoted $F_{M,n}$. The distribution was first derived by George Snedecor but was named in honor of Sir Ronald Fisher.

$$\frac{W/M}{V/n} \sim F_{M,n} \quad \text{with} \quad W \sim \chi^2_M \quad , \quad V \sim \chi^2_n$$

By definition, the domain of both PDF and CDF of a $F_{M,n}$ distributed random variable is $\mathbb{R}_{\geq 0}$.

Say we have a F distributed random variable Y with numerator degrees of freedom 3 and denominator degrees of freedom 14 and are interested in $P(Y \geq 2)$. This can be computed with help of the function `pf()`. By setting the argument `lower.tail` to `TRUE` we ensure that R computes $1 - P(Y \leq 2)$, i.e. the probability mass in the tail right of 2.

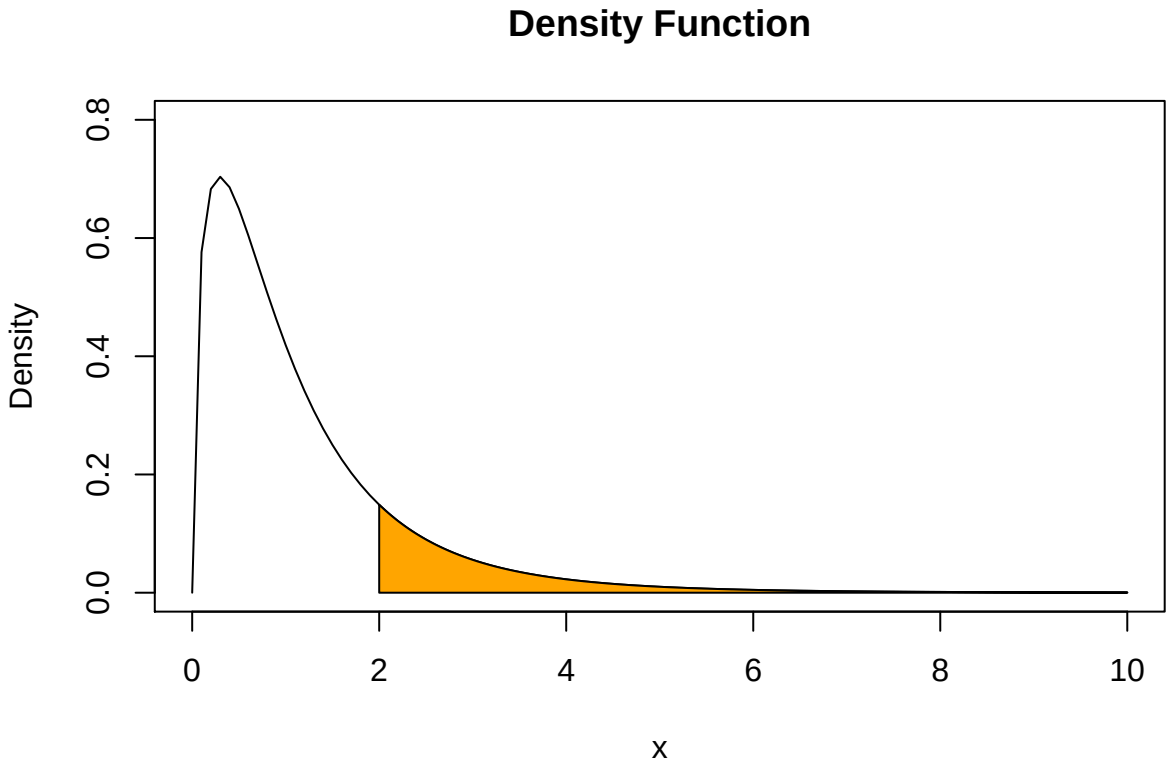
```
pf(2, 3, 13, lower.tail = F)
```

```
## [1] 0.1638271
```

We can visualize this probability by drawing a line plot of the related density function and adding a color shading with `polygon()`.

```
# define coordinate vectors for vertices of the polygon
x <- c(2, seq(2, 10, 0.01), 10)
```

```
y <- c(0, df(seq(2, 10, 0.01), 3, 14), 0)

# draw density of F_{3, 14}
curve(df(x ,3 ,14),
      ylim = c(0, 0.8),
      xlim = c(0, 10),
      ylab = "Density",
      main = "Density Function"
      )

# draw the polygon
polygon(x, y, col="orange")
```



The F distribution is related to many other distributions. An important special case encountered in econometrics arises if the denominator degrees of freedom are large such that the $F_{M,n}$ distribution can be approximated by the $F_{M,\infty}$ distribution which turns out to be simply the distribution of a χ^2_M random variable divided by its degrees of freedom M.

$$W/M \sim F_{M,\infty} \quad , \quad W \sim \chi^2_M$$

3.3 Random Sampling and the Distribution of Sample Averages

To clarify the basic idea of random sampling, let us jump back to the die rolling example:

Suppose we are rolling the die n times. This means we are interested in the outcomes of n random processes Y_i, $i = 1, ..., n$ which are characterized by the same distribution. Since these outcomes are selected randomly, they are *random variables* themselves and their realisations will differ each time we draw a sample, i.e. each time we roll the die n times. Furthermore, each observation is randomly drawn from the same population,

that is the numbers from 1 to 6, and their individual distribution is the same. Hence we say that $Y_1, \dots, Y_n$ are identically distributed. Moreover, we know that the value of any of the Y_i does not provide any information on the remainder of the sample. In our example, rolling a six as the first observation in our sample does not alter the distributions of $Y_2, \dots, Y_n$: all numbers are equally likely to occur. This means that all Y_i are also independently distributed. Thus, we say that $Y_1, \dots, Y_n$ are independently and identically distributed (*i.i.d*). The die example uses the most simple sampling scheme. That is why it is called *simple random sampling*. This concept is condensed in Key Concept 2.5.

Key Concept 2.5

Simple Random Sampling and i.i.d. Random Variables

In simple random sampling, n objects are drawn at random from a population. Each object is equally likely to end up in the sample. We denote the value of the random variable Y for the i^{th} randomly drawn object as Y_i. Since all objects are equally likely to be drawn and the distribution of Y_i is the same for all i, the $Y_i, \dots, Y_n$ are independently and identically distributed (i.i.d.). This means the distribution of Y_i is the same for all $i = 1, \dots, n$ and Y_1 is distributed independently of $Y_2, \dots, Y_n$ Y_2 is distributed independently of $Y_1, Y_3, \dots, Y_n$ and so forth.

What happens if we consider functions of the sample data? Consider the example of rolling a die two times in a row once again. A sample now consists of two independent random draws from the set $\{1, 2, 3, 4, 5, 6\}$. In view of the aforementioned, it is apparent that any function of these two random variables is also random, e.g. their sum. Convince Yourself by executing the code below several times.

```
sum(sample(1:6, 2, replace = T))
```

```
## [1] 6
```

Clearly this sum, let us call it S, is a random variable as it depends on randomly drawn summands. For this example, we can completely enumerate all outcomes and hence write down the theoretical probability distribution of our function of the sample data, S:

We face $6^2 = 36$ possible pairs. Those pairs are

$$(1,1)(1,2)(1,3)(1,4)(1,5)(1,6) \tag{3.21}$$
$$(2,1)(2,2)(2,3)(2,4)(2,5)(2,6) \tag{3.22}$$
$$(3,1)(3,2)(3,3)(3,4)(3,5)(3,6) \tag{3.23}$$
$$(4,1)(4,2)(4,3)(4,4)(4,5)(4,6) \tag{3.24}$$
$$(5,1)(5,2)(5,3)(5,4)(5,5)(5,6) \tag{3.25}$$
$$(6,1)(6,2)(6,3)(6,4)(6,5)(6,6) \tag{3.26}$$

Thus, possible outcomes for S are

$$\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}.$$

Enumeration of outcomes yields

$$P(S) = \begin{cases} 1/36, & S = 2 \\ 2/36, & S = 3 \\ 3/36, & S = 4 \\ 4/36, & S = 5 \\ 5/36, & S = 6 \\ 6/36, & S = 7 \\ 5/36, & S = 8 \\ 4/36, & S = 9 \\ 3/36, & S = 10 \\ 2/36, & S = 11 \\ 1/36, & S = 12 \end{cases} \tag{3.27}$$

We can also compute $E(S)$ and $\text{Var}(S)$ as stated in Key Concept 2.1 and Key Concept 2.2.

```
# Vector of outcomes
S <- 2:12

# Vector of probabilities
PS <- c(1:6,5:1)/36

# Expectation of S
ES <- S %*% PS; ES
```

```
##      [,1]
## [1,]    7
```

```
# Variance of S
VarS <- (S - c(ES))^2 %*% PS; VarS
```

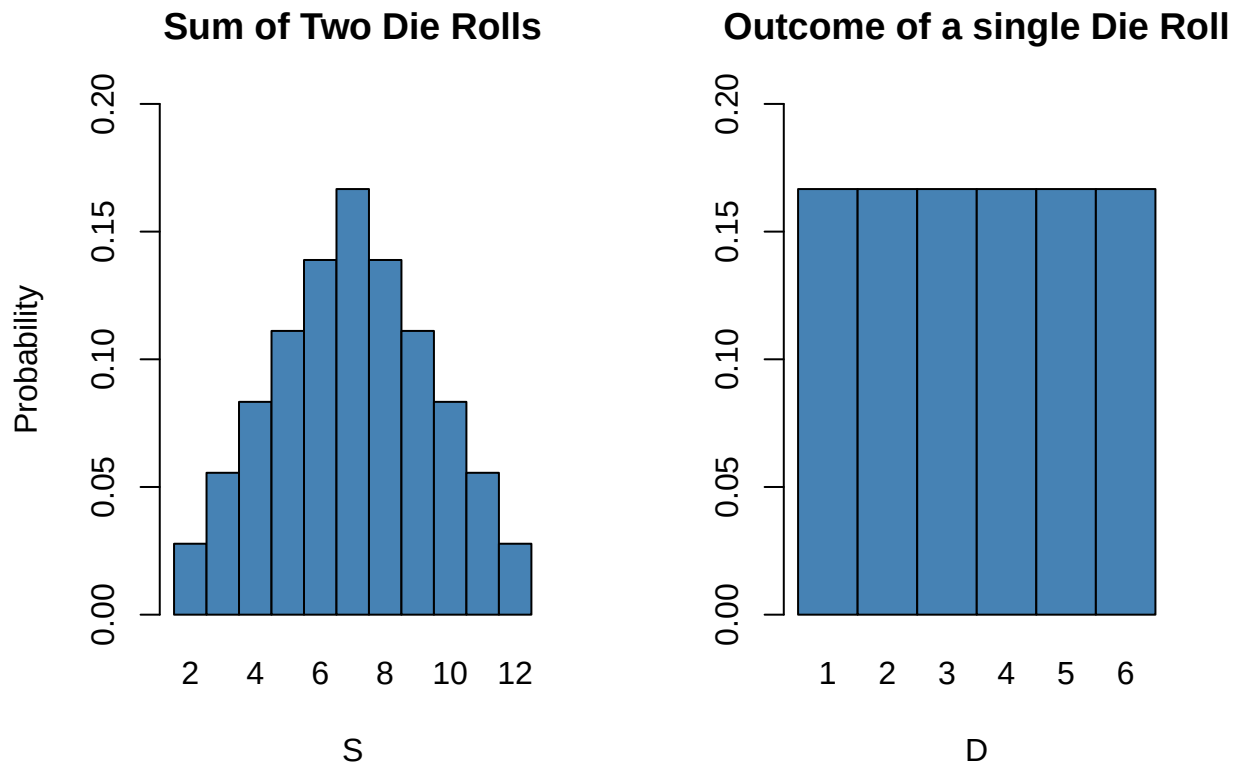
```
##          [,1]
## [1,] 5.833333
```

(The `%*%` operator is used to compute the scalar product of two vectors.)

So the distribution of S is known. It is also evident that its distribution differs considerably from the marginal distribution, i.e. the distribution of a single die roll's outcome, D . Let us visualize this using barplots.

```
# divide the plotting area in one row with two columns
par(mfrow = c(1, 2))

# plot the distribution of S
names(PS) <- 2:12

barplot(PS, ylim=c(0,0.2),
        xlab = "S",
        ylab ="Probability",
        col="steelblue",
        space=0,
        main="Sum of Two Die Rolls"
        )

# plot the distribution of D
probability <- rep(1/6,6)
```

```
names(probability) <- 1:6

barplot(probability,
        ylim=c(0,0.2),
        xlab = "D",
        col="steelblue",
        space = 0,
        main = "Outcome of a single Die Roll"
        )
```



Many econometric procedures deal with averages of sampled data. It is almost always assumed that observations are drawn randomly from a larger, unkown population. As demonstrated for the sample function S, computing an average of a random sample also has the effect to make the average a random variable itself. This random variable in turn has a probability distribution which is called the sampling distribution. Knowledge about the sampling distribution of an average is therefore crucial for understanding the performance of econometric procedures.

The *sample average* of a sample of n observations $Y_1, \dots, Y_n$ is

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i = \frac{1}{n}(Y_1 + Y_2 + \cdots + Y_n).$$

$\overline{Y}$ is also called the sample mean.

Mean and Variance of the Sample Mean

Denote μ_Y and σ_Y^2 the mean and the variance of the Y_i and suppose that all observations $Y_1, \dots, Y_n$ are i.i.d. such that in particular mean and variance are the same for all $i = 1, \dots, n$. Then we have that

$$E(\overline{Y}) = E\left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right) = \frac{1}{n}E\left(\sum_{i=1}^{n} Y_i\right) = \frac{1}{n}\sum_{i=1}^{n} E\left(Y_i\right) = \frac{1}{n}\cdot n \cdot \mu_Y = \mu_Y$$

and

$$\begin{aligned}
\text{Var}(\overline{Y}) =& \text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right) && (3.28)\\
=& \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(Y_i) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\text{cov}(Y_i, Y_j) && (3.29)\\
=& \frac{\sigma_Y^2}{n} && (3.30)\\
=& \sigma_{\overline{Y}}^2. && (3.31)
\end{aligned}$$

Note that the second summand vanishes since $\text{cov}(Y_i, Y_j) = 0$ for $i \neq j$ due to independence of the observations.

Consequently, the standard deviation of the sample mean is given by

$$\sigma_{\overline{Y}} = \frac{\sigma_Y}{\sqrt{n}}.$$

It is worthwhile to mention that these results hold irrespective of the underlying distribution of the Y_i.

The Sampling Distribution of $\overline{Y}$ when Y Is Normally Distributed

If the $Y_1, \ldots, Y_n$ are i.i.d. draws from a normal distribution with mean μ_Y and variance σ_Y^2, the following holds for their sample average $\overline{Y}$:

$$\overline{Y} \sim N(\mu_y, \sigma_Y^2/n) \tag{2.4}$$

For example, if a sample Y_i with $i = 1, \ldots, 10$ is drawn from a standard normal distribution with mean $\mu_Y = 0$ and variance $\sigma_Y^2 = 1$ we have

$$\overline{Y} \sim N(0, 0.1).$$

We can use R's random number generation facilities to verifiy this result. The basic idea is to simulate outcomes of the true distribution of $\overline{Y}$ by repeatedly drawing random samples of 10 observation from the $N(0,1)$ distribution and computing their respective averages. If we do this for a large number of repititions, the simulated dataset of averages should quite accurately reflect the theoretical distribution of $\overline{Y}$ if the theoretical result holds.

The approach sketched above is an example of what is commonly known as *Monte Carlo Simulation* or *Monte Carlo Experiment*. To perform this simulation in R, we proceed as follows:

1. Choose a sample size `n` and the number of samples to be drawn `reps`.
2. Use the function `replicate()` in conjunction with `rnorm()` to draw `n` observations from the standard normal distribution `rep` times. **Note**: the outcome of `replicate()` is a matrix with dimensions `n` $\times$ `rep`. It contains the drawn samples as *columns*.
3. Compute sample means using `colMeans()`. This function computes the mean of each column i.e. of each sample and returns a vector.

```
# Set sample size and number of samples
n <- 10
reps <- 10000

# Perform random sampling
samples <- replicate(reps, rnorm(n)) # 10 x 10000 sample matrix

# Compute sample means
sample.avgs <- colMeans(samples)
```

After performing these steps we end up with a vector of sample averages. You can check the vector property of `sample.avgs`:

```
# Check that sample.avgs is a vector
is.vector(sample.avgs)
```

```
## [1] TRUE
```

```
# print the first few entries to the console
head(sample.avgs)
```

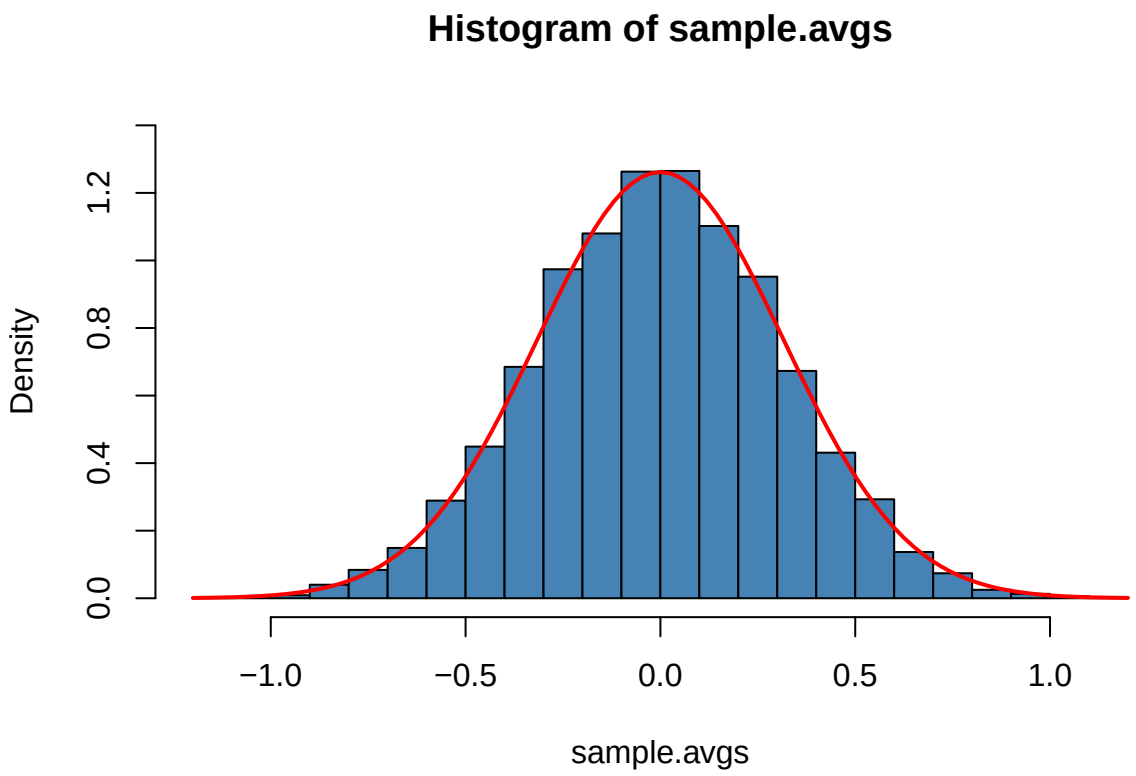
```
## [1] -0.12406767 -0.10649421 -0.01033423 -0.39905236 -0.41897968 -0.90883537
```

A straightforward approach to examine the distribution of univariate numerical data is to plot it as a histogram and compare it to some known or assumed distribution. This comparison can be done with help of a suitable statistical test or by simply eyeballing some graphical representations of these distributions. For our simulated sample averages, we will do the latter by means of the functions `hist()` and `curve()`.

By default, `hist()` will give us a frequency histogram i.e. a bar chart where observations are grouped into ranges, also called bins. The ordinate reports the number of observations falling into each of the bins. Instead, we want it to report density estimates for comparison purposes. This is achieved by setting the argument `freq = FALSE`. The number of bins is adjusted by the argument `breaks`.

Using `curve()`, we overlay the histogram with a red line which represents the theoretical density of a $N(0, 0.1)$ distributed random variable. Remember to use the argument `add = TRUE` to add the curve to the current plot. Otherwise R will open a new graphic device and discard the histogram plot!

```
# Plot the density histogram
hist(sample.avgs,
     ylim=c(0,1.4),
     col="steelblue" ,
     freq = F,
     breaks = 20
     )

# overlay the theoretical distribution of sample averages on top of the histogram
curve(dnorm(x, sd = 1/sqrt(n)),
      col="red",
      lwd="2",
      add=T
      )
```



From inspection of the plot we can tell that the distribution of $\overline{Y}$ is indeed very close to that of a $N(0, 0.1)$ distributed random variable so that evidence obained from the Monte Carlo Simulation supports the theoretical claim.

Let us discuss another example where using simple random sampling in a simulation setup helps to verify a well known result. As discussed before, the Chi-squared distribution with m degrees of freedom arises as the distribution of the sum of m independent squared standard normal distributed random variables.

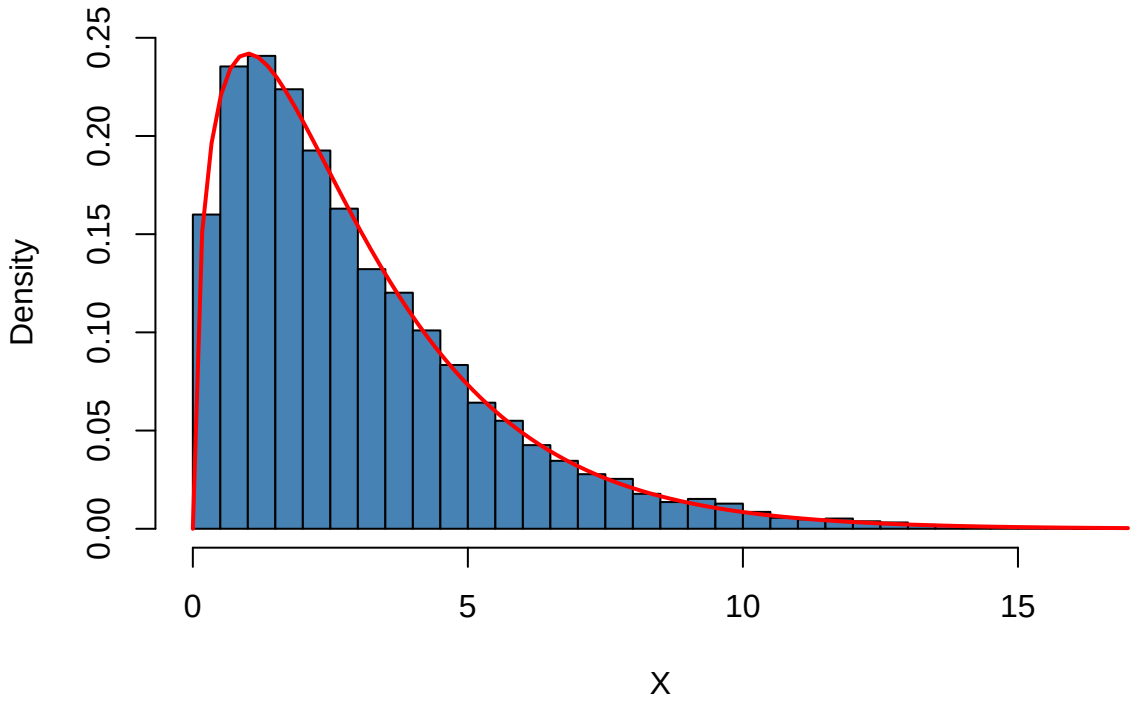
To visualize the claim stated in equation (2.3), we proceed similarly as in the example before:

1. Choose the degrees of freedom `DF` and the number of samples to be drawn `reps`.
2. Draw `reps` random samples of size `DF` from the standard normal distribution using `replicate()`.
3. For each sample, by squaring the outcomes and summing them up columnwise. Store the results

Again, we produce a density estimate for the distribution underlying our simulated data using a density histogram and overlay it with a line graph of the theoretical density function of the χ^2_3 distribution.

```
# Number of repititions
reps <- 10000

# Set degrees of freedom of a chi-Square Distribution
DF <- 3

# Sample 10000 column vectors à 3 N(0,1) R.V.S
Z <- replicate(reps, rnorm(DF))

# Column sums of squares
X <- colSums(Z^2)

# Histogram of column sums of squares
hist(X,
     freq = F,
```

```
     col="steelblue",
     breaks = 40,
     ylab="Density",
     main=""
     )

# Add theoretical density
curve(dchisq(x, df = DF),
      type = 'l',
      lwd = 2,
      col="red",
      add = T
      )
```



Large Sample Approximations to Sampling Distributions

Sampling distributions as considered in the last section play an important role in the development of econometric methods. In general, there are two different approaches in characterizing sampling distributions: an "exact" approach and an "approximate" approach.

The exact approach aims to find a general formula for the sampling distribution that holds for any sample size n. We call this the *exact distribution* or *finite sample distribution*. In the previous examples of die rolling and normal variates, we have dealt with functionals of random variables whose sample distributions are *excactly known* in the sense that we can write them down as analytical expressions and do calculations. However, this is not always possible. For $\overline{Y}$, result (2.4) tells us that normality of the Y_i implies normality of $\overline{Y}$ (we demonstrated this for the special case of $Y_i \overset{i.i.d.}{\sim} N(0,1)$ with $n = 10$ using a simulation study that involves random sampling). Unfortunately, the *exact* distribution of $\overline{Y}$ is unknown, very hard to derive or even untractable if we drop the assumption of Y having a normal distribution.

Therefore, as can be guessed from its name, the "approximate" approach aims to find an approximation to the sampling distribution wherby it is required that the sample size n is large. A distribution that is used as

a large-sample approximation to the sampling distribution is also called the *asymptotic distribution*. This is due to the fact that the asymptotic distribution *is* the sampling distribution for $n \to \infty$ i.e. the approximation becomes exact if the sample size goes to infinity. However, there are cases where the difference between the sampling distribution and the asymptotic distribution is negligible for moderate or even small samples sizes so that approximations work very good.

In this section we will discuss two well known results that are used to approximate sampling distributions and thus consitute key tools in econoemtric theory: the *law of large numbers* and the *central limit theorem*. The law of large numbers states that in large samples, $\overline{Y}$ is close to μ_Y with high probability. The central limit theorem says that the sampling distribution of the standardized sample average, that is $(\overline{Y} - \mu_Y)/\sigma_{\overline{Y}}$ is asymptotically normal distributed. It is particualarly interesting that both results do not depend on the distribution of Y. In other words, beeing unable to describe the complicated sampling distribution of $\overline{Y}$ if Y is not normal, approximations of the latter using the central limit theorem simplify the development and applicability of econometric procedures enormously. This is a key component underlying the theory of statistical inference for regression models. Both results are summarized in Key Concept 2.6 and Key Concept 2.7.

Key Concept 2.6

Convergence in Probability, Consistency and the Law of Large Numbers

The sample average $\overline{Y}$ converges in probability to μ_Y — we say that $\overline{Y}$ is *consistent* for μ_Y — if the probability that $\overline{Y}$ is in the range $(\mu_Y - \epsilon)$ to $(\mu_Y + \epsilon)$ becomes arbitrarly close to 1 as n increases for any constant $\epsilon > 0$. We write this short as

$$\overline{Y} \xrightarrow{p} \mu_Y.$$

Consider the independently and identically distributed random variables $Y_i, i = 1, \dots, n$ with expectation $E(Y_i) = \mu_Y$ and variance $\text{Var}(Y_i) = \sigma_Y^2$. Under the condition that $\sigma_Y^2 < \infty$, that is large outliers are unlikely, the law of large numbers states

$$\overline{Y} \xrightarrow{p} \mu_Y.$$

The core statement of the law of large numbers is that under quite general conditions, the probability of obtaining a sample average $\overline{Y}$ that is close to μ_Y is high if we have a large sample size.

Consider the example of repeatedly tossing a coin where Y_i is the result of the i^{th} coin toss. Y_i is a Bernoulli distributed random variable with

$$P(Y_i) = \begin{cases} p, & Y_i = 1 \\ 1 - p, & Y_i = 0 \end{cases}$$

where $p = 0.5$ as we assume a fair coin. It is straightforward to show that

$$\mu_Y = p = 0.5.$$

Say p is the probabiliy of observing head and denote R_n the proportion of heads in the first n tosses,

$$R_n = \frac{1}{n} \sum_{i=1}^{n} Y_i. \tag{2.5}$$

According to the law of large numbers, the observed proportion of heads converges in probability to $\mu_Y = 0.5$, the probability of tossing head in a single coin toss,

$$R_n \xrightarrow{p} \mu_Y = 0.5 \quad \text{as} \quad n \to \infty.$$

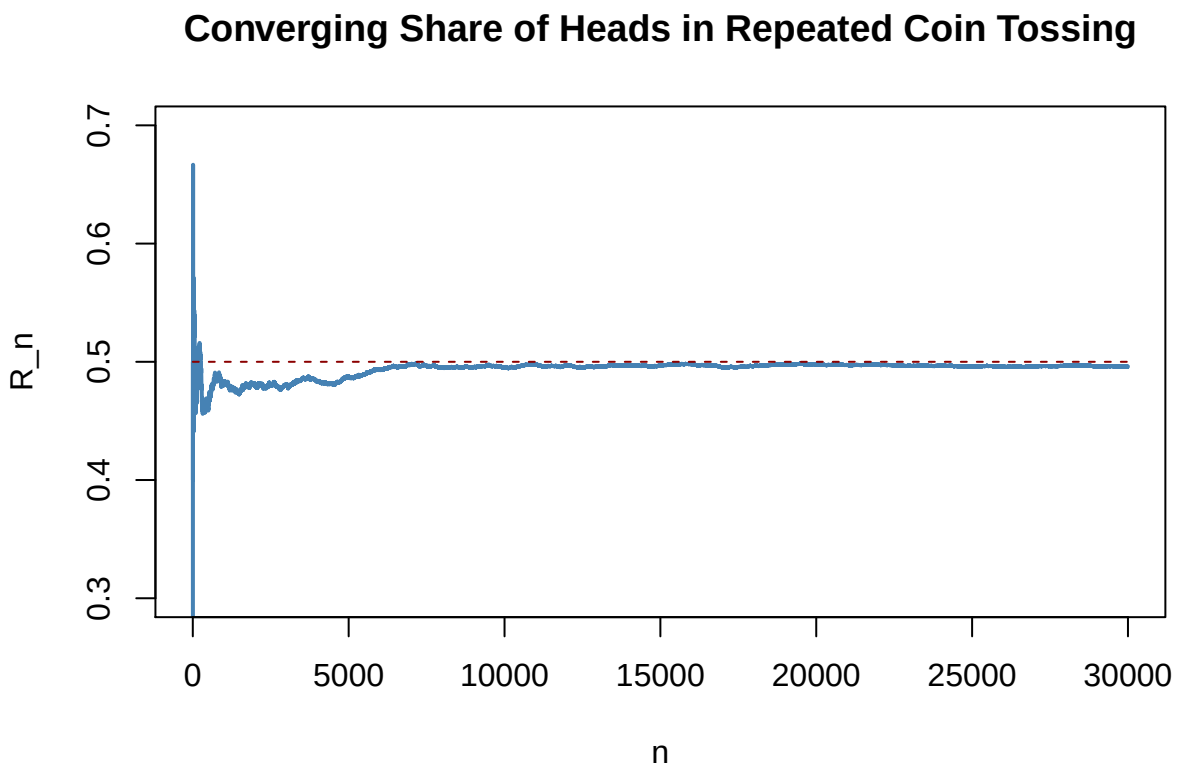
The following application simulates 1000 coin tosses with a fair coin and computes the fraction of heads observed for each additional toss interactively. The result is a random path that, as stated by the law of large numbers, shows a tendency to approach the vaule of 0.5 as n grows.

We can use R to compute and illustrate such paths by simulation. The procedure is as follows:

1. Sample `N` observations from the Bernoulli distribution e.g. using `sample()`.
2. Calculate the proportion of heads R_n as in (2.5). A way to achieve this is to call `cumsum()` on the vector of observations `Y` to obtain its cumulative sum and then divide by the respective number of observations.

We continue by plotting the path and also add a dashed line for the benchmark $R_n = p = 0.5$.

```
# set random seed
set.seed(1)

# Set number of coin tosses and simulate
N <- 30000
Y <- sample(0:1, N, replace =T)

# Calculate R_n for 1:N
S <- cumsum(Y)
R <- S/(1:N)

# Plot the path.
plot(R,
     ylim=c(0.3, 0.7),
     type = "l",
     col = "steelblue",
     lwd = 2,
     xlab = "n",
     ylab = "R_n",
     main = "Converging Share of Heads in Repeated Coin Tossing"
     )

# Add a dashed line for R_n = 0.5
lines(c(0,N),
      c(0.5, 0.5),
      col = "darkred",
      lty = 2,
      lwd = 1
      )
```



There are several things to be said about this plot.

- The blue graph shows the observed proportion of heads when tossing a coin n times.
- Since the Y_i are radnom variables, R_n is a random variate, too. The path depicted is only one of many possible realisations of R_n as it is determined by the 30000 observations sampled from the Bernoulli distribution. Thus, the code chunk above produces a differnt path every time You execute it (try this!).
- If the number of coin tosses n is small, we observe the proportion of heads to be anything but close to its theoretical value, $\mu_Y = 0.5$. However, as more and more observation are included in the sample we find that the path stabilizes in neighbourhood of 0.5. This is the message to take away: the average of multiple trials shows a clear tendency to converge to its expected value as the sample size increases, just as claimed by the law of large numbers.

Key Concept 2.6

The Central Limit Theorem

Suppose that $Y_1, \dots, Y_n$ are independently and identically distributed random variables with expectation $E(Y_i) = \mu_Y$ and variance $\text{Var}(Y_i) = \sigma_Y^2$, $0 < \sigma_Y^2 < \infty$. The central limit theorem states that, if the sample size n goes to infinity, the distribution of the scaled (by $\sqrt{n}$) standardized sample average

$$\frac{\overline{Y} - \mu_Y}{\sigma_{\overline{Y}}} = \frac{\overline{Y} - \mu_Y}{\sigma_Y / \sqrt{n}}$$

becomes arbitrarily well approximated by the standard normal distribution.

According to the central limit theorem, the distribution of the sample mean $\overline{Y}$ of the bernoulli distributed random variables Y_i, $i = 1, ..., n$ is well approximated by the normal distribution with parameters $\mu_Y = p = 0.5$ and $\sigma_{\overline{Y}}^2 = p(1-p) = 0.25$ for large n. Consequently, for the standardized sample mean we conclude that the ratio

$$\frac{\overline{Y} - 0.5}{0.5/\sqrt{n}} \tag{2.6}$$

should be well approximated by the standard normal distribution $N(0,1)$. We employ another simulation study to demonstrate this graphically. The idea is as follows.

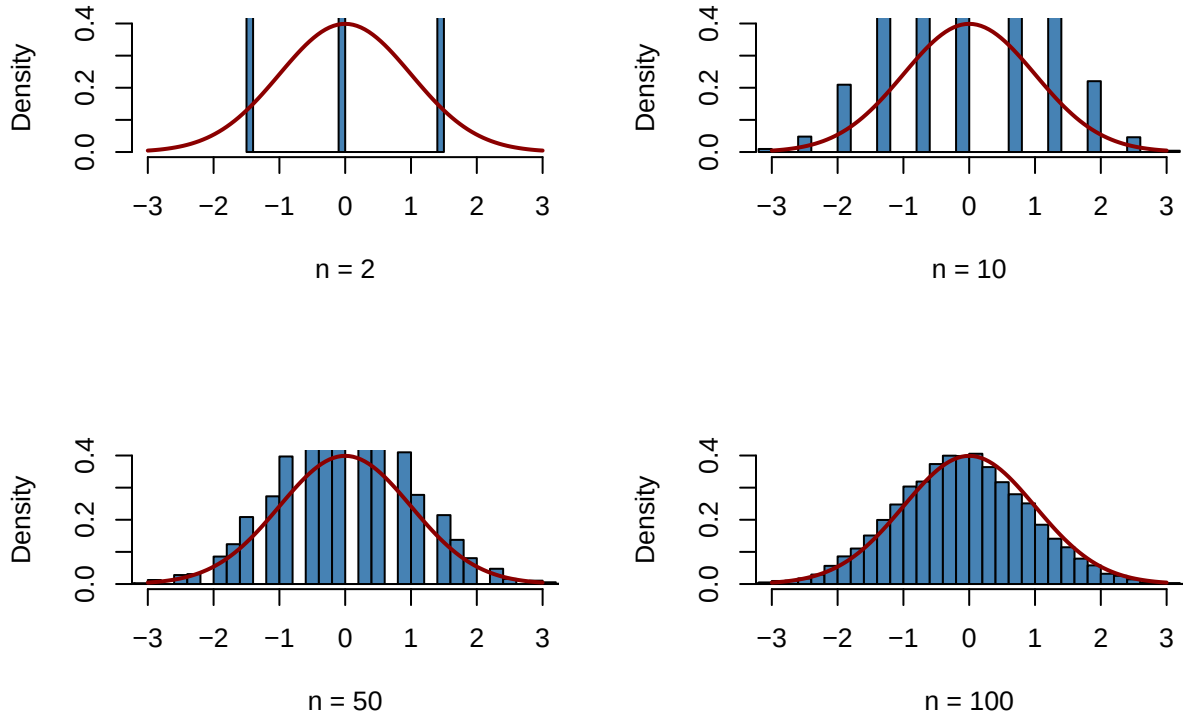
Draw a large number of random samples, 10000 say, of size n from the Bernoulli distribution and compute the sample averages. Standardize the averages as shown in (2.6). Next, visualize the distribution of the generated standardized sample averages by means of a density histogram and compare to the standard normal distribution. Repeat this for different sample sizes n to see how increasing the sample size n impacts the simulated distribution of the averages.

In R, we realized this as follows:

1. We start by defining that the next four subsequently generated figures shall be drawn in a 2×2 array such that they can be easily compared. This is done by calling `par(mfrow = c(2, 2))` before the figures are generated.
2. We define the number of repetitions `reps` as 10000 and create a vector of sample sizes named `sample.sizes`. We consider samples of sizes 2, 10, 50 and 100.
3. Next, we combine two `for()` loops to simulate the data and plot the distributions. The inner loop generates 10000 random samples, each consisting of `n` observations that are drawn from the bernoulli distribution, and computes the standardized averages. The outer loop executes the inner loop for the different sample sizes `n` and produces a plot for each iteration.

```
# Subdivide the plot panel into a 2-by-2 array
par(mfrow = c(2, 2))

# Set number of repetitions and the sample sizes
reps <- 10000
sample.sizes <- c(2, 10, 50, 100)

# outer loop (loop over the sample sizes)
  for (n in sample.sizes) {

    samplemean <- rep(0,reps) #initialize the vector of sample menas
    stdsamplemean <- rep(0,reps) #initialize the vector of standardized sample menas

# inner loop (loop over repetitions)
    for (i in 1:reps) {
      x <- rbinom(n,1,0.5)
      samplemean[i] <- mean(x)
      stdsamplemean[i] <- sqrt(n)*(mean(x)-0.5)/0.5
    }

# plot the histogram and overlay it with the N(0,1) density for every iteration
    hist(stdsamplemean,
         col = "steelblue",
         breaks = 40,
         freq = FALSE,
         xlim=c(-3, 3),
         ylim = c(0, 0.4),
         xlab = paste("n =", n),
         main = ""
         )

    curve(dnorm(x),
          lwd = 2,
          col="darkred",
```

```
        add = TRUE
        )
}
```



We see that the simulated sampling distribution of the standardized average tends to deviate strongly from the standard normal distribution if the sample size is small, e.g. for $n = 5$ and $n = 10$. However as n grows, the histograms are approaching the bell shape of a standard normal and we can be confident that the approximation works quite well as seen for $n = 100$.

3.4 Exercises

1. Sampling

Suppose you are the lottery fairy in a weekly lottery, where 6 out of 49 unique numbers are drawn.

Instructions:

- Draw the winning numbers for this week.

set.seed(123) # Draw the winning numbers for this week

set.seed(123) # Draw the winning numbers for this week sample(1:49, size = 6)

test_output_contains("set.seed(123);sample(1:49, 6)")

Hints:

- You can use the function sample() to draw random numbers.
- The set of elements to be sampled from is $\{1, ..., 49\}$.
- In sample() you can specify the sample size using the argument size.

2. Probability Density Function

Consider a random variable X with probability density function (pdf)

$$f_X(x) = \frac{x}{4} e^{-\frac{x^2}{8}}, \quad x \geq 0.$$

Instructions:

- Define the pdf from above as a function f(). Use exp() for the natural exponential function.
- Check whether the function you have defined is indeed a pdf.

```
<code data-type="sample-code">

# Define the pdf

# Integrate f over the domain

</code>

<code data-type="solution">

# Define the pdf f <- function(x){x/4*exp(-x^2/8)}

# Integrate f over the domain integrate(f, 0, Inf)$value

<code data-type="sct">

test_function_definition("f", function_test = { test_expression_result("f(1)") test_expression_result("f(4)") test_expression_result("f(10)") test_expression_result("f(100)") }) test_function('integrate', args = c('f', 'lower', 'upper'), eval = c(F, T, T)) test_output_contains('1', incorrect_msg = 'Did you access the value of the integral?')
```

Hints:

- Use function(x) {…} to define a function which takes the argument x.
- In order for f() to be a pdf, its integral over the whole domain has to be equal to 1, that is, $\int_0^\infty f_X(x)dx = 1$.
- integrate() performs integration in R and with $value you can access the numerical value of the computed integral.

3. Expected Value and Variance

In this exercise you have to compute the expected value and the variance of the random variable X considered in the previous exercise.

The pdf f() from the previous exercise is available in your working environment.

Instructions:

- Define a suitable function ex() which integrates to the expected value of X.
- Compute the expected value of X. Store the result in expected_value.
- Define a suitable function ex2() which integrates to the expected value of X^2.
- Compute the variance of X. Store the result in variance.

```
<code data-type="pre-exercise-code">
f <- function(x){(x/4)*exp(-x^2/8)}
</code>

<code data-type="sample-code">

# Define the function ex

# Compute the expected value of X

# Define the function ex2

# Compute the variance of X

</code>

<code data-type="solution">

# Define the function ex ex <- function(x){x*f(x)}

# Compute the expected value of X expected_value <- integrate(ex, 0, Inf)$value

# Define the function ex2 ex2 <- function(x){x^2*f(x)}

# Compute the variance of X variance <- integrate(ex2, 0, Inf)$value - expected_value^2

<code data-type="sct">

test_function_definition("ex", function_test = { test_expression_result("ex(1)") test_expression_result("ex(4)")
test_expression_result("ex(10)") test_expression_result("ex(100)") }) test_function('integrate', args =
c('f', 'lower', 'upper'), eval = c(F, T, T)) test_object('expected_value') test_function_definition("ex2",
function_test = { test_expression_result("ex2(1)") test_expression_result("ex2(4)") test_expression_result("ex2(10)")
test_expression_result("ex2(100)") }) test_function('integrate', args = c('f', 'lower', 'upper'), eval = c(F,
T, T)) test_object('variance')
```

Hints:

- The expected value of X is defined as $\mathbb{E}[X] = \int_0^\infty x f_X(x) dx$.
- The value of an integral computed by integrate() can be obtained via $value.
- The variance of X is defined as $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, where $\mathbb{E}[X^2] = \int_0^\infty x^2 f_X(x) dx$.

4. Standard Normal Distribution I

Let $Z \sim \mathcal{N}(0, 1)$.

Instructions:

- Compute $\phi(3)$, that is, the value of the standard normal density at $c = 3$.

```
<code data-type="sample-code">

# Compute the value of the standard normal density at c=3

</code>

<code data-type="solution">

# Compute the value of the standard normal density at c=3 dnorm(3)

</code>

<code data-type="sct">
```

test_function_result(‘dnorm’)

Hints:

- Values of $\phi(\cdot)$ can be computed using dnorm().

5. Standard Normal Distribution II

Let $Z \sim \mathcal{N}(0, 1)$.

Instructions:

- Compute $P(|Z| \leq 1.64)$ by using the function pnorm().

```
<code data-type="sample-code">
```

Compute the probability

```
</code>
```

```
<code data-type="solution">
```

Compute the probability pnorm(1.64)-pnorm(-1.64)

```
</code>
```

```
<code data-type="sct">
test_function("pnorm")
```

test_output_contains(“1 - 2*pnorm(-1.64)“, incorrect_msg =”Not correct¡‘)

Hints:

- $P(|Z| \leq z)$ equals $P(-z \leq Z \leq z)$.
- Probabilities of the form $P(a \leq Z \leq b)$ can be computed as $P(Z \leq b) - P(Z \leq a) = F_Z(b) - F_Z(a)$ where $F_Z(\cdot)$ is the cumulative distribution function of Z. Alternatively you may exploit the symmetry of the standard normal distribution.

6. Normal Distribution I

Let $Y \sim \mathcal{N}(5, 25)$.

Instructions:

- Compute the 99% quantile of the given distribution. That is find y such that $\Phi(y) = 0.99$.

Compute the 99% quantile of a normal distribution with mu = 5 and sigma^2 = 25.

Compute the 99% quantile of a normal distribution with mu = 5 and sigma^2 = 25. qnorm(0.99, mean = 5, sd = sqrt(25))

test_function_result(‘qnorm’)

Hints:

- You can compute quantiles of the normal distribution using the function qnorm().
- Besides the quantile to be computed you have to specify μ and σ^2. This is done via the arguments mean and sd. However note that sd sets the standard deviation, **not** the variance.

7. Normal Distribution II

Let $Y \sim \mathcal{N}(2, 12)$.

Instructions:

- Generate 10 random numbers from this distribution.

set.seed(123) # Generate 10 random numbers from the given distribution.

set.seed(123) # Generate 10 random numbers from the given distribution. rnorm(10, mean = 2, sd = sqrt(12))

test_output_contains("set.seed(123);rnorm(10, mean = 2, sd = sqrt(12))")

Hints:

- You can use rnorm() to draw random numbers from a normal distribution.
- Besides the number of draws you have to specify μ and σ^2. This can be done via the arguments mean and sd, however note that sd requires the standard deviation and **not** the variance.

8. Chi-squared Distribution I

Let $W \sim \chi^2_{10}$.

Instructions:

- Plot the corresponding probability density function (pdf) using the function curve(). Specify the range of x-values as $[0, 25]$ via the argument xlim.

<code data-type="sample-code">

Plot the pdf of a chi^2 random variable with df = 10

</code>

<code data-type="solution">

Plot the pdf of a chi^2 random variable with df = 10 curve(dchisq(x, df = 10), xlim = c(0, 25))

</code>

<code data-type="sct">

test_function('curve', args = c('expr', 'xlim'), eval = c(NA, T)) test_student_typed('dchisq')

Hints:

- curve() expects the function with their respective parameter(s) (here: degrees of freedom df) as an argument.
- The range of x-values in xlim can be passed as a vector consisting of both endpoints.

9. Chi-squared Distribution II

Let X_1, X_2 be two independent normally distributed random variables with $\mu = 0$ and $\sigma^2 = 15$.

Instructions:

- Compute $P(X_1^2 + X_2^2 > 10)$.

<code data-type="sample-code">

Compute the probability

</code>

<code data-type="solution">

Compute the probability pchisq(10/15, df = 2, lower.tail = F)

</code>

<code data-type="sct">

test_output_contains('pchisq(10/15, df = 2, lower.tail = F)', incorrect_msg = "Not correct!")

Hints:

- Note that both random variables are not $\mathcal{N}(0,1)$ but $\mathcal{N}(0,\sigma^2)$ distributed. Hence you have to scale appropriately. Afterwards you can use pchisq() to compute the probability.

10. Student t Distribution I

Let $X \sim t_{10000}$ and $Z \sim \mathcal{N}(0,1)$.

Instructions:

- Compute the 95% quantile of both distributions. What is noticeable?

<code data-type="sample-code">

Compute the 95% quantile of a t distribution with 10000 degrees of freedom

Compute the 95% quantile of a standard normal distribution

</code>

<code data-type="solution">

Compute the 95% quantile of a t distribution with 10000 degrees of freedom qt(0.95, df = 10000)

Compute the 95% quantile of a standard normal distribution qnorm(0.95)

Both values are very close to each other. This is not surprising as for sufficient large degrees of freedom the t distribution can be approximated by the standard normal distribution.

<code data-type="sct">

test_function_result('qt') test_function_result('qnorm') success_msg('Correct! A t-distributed RV converges to a standard normal as the df get large.')

Hints:

- You can use qt() and qnorm() to compute quantiles of the given distributions.
- For the t distribution you have to specify the degrees of freedom df.

11. Student t Distribution II

Let $X \sim t_1$. Once the session has initialized you will see the plot of the corresponding probability density function (pdf).

Instructions:

- Generate 1000 random numbers from this distribution and assign them to the variable x.

- Compute the sample mean of x. Can you explain the result?

<code data-type="pre-exercise-code">

curve(dt(x, df = 1), xlim = c(-4, 4))

<code data-type="sample-code">

set.seed(123) # Generate 1000 random numbers from the given distribution. Assign them to the variable x.

Compute the sample mean of x.

</code>

<code data-type="solution">

set.seed(123) # Generate 1000 random numbers from the given distribution. Assign them to the variable x.
x <- rt(1000, df = 1)

Compute the sample mean of x. mean(x)

Although a t distribution with M = 1 is, as every other t distribution, symmetric around zero it actually has no expectation. This explains the highly non-zero value for the sample mean.

<code data-type="sct">

test_object('x') test_function_result('mean') success_msg("Right! The expectation is not defined for a t distributed RV with one degree of freedom.")

Hints:

- You can use rt() to draw random numbers from a t distribution.
- Note that the t distribution is fully parameterized through the degree(s) of freedom. Specify them via the argument df.
- To compute the sample mean of a vector you can use the function mean().

12. F Distribution I

Let $Y \sim F(10, 4)$.

Instructions:

- Plot the quantile function of the given distribution using the function curve().

<code data-type="sample-code">

Plot the quantile function of the given distribution

</code>

<code data-type="solution">

Plot the quantile function of the given distribution curve(qf(x, df1 = 10, df2 = 4))

</code>

<code data-type="sct">

test_function('curve', args = 'expr', eval = NA) test_student_typed('qf')

Hints:

- curve() expects the function with their respective parameters (here: degrees of freedom df1 and df2) as an argument.

13. F Distribution II

Let $Y \sim F(4, 5)$.

Instructions:

- Compute $P(1 < Y < 10)$ by integration of the pdf.


```
<code data-type="sample-code">
```


Compute the probability by integration


```
</code>

<code data-type="solution">
```


Compute the probability by integration integrate(df, lower = 1, upper = 10, df1 = 4, df2 = 5)$value


```
</code>

<code data-type="sct">
```


test_function('integrate', args = c('f', 'lower', 'upper', 'df1', 'df2')) test_output_contains('integrate(df, lower = 1, upper = 10, df1 = 4, df2 = 5)$value')

Hints:

- Besides the function that has to be integrated you have to specify lower and upper bounds.
- The additional parameters of the distribution (here df1 and df2) also have to be passed **inside** the call of integrate().
- The value of the integral can be obtained via $value.

Chapter 4

A Review of Statistics using R

This section reviews important statistical concepts:

- Estimation
- Hypothesis testing
- Confidence intervals

Since these types of statistical methods are heavily used in econometrics, we will discuss them in the context of inference about an unknown population mean and discuss several applications in R.

4.1 Estimation of the Population Mean

Key Concept 3.1

Estimators and Estimates

Estimators are functions of sample data that are drawn randomly from an unknown population. Estimates are numerical values computed by estimators based on the sample data. Estimators are random variables because they are functions of *random* data. Estimates are nonrandom numbers.

Think of some economic variable, for example hourly earnings of college graduates, denoted by Y. Suppose we are interested in μ_Y the mean of Y. In order to exactly calculate μ_Y we would have to interview every graduated member of the working population in the economy. We simply cannot do this for time and cost reasons. However, we could draw a random sample from n i.i.d. observations $Y_1, \dots, Y_n$ and estimate μ_Y using one of the simplest estimators in the sense of Key Concept 3.1 one can think of:

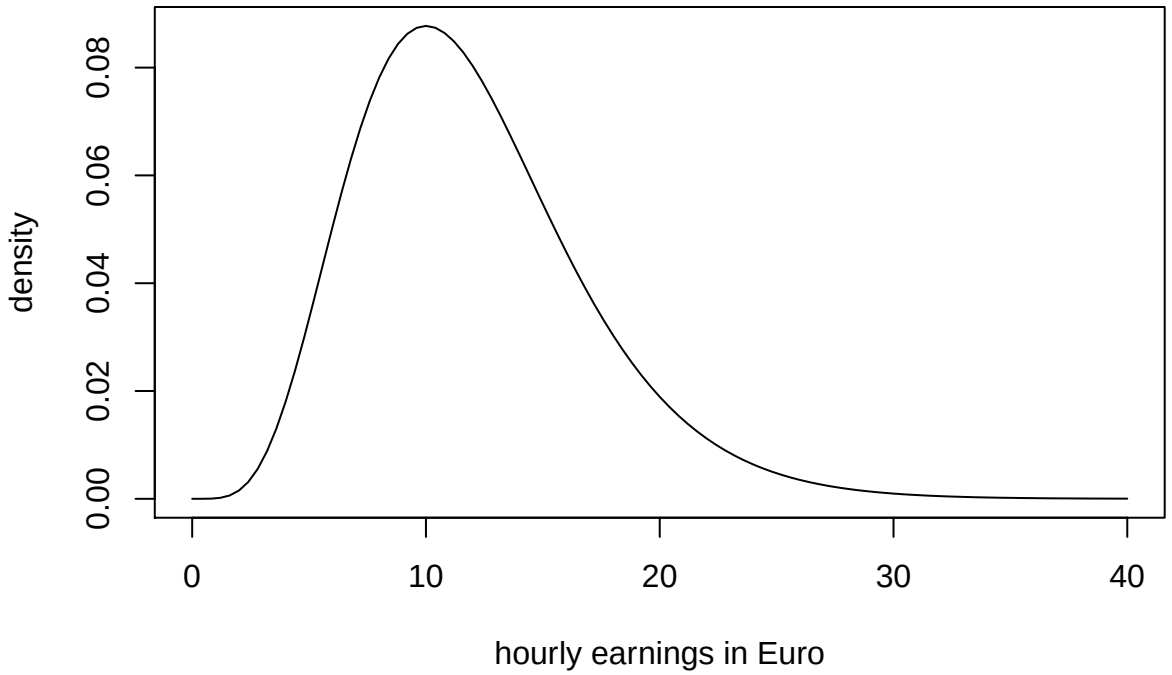
$$\overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i,$$

the sample mean of Y. Then again, we could use an even simpler estimator for μ_Y: the very first observation in the sample, Y_1. Is Y_1 a good estimator? For now, assume that

$$Y \sim \chi^2_{12}$$

which is not too unreasonable as the measure is nonnegative and we expect many hourly earnings to be in a range of 5 to 15. Moreover, it is common for income distributions to be skewed to the right.

```
# plot the chi_12^2 distribution
curve(dchisq(x, df=12),
      from = 0,
      to = 40,
      ylab = "density",
      xlab = "hourly earnings in Euro"
      )
```



We draw a sample of $n = 100$ observations and take the first observation Y_1 as an estimate for μ_Y

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

# sample from the chi_12^2 distribution, keep only the first observation
rchisq(n = 100, df = 12)[1]
```

```
## [1] 8.257893
```

The estimate 8.26 is not too far away from $\mu_Y = 12$ but it is somewhat intuitive that we could do better: the estimator Y_1 discards a lot of information and its variance is the population variance:

$$\text{Var}(Y_1) = \text{Var}(Y) = 2 \cdot 12 = 24$$

This brings us to the following question: What is a 'good' estimator in the first place? This question is tackled in Key Concepts 3.2 and 3.3

Key Concept 3.2

Bias, Consistency and Efficiency

Disirable characteristics of an estimator are unbiasedness, consitency and Efficiency.

Unbiasedness:
If the mean of the sampling distribution of some estimator $\hat{\mu}_Y$ for the population mean μ_Y equals μ_Y

$$E(\hat{\mu}_Y) = \mu_Y$$

we say that the estimator is unbiased for μ_Y. The *bias* of $\hat{\mu}_Y$ is 0:

$$E(\hat{\mu}_Y) - \mu_Y = 0$$

Consistency:

We want the uncertainty of the estimator μ_Y to decrease as the number of observations in the sample grows. More precisely, we want the proabability that the estimate $\hat{\mu}_Y$ falls within a small interval of the true value μ_Y to get increasingly closer to 1 as n grows. We write this as

$$\hat{\mu}_Y \xrightarrow{p} \mu_Y.$$

Variance and efficiency:

We want the estimator to be efficient. Suppose we have two estimators, $\hat{\mu}_Y$ and $\widetilde{\mu}_Y$ and for some given sample size n it holds that

$$E(\hat{\mu}_Y) = E(\widetilde{\mu}_Y) = \mu_Y$$

but

$$\text{Var}(\hat{\mu}_Y) < \text{Var}(\widetilde{\mu}_Y).$$

We then would prefer to use $\hat{\mu}_Y$ as it has a lower variance than $\widetilde{\mu}_Y$, meaning that $\hat{\mu}_Y$ is more *efficient* in using the information provided by the observations in the sample.

Key Concept 3.3

Efficiency of $\overline{Y}$: The BLUE property

Let $\hat{\mu}_Y$ be a linear and unbiased estimator of μ_Y in the fashion of

$$\hat{\mu}_Y = \frac{1}{n}\sum_{i=1}^{n} a_i Y_i$$

with nonrandom constants a_i. We see that $\hat{\mu}_Y$ is a weighted average of the Y_i and the a_i are weights. For these type of estimators, $\overline{Y}$ with $a_i = 1$ for all $i = 1, \dots, n$ is the most efficient estimator. We say that $\overline{Y}$ is the BestLinear Unbiased Estimator (BLUE).

4.2 Properties of the Sample Mean

To examine properties of the sample mean as an estimator for the corresponding population mean, consider the following R example.

We generate a population `pop` which consists observations Y_i , $i = 1, \dots, 10000$ that origin from a normal distribution with mean $\mu = 10$ and variance $\sigma^2 = 1$. To investigate how the estimator $\hat{\mu} = \bar{Y}$ behaves we can draw random samples from this population and calculate $\bar{Y}$ for each of them. This is easily done by making use of the function `replicate()`. Its argument `expr` is evaluated `n` times. In this case we draw samples of sizes $n = 5$ and $n = 25$, compute the sample means and repeat this exactly $n = 25000$ times.

For comparison purposes we store results for the estimator Y_1, the first observation in a sample for a sample of size 5 separately.

```
# generate a fictive population
pop <- rnorm(10000, 10, 1)

# sample form pop and estimate the mean
est1 <- replicate(expr = mean(sample(x = pop, size = 5)), n = 25000)

est2 <- replicate(expr = mean(sample(x = pop, size = 25)), n = 25000)

fo <- replicate(expr = sample(x = pop, size = 5)[1], n = 25000)
```

Check that `est1` and `est2` are vectors of length 25000:

```
# check if object type is vector
is.vector(est1)
```

```
## [1] TRUE
```

```
is.vector(est2)
```

```
## [1] TRUE
```

```
# check lengths
length(est1)
```

```
## [1] 25000
```

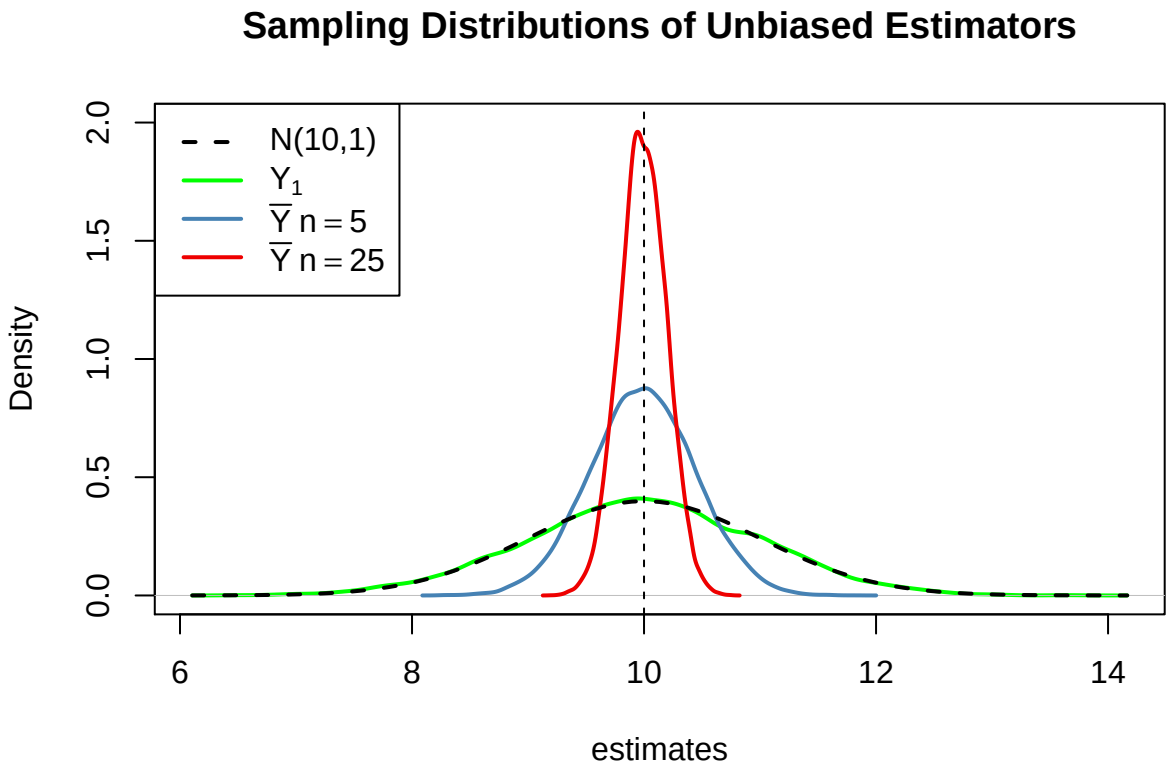
```
length(est2)
```

```
## [1] 25000
```

The code chunk below produces a plot of the sampling distributions of the estimators $\bar{Y}$ and Y_1 on the basis of the 25000 samples in each case. We also plot a curve depicting the density function of the $N(10, 1)$ distribution.

```
# plot density estimate Y_1
plot(density(fo),
      col = 'green',
      lwd = 2,
      ylim = c(0,2),
      xlab = 'estimates',
      main = 'Sampling Distributions of Unbiased Estimators'
      )

# add density estimate for the distribution of the sample mean with n=5 to the plot
lines(density(est1),
     col = 'steelblue',
     lwd = 2,
     bty = 'l'
     )

# add density estimate for the distribution of the sample mean with n=25 to the plot
lines(density(est2),
      col = 'red2',
      lwd = 2
      )

# add a vertical line marking the true parameter
abline(v = 10, lty = 2)
```

```
# add N(10,1) density to the plot
curve(dnorm(x, mean=10),
      lwd = 2,
      lty = 2,
      add = T
      )

# add a legend
legend("topleft",
       legend = c("N(10,1)",
                  expression(Y[1]),
                  expression(bar(Y) ~ n==5),
                  expression(bar(Y) ~ n==25)
                  ),
       lty = c(2, 1, 1, 1),
       col = c('black','green', 'steelblue', 'red2'),
       lwd = 2
       )
```



At first, notice how *all* sampling distributions (represented by the solid lines) are centered around $\mu = 10$. This is evidence for the *unbiasedness* of Y_1 and $\overline{Y}$. Of course, the theoretical density the $N(10, 1)$ distribution is centered at 10, too.

Next, have a look add the spread of the sampling distributions. Several things are remarkable:

- First, the sampling distribution of Y_1 (green curve) tracks the density of the $N(10, 1)$ distribution (black dashed line) pretty closely In fact, the sampling distribution of Y_1 *is* the $N(10, 1)$ distribution. This is less surprising if You keep in mind that Y_1 estimator does nothing but reporting an observation that is randomly selected from a population with $N(10, 1)$ distribution. Hence, $Y_1 \sim N(10, 1)$. Note that this result is invariant to the sample size n: the sampling distribution of Y_1 is always the population

distribution, no how large the sample is.

- Second, both sampling distributions of $\overline{Y}$ show less dispersion than the sampling distribution of Y_1. This means that $\overline{Y}$ has a lower variance than Y_1. In view of Key Concepts 3.2 and 3.3, we find that $\overline{Y}$ is a more efficient estimator than Y_1. In fact, one can show that this holds for all $n > 1$.
- Third, $\overline{Y}$ shows a behaviour that is termed consistency (see Key Concept 3.2). Notice that the blue and the red density curves are much more concentrated around $\mu = 10$ then the green one. As the number of observations is increased from 1 to 5, the sampling distribution tightens around the true parameter. This effect is more dominant as the sample size is increased to 25. This implies that the probability of obtaining estimates that are close to the true value increases with n.

A more precise way to express consitency of an estimator $\hat{\mu}$ for a parameter μ is

$$P(|\hat{\mu} - \mu| < \epsilon) \xrightarrow[n \to \infty]{p} 1 \quad \text{for any} \quad \epsilon > 0.$$

This expression says that the probability of observing a deviation from the true value μ that is smaller than some arbitrary $\epsilon > 0$ converges to 1 as n grows. Note that consistency does not require unbiasedness:

We encourage You to go ahead and modify the code. Try out different values for the sample size and see how the sampling distribution of $\overline{Y}$ changes!

$\overline{Y}$ is the least squares estimator of μ_Y

Assume You have some observations $Y_1, \dots, Y_n$ on $Y \sim N(10, 1)$ (which is unknown) and would like to find an estimator m that predicts the observations as good as possible where good means to choose m such that the total deviation between the predicted value and the observed values is small. Mathematically this means we want to find an m that minimizes

$$\sum_{i=1}^{n} (Y_i - m)^2. \tag{4.1}$$

Think of $Y_i - m$ as the comitted mistake when predicting Y_i by m. We could just as well minimize the sum of absolute deviations from m but minimizing the sum of squared deviations is mathematically more convenient and leads, roughly speaking, to the same result. That is why the estimator we are looking for is called the *least squares estimator*. As It turns out $m = \overline{Y}$, the estimator of $\mu_Y = 10$ is this wanted estimator.

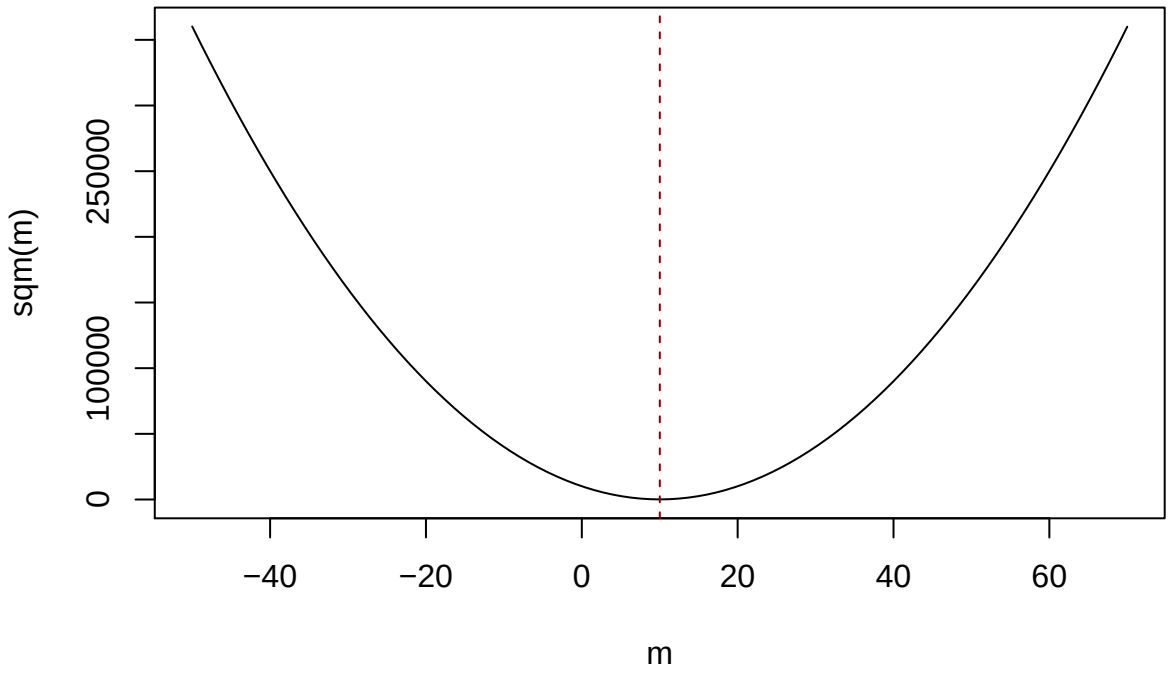
We can show this by generating a random sample of fair size and plotting (4.1) as a function of m.

```
# define the function and vectorize it
sqm <- function(m) {
 sum((y-m)^2)
}
sqm <- Vectorize(sqm)

# draw random sample and compute the mean
y <- rnorm(100, 10, 1)
mean(y)
```

```
## [1] 10.00543
```

```
# plot the objective function
curve(sqm(x),
      from = -50,
```

```
      to = 70,
      xlab = "m",
      ylab ="sqm(m)"
      )

# add vertical line at mean(y)
abline(v = mean(y),
       lty = 2,
       col = "darkred"
       )
```



Notice that (4.1) is a quadratic function so there is only one minimum. The plot shows that this minimum lies exactly at the sample mean of the sample data.

Some R functions can only interact with functions that take a vector as input evaluate the function body on every values of the vector, for example curve(). We call such functions vectorized functions and it is often a good idea to write vectorized functions although this is cumbersome in some cases. Having a vectorized function in R is never a drawback since these functions work on both single values and vectors.
Let us look at the function sqm() which is nonvectorized

```
sqm <- function(m) {
    sum((y-m)^2) #body of the function
}
```

Providing e.g. c(1,2,3) as the argument m would cause an error since then the operation y-m is invalid: the vecors y and m are of incompatible dimensions. This is why we cannot use sqm() in conjunction with curve().
Here comes Vectorize() into play. It generates a vectorized version of a non-vectorized function.

Why Random Sampling is important

So far, we assumed (somtimes implicitly) that observed data $Y_1, \dots, Y_n$ are the result of a sampling process that satisfies the assumption of i.i.d. random sampling. It is very important that this assumption is fulfilled when estimating a population mean using $\overline{Y}$. If this is not the case, estimates are biased.

Let us fall back to `pop`, the fictive population of 10000 observations and compute the population mean μ_{pop}:

```
# compute the population mean of pop
mean(pop)
```

```
## [1] 9.992604
```

Next we sample 10 observations from `pop` with `sample()` and estimate μ_{pop} using $\overline{Y}$ repeatedly. However this time we use a sampling scheme that deviates from simple random sampling: instead of ensuring that each member of the population has the same chance to end up in a sample, we assign a higher probability of beeing sampled to the 2500 smallest observations of the population by setting the argument `prob` to a suitable vector of probability weights:

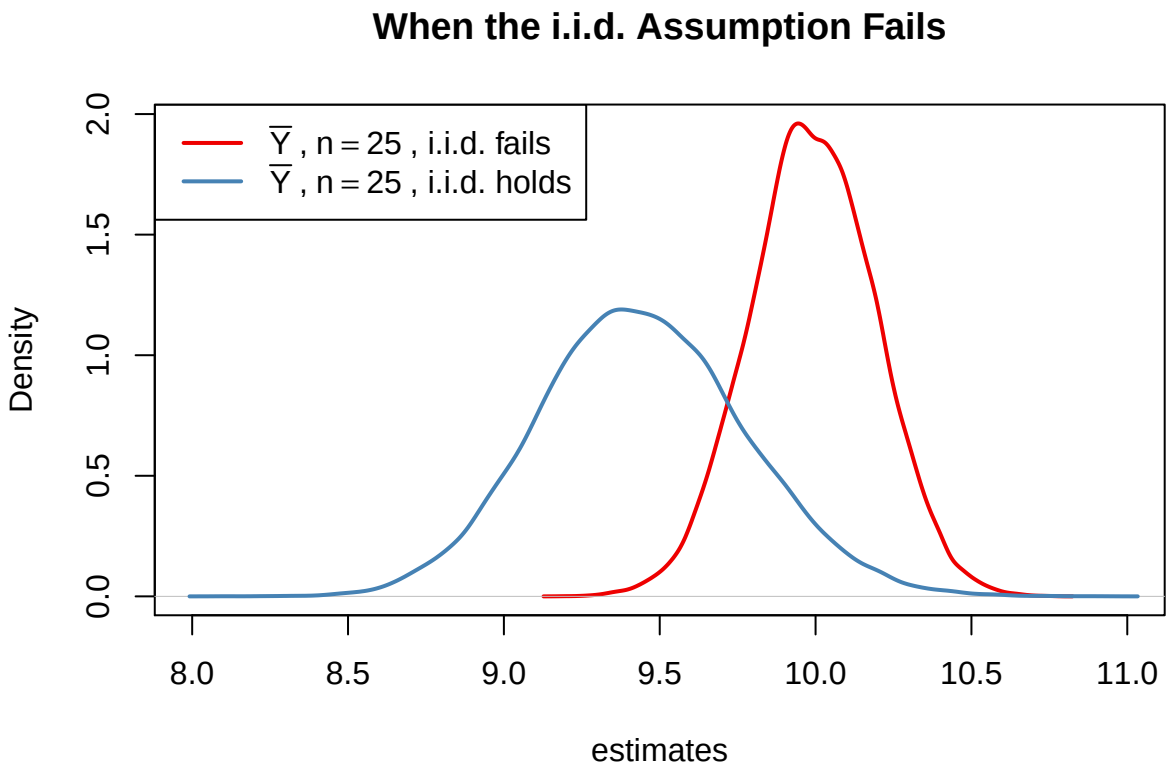
```
# simulate outcome for the sample mean when the i.i.d. assumption fails
est3 <-  replicate(n = 25000,
                   expr = mean(sample(x = sort(pop),
                                      size = 10,
                                      prob = c(rep(4,2500),rep(1,7500))
                                      )
                               )
                   )

# compute the sample mean of the outcomes
mean(est3)
```

```
## [1] 9.443454
```

Next we plot the sampling distribution of $\overline{Y}$ for this non-i.i.d. case an compare it to the sampling distribution when the i.i.d. assumption holds.

```
# sampling distribution of sample mean, i.i.d. holds, n=25
plot(density(est2),
      col = 'red2',
      lwd = 2,
      xlim = c(8,11),
      xlab = 'estimates',
      main = 'When the i.i.d. Assumption Fails'
     )

# sampling distribution of sample mean, i.i.d. fails, n=25
lines(density(est3),
      col = 'steelblue',
      lwd = 2
      )

# add a legend
legend("topleft",
       legend = c(expression(bar(Y) ~ "," ~ n==25 ~ ", i.i.d. fails"),
                  expression(bar(Y) ~ "," ~ n==25 ~ ", i.i.d. holds")
                  ),
       lty = c(1, 1),
       col = c('red2', 'steelblue'),
```

```
      lwd = 2
      )
```



We find that in this case failure of the i.i.d. assumption implies that, on average, we underestimate μ_Y using $\overline{Y}$: the corresponding distribution of $\overline{Y}$ is shifted to the left. In other words, $\overline{Y}$ is a *biased* estimator for μ_Y if the i.i.d. assumption does not hold.

4.3 Hypothesis Tests Concerning the Population Mean

In this section we briefly review concepts in hypothesis testing and discuss how to conduct hypothesis tests in R. We focus on drawing inference about an unkown population mean.

About Hypotheses and Hypothesis Testing

In a significance test we want to exploit the information contained in a random sample as evidence in favour or against a hypothesis. Essentially, hypotheses are simple question that can be answered by 'yes' or 'no'. When conducting a hypothesis test we always deal with two different hypotheses:

- The null hypothesis, denoted H_0 is the hypothesis we are interested in testing
- The alternative hypothesis, denoted H_1, is the hypothesis that holds if the null hypothesis is false

The null hypothesis that the population mean of Y equals the value $\mu_{Y,0}$ is written down as

$$H_0 : E(Y) = \mu_{Y,0}.$$

The alternative hypothesis states what holds if the null hypothesis is false. Often the alternative hypothesis chosen is the most general one,

$$H_1 : E(Y) \neq \mu_{Y,0},$$

meaning that $E(Y)$ may be anything else but the value as the null hypothesis. This is called a two-sided alternative.

For brevity, we will only consider the case of a two-sided alternative in the subsequent sections of this chapter.

p-Value

Assume that the null hypothesis is *true*. The p-value is the probability of drawing data and observing a corresponding test statistics that is at least as adverse to what is stated under the null hypothesis as the test statistic actually computed using the sample data.

In context of population mean and sample mean, this definition can be stated mathematically in the following way:

$$p\text{-value} = P_{H_0}\left[|\overline{Y} - \mu_{Y,0}| > |\overline{Y}^{act} - \mu_{Y,0}|\right] \tag{4.2}$$

In (4.2), $\overline{Y}^{act}$ is the acutally computed mean of the random sample. Visualized, the p-value is the area in the part of tails of the distribution of $\overline{Y}$ that lies beyond

$$\mu_{Y,0} \pm |\overline{Y}^{act} - \mu_{Y,0}|.$$

Consequently, in order to compute the p-value as in (4.2), knowledge about the sampling distribution of $\overline{Y}$ when the null hypothesis is true is required. However in most cases the sampling distribution of $\overline{Y}$ is unkown. Furtunately, due to the large-sample normal approximation (see chapter 3) we know that under the null hypothesis

$$\overline{Y} \sim N(\mu_{Y,0},\, \sigma^2_{\overline{Y}}) \quad , \quad \sigma^2_{\overline{Y}} = \frac{\sigma^2_Y}{n}$$

and thus

$$\frac{\overline{Y} - \mu_{Y,0}}{\sigma_Y/\sqrt{n}} \sim N(0,1).$$

So in large samples, the p-value can be computed *without* knowledge about the sampling distribution of $\overline{Y}$.

Calculating the p-Value When σ_Y Is Known

For now, let us assume that $\sigma_{\overline{Y}}$ is known. Then we can rewrite (4.2) as

$$p\text{-value} = P_{H_0}\left[\left|\frac{\overline{Y} - \mu_{Y,0}}{\sigma_{\overline{Y}}}\right| > \left|\frac{\overline{Y}^{act} - \mu_{Y,0}}{\sigma_{\overline{Y}}}\right|\right] \tag{4.3}$$

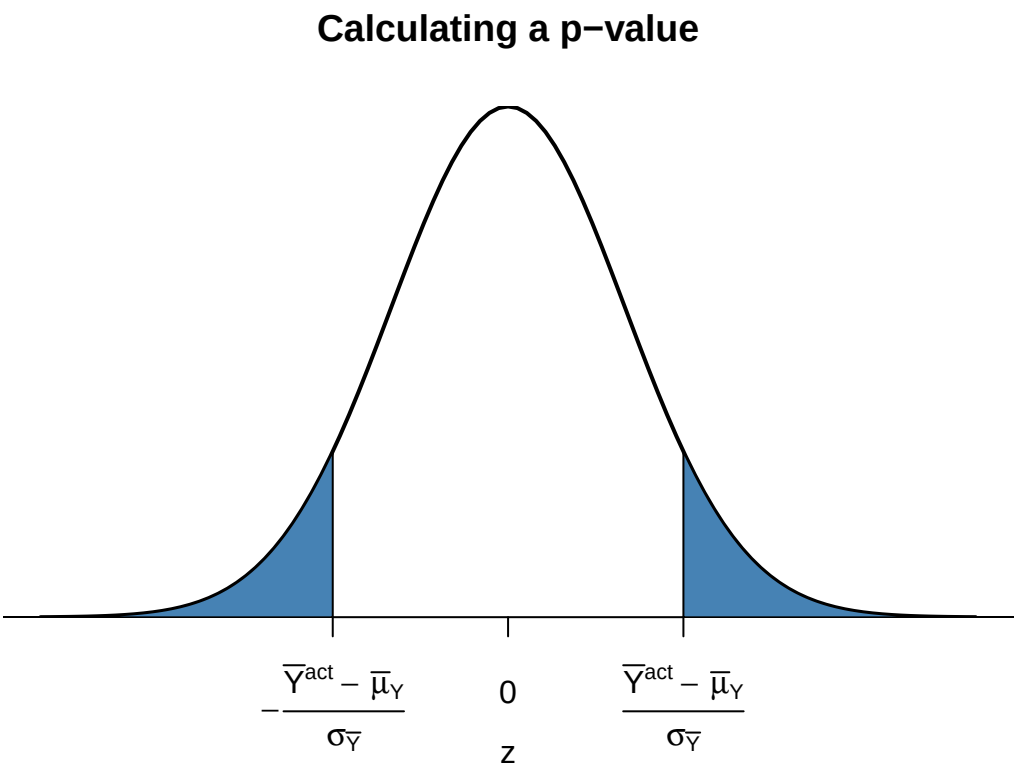
$$= 2 \cdot \Phi\left[-\left|\frac{\overline{Y}^{act} - \mu_{Y,0}}{\sigma_{\overline{Y}}}\right|\right]. \tag{4.4}$$

so the p-value can be seen as the area in the tails of the $N(0,1)$ distribution that lies beyond

$$\pm \left| \frac{\overline{Y}^{act} - \mu_{Y,0}}{\sigma_{\overline{Y}}} \right| \tag{4.5}$$

Whew, that was a lot of theory. Now we use R to visualize what is stated in (4.4) and (4.5). The next code chunck replicates figure 3.1 of the book.

```
# plot the standard normal density on the domain [-4,4]
curve(dnorm(x),
      xlim = c(-4,4),
      main = 'Calculating a p-value',
      yaxs = 'i',
      xlab = 'z',
      ylab = '',
      lwd = 2,
      axes = 'F'
      )

# add x-axis
axis(1,
     at = c(-1.5,0,1.5),
     padj = 0.75,
     labels = c(expression(-frac(bar(Y)^"act"~~~bar(mu)[Y,0],sigma[bar(Y)])),
                0,
                expression(frac(bar(Y)^"act"~~~bar(mu)[Y,0],sigma[bar(Y)])))
     )

# shade p-value/2 region in left tail
polygon(x = c(-6, seq(-6,-1.5,0.01),-1.5),
        y = c(0, dnorm(seq(-6,-1.5,0.01)),0),
        col = 'steelblue'
        )

## shade p-value/2 region in right tail
polygon(x = c(1.5, seq(1.5, 6, 0.01), 6),
        y = c(0, dnorm(seq(1.5, 6, 0.01)), 0),
        col = 'steelblue'
        )
```



Sample Variance, Sample Standard Deviation and Standard Error

If σ_Y^2 is unknown, it must be estimated. This can be done efficiently using the sample variance

$$s_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \overline{Y})^2. \tag{4.6}$$

Furthermore

$$s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}. \tag{4.7}$$

is a suitable estimator for the standard deviation of Y. In R, s_y is implemented in the function `sd()`, see `?sd`.

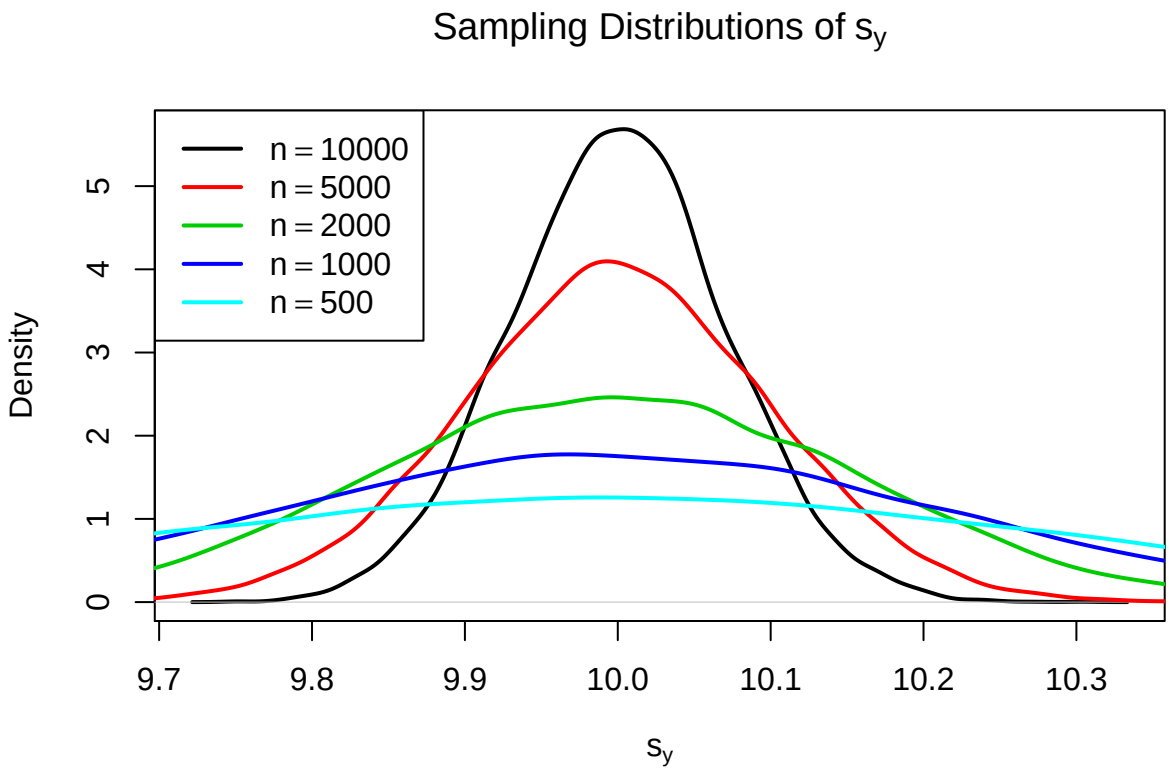
Using R we can get a notion that s_y is a consistent estimator for σ_Y, that is

$$s_Y \xrightarrow{p} \sigma_Y.$$

The idea here is to generate a large number of samples $Y_1, \dots, Y_n$ where $Y \sim N(10, 10)$, estimate σ_Y using s_y and investigate how the distribution of s_Y changes as n grows.

```
# vector of sample sizes
n <- c(10000, 5000, 2000, 1000, 500)

# sample observations, estimate using sd() and plot estimated distributions
s2_y <- replicate(n = 10000, expr = sd(rnorm(n[1], 10, 10)))
```

```
plot(density(s2_y),
     main = expression('Sampling Distributions of' ~ s[y]),
     xlab = expression(s[y]),
     lwd = 2
     )

for (i in 2:length(n)) {
  s2_y <- replicate(n = 10000, expr = sd(rnorm(n[i],10,10)))
  lines(density(s2_y),
        col=i,
        lwd=2)
}

# add a legend
legend("topleft",
       legend = c(expression(n==10000),
                  expression(n==5000),
                  expression(n==2000),
                  expression(n==1000),
                  expression(n==500)
       ),
       col = 1:5,
       lwd = 2
)
```



The plot shows that the distribution of s_Y tightens around the true value $\sigma_Y = 10$ as n increases.

The function that estimates the standard deviation of an estimator is called the *standard error of the estimator*. Key Concept 3.4 summarizes the terminology in the context of the sample mean.

Key Concept 3.4

The Standard Error of $\overline{Y}$

Take an i.i.d. sample $Y_1, \dots, Y_n$. The mean of Y can be consistently estimated using $\overline{Y}$, the sample mean of the Y_i. Since $\overline{Y}$ is a random variable, it has a sampling distribution with variance $\frac{\sigma_Y^2}{n}$.

The standard error of $\overline{Y}$, denoted $SE(\overline{Y})$ is an estimator of the standard deviation $\overline{Y}$:

$$SE(\overline{Y}) = \hat{\sigma}_{\overline{Y}} = \frac{s_Y}{\sqrt{n}}$$

The caret ($\widehat{}$) over σ indicates that $\hat{\sigma}_{\overline{Y}}$ is an estimator for $\sigma_{\overline{Y}}$.

As an example to underpin Key Concept 3.4, consider a sample of $n = 100$ i.i.d. observations of the bernoulli distributed variable Y with success probability $p = 0.1$ and thus $E(Y) = p = 0.1$ and $\text{Var}(Y) = p(1-p)$. $E(Y)$ can be estimated by $\overline{Y}$ which then has variance

$$\sigma_{\overline{Y}}^2 = p(1-p)/n = 0.0009$$

and standard deviation

$$\sigma_{\overline{Y}} = \sqrt{p(1-p)/n} = 0.03.$$

In this case the standard error of $\overline{Y}$ is given as

$$SE(\overline{Y}) = \sqrt{\overline{Y}(1-\overline{Y})/n}$$

Let verify whether $\overline{Y}$ and $SE(\overline{Y})$ estimate the respective true values on average.

```
# draw 10000 samples of size 100 and estimate the mean of Y and
# estimate the standard error of the sample mean

mean_estimates <- numeric(10000)
se_estimates <- numeric(10000)

for (i in 1:10000) {
  s <- sample(0:1,
              size = 100,
              prob = c(0.9, 0.1),
              replace = T
              )
  mean_estimates[i] <- mean(s)
  se_estimates[i] <- sqrt(mean(s)*(1-mean(s))/100)
}

mean(mean_estimates)
```

```
## [1] 0.099693
```

```
mean(se_estimates)
```

```
## [1] 0.02953467
```

Both estimators seem to be unbiased for the true parameters.

Calculating the p-value When σ_Y is Unknown

When σ_Y is unkown, the p-value for a hypothesis test about μ_Y using $\overline{Y}$ can be computed by replacing $\sigma_{\overline{Y}}$ in (4.4) by the standard error $SE(\overline{Y}) = \hat{\sigma}_Y$. Then,

$$p\text{-value} = 2 \cdot \Phi\left(-\left|\frac{\overline{Y}^{act} - \mu_{Y,0}}{SE(\overline{Y})}\right|\right).$$

This is easily done in R:

```
# sample and estimate, compute standard error and make a hypothesis
samplemean_act <- mean(
  sample(0:1,
         prob = c(0.9,0.1),
         replace = T,
         size = 100
         )
  )

SE_samplemean <- sqrt(samplemean_act * (1-samplemean_act)/100)

mean_h0 <- 0.1 #true null hypothesis

# compute the pvalue
pvalue <- 2 * pnorm(-abs(samplemean_act-mean_h0)/SE_samplemean)
pvalue
```

```
## [1] 0.5382527
```

The t-statistic

In hypothesis testing, the standardized sample average

$$t = \frac{\overline{Y} - \mu_{Y,0}}{SE(\overline{Y})} \tag{4.8}$$

is called t-statistic. This t-statistic has an important role when testing hypothesis about μ_Y. It is a prominent example of a test statistic.

Implicitly, we already have computed a t-statistic for $\overline{Y}$ in the previous code chunk.

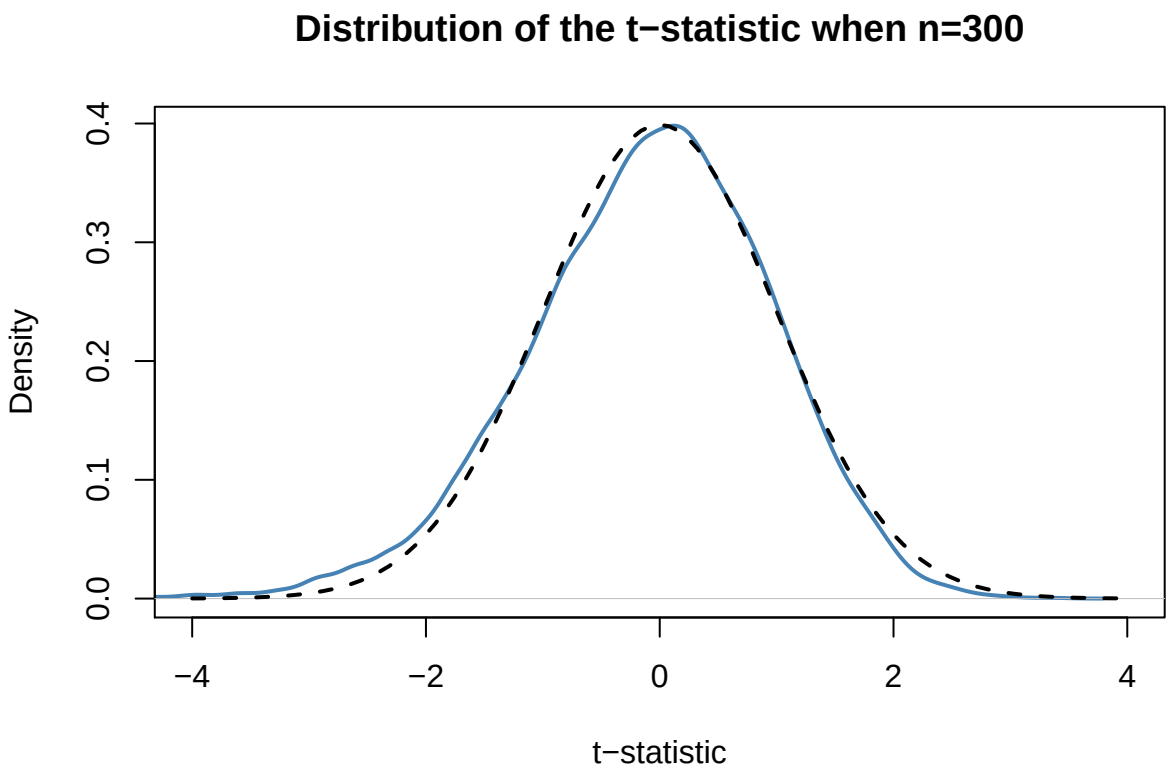
```
# compute a t-statistic for the sample mean
tstatistic <- (samplemean_act - mean_h0) / SE_samplemean
tstatistic
```

```
## [1] 0.6154575
```

Using R we can show that if $\mu_{Y,0}$ equals the true value, that is the null hypothesis is true, (4.8) is approximately distributed $N(0,1)$ when n is large.

```
# initialize empty vector for t-statistics
tstatistics <- numeric(10000)

# set sample size
n <- 300
```

```
# simulate 10000 t-statistics
for (i in 1:10000) {
  s <- sample(0:1,
              size = n,
              prob = c(0.9, 0.1),
              replace = T
              )
  tstatistics[i] <- (mean(s)-0.1)/(sqrt(mean(s)*(1-mean(s))/n))
}

# plot density and compare to N(0,1) density
plot(density(tstatistics),
     xlab = 't-statistic',
     main = 'Distribution of the t-statistic when n=300',
     lwd = 2,
     xlim = c(-4,4),
     col = 'steelblue'
     )

# N(0,1) density (dashed)
curve(dnorm(x),
      add = T,
      lty = 2,
      lwd= 2
      )
```



Judging from the plot, the normal approximation works reasonably well for the chosen sample size. This normal approximation has already been used in the definition of the p-value, see (4.8).

Hypothesis Testing with a Prespecified Significance Level

Key Concept 3.5

The Terminology of Hypothesis Testing

In hypothesis testing, two types of mistakes are possible:

1. The null hypothesis *is* rejected although it is true (α-error / type-I-error)
2. The null hypothesis *is not* rejected although it is false (β-error / type-II-error)

The significance level of the test is the probability to commit a type-I-error we are willing to accept in advance. E.g. using a prespecified significance level of 0.05, we reject the null hypothesis if and only if the p-value is less than 0.05. The significance level is chosen before the test is conducted.

An equivalent procedure is to reject the null hypothesis if the test statistic observed is, in absolute value terms, larger than the critical value of the test statistic. The critical value is determined by the significance level chosen and defines two disjoint sets of values which are called acceptance region and rejection region. The acceptance region contains all values of the test statistic for which the test does not reject while the rejection region contains all the values for which the test does reject.

The p-value is the probability that, in repeated sampling under the same conditions, meaning i.i.d. sampling, the same null hypothesis and the same sample size, a test statistic is observed that provides just as much evidence against the null hypothesis as the test statistic actually observed.

The actual probability that the test rejects the true null hypothesis is called the size of the test. In an ideal setting, the size does not exceed the significance level.

The probability that the test correctly rejects a false null hypothesis is called power.

Reconsider `pvalue` computed further above:

```
# check whether p-value < 0.05
pvalue < 0.05
```

```
## [1] FALSE
```

The condition is not fulfilled so we do not reject the null hypotheis (remember that the null hypothesis is true in this example).

When working with a t-statistic instead, it is equivalent to apply the following rule:

$$\text{Reject } H_0 \text{ if } |t^{act}| > 1.96$$

We reject the null hypothesis at the significance level of 5% if the computed t-statistic lies beyond the critical value of 1.96 in absolute value terms. 1.96 is the 0.05-quantile of the standard normal distribution.

```
# check the critical value
qnorm(p = 0.05)
```

```
## [1] -1.644854
```

```
# check whether the null is rejected using the t-statistic computed further above
abs(tstatistic) > 1.96
```

```
## [1] FALSE
```

As when using the p-value, we cannot reject the null hypothesis using the corresponding t-statistic. Key Concept 3.6 summarizes the procedure of performing a two-sided hypothesis about the population mean $E(Y)$.

Key Concept 3.6

Testing the Hypothesis $E(Y) = \mu_{Y,0}$ Against the Alternative $E(Y) \neq \mu_{Y,0}$

1. Estimate μ_Y using $\overline{Y}$ and compute the standard error of $\overline{Y}$, $SE(\overline{Y})$.
2. Compute the t-statistic.
3. Compute the p-value and reject the null hypothesis at the 5% level of significance if the p-value is smaller than 0.05 or equivalently, if

$$\left|t^{act}\right| > 1.96.$$

One-sided Alternatives

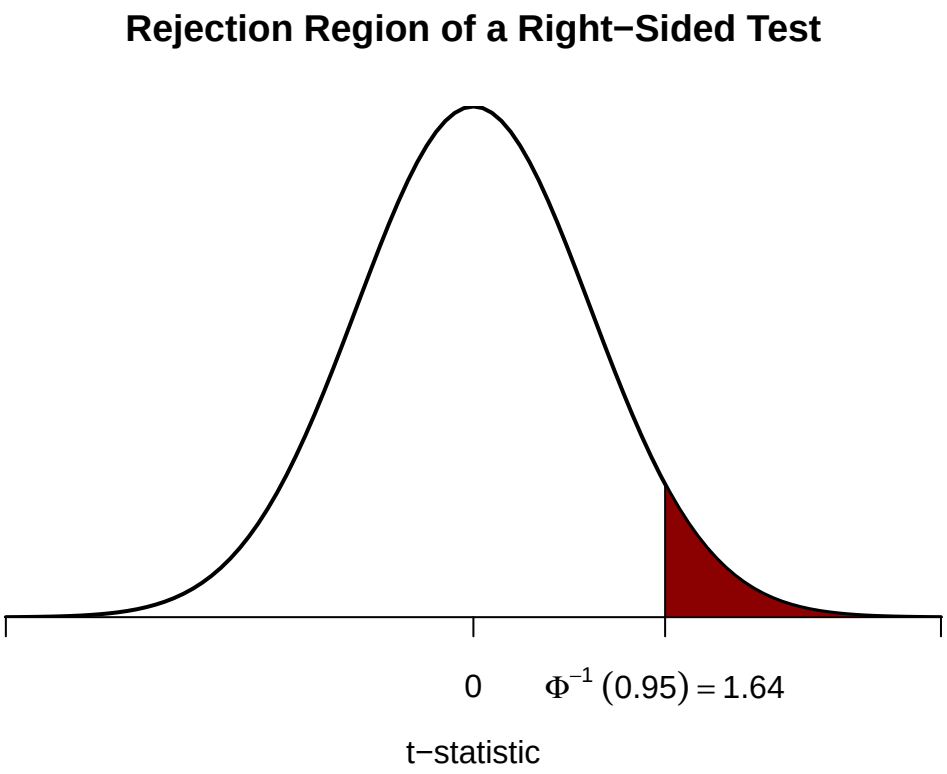
Sometimes we are interested in finding evidence that the mean is bigger or smaller than the some value hypothesized under the null. One can come up with many examples here but, to stick to the book, take the presumed wage differential between good and less educated working individuals. Since we hope that this differential exists, a relevant alternative (to the null hypothesis that there is no wage differential) is that good educated individuals earn more, i.e. that the average hourly wage for this group, μ_Y is *bigger* than $\mu_{Y,0}$ the know average wage of less educated workers.

This is an example of a *right-sided test* and the hypotheses pair is chosen as

$$H_0 : \mu_Y = \mu_{Y,0} \quad \text{vs} \quad H_1 : \mu_Y > \mu_{Y,0}.$$

We reject the null hypothesis if the computed test-statistic is larger than the critical value 1.64, the 0.95-quantile of the $N(0,1)$ distribution. This ensures that $1 - 0.95 = 5\%$ probability mass remains in the area to the right of the critical value. Similar as before we can visualize this in R using the function `polygon()`.

```
# plot the standard normal density on the domain [-4,4]
curve(dnorm(x),
      xlim = c(-4,4),
      main = 'Rejection Region of a Right-Sided Test',
      yaxs = 'i',
      xlab = 't-statistic',
      ylab = '',
      lwd = 2,
      axes = 'F'
)

# add x-axis
axis(1,
     at = c(-4,0,1.64,4),
     padj = 0.5,
     labels = c('',0,expression(Phi^-1~(.95)==1.64),'')
)

# shade rejection region in right tail
polygon(x = c(1.64, seq(1.64, 4, 0.01), 4),
        y = c(0, dnorm(seq(1.64, 4, 0.01)), 0),
        col = 'darkred'
)
```



In an analogously manner for the *left-sided test* we have

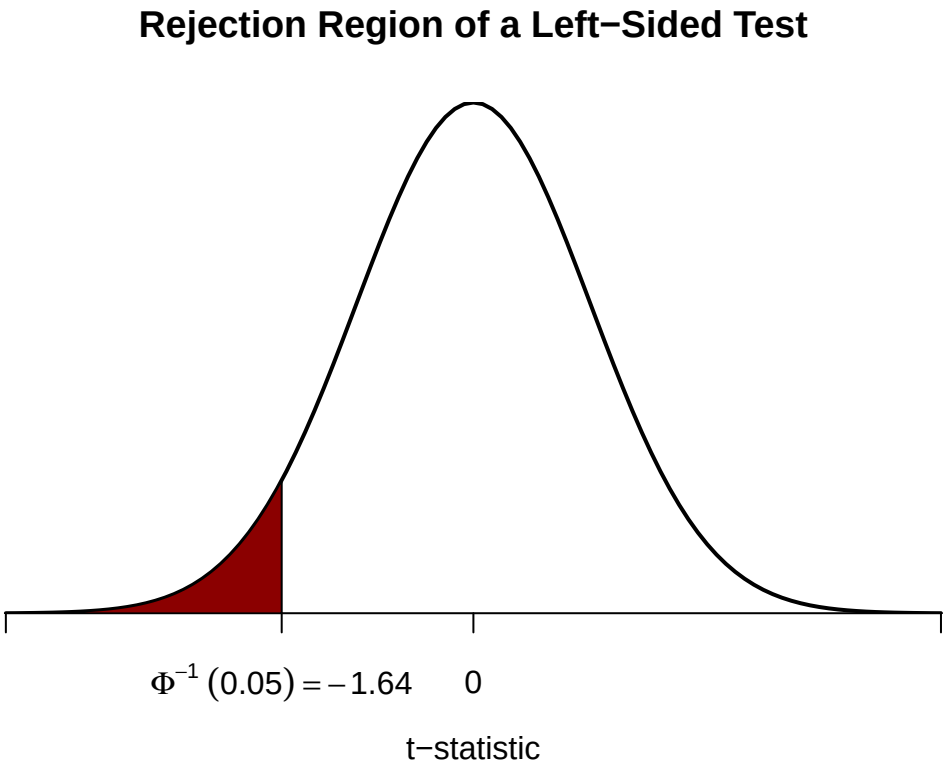
$$H_0 : \mu_Y = \mu_{Y,0} \quad \text{vs.} \quad H_1 : \mu_Y < \mu_{Y,0}.$$

The null is rejected if the observed test statistic falls short of the critical value which, for a test at the 0.05 level of significance, is given by -1.64, the 0.05-quantile of the $N(0,1)$ distribution. 5% probability mass lies to the left of the critical value.

It is straight forward to adapt the code chunk above to the case of a left-sided test. We only have to fiddle with the color shading and the tick marks.

```
# plot the standard normal density on the domain [-4,4]
curve(dnorm(x),
      xlim = c(-4,4),
      main = 'Rejection Region of a Left-Sided Test',
      yaxs = 'i',
      xlab = 't-statistic',
      ylab = '',
      lwd = 2,
      axes = 'F'
)

# add x-axis
axis(1,
     at = c(-4,0,-1.64,4),
     padj = 0.5,
     labels = c('',0,expression(Phi^-1~(.05)==-1.64),'')
)

# shade rejection region in right tail
```

```
polygon(x = c(-4, seq(-4, -1.64, 0.01), -1.64),
        y = c(0, dnorm(seq(-4, -1.64, 0.01)), 0),
        col = 'darkred'
)
```



4.4 Confidence intervals for the Population Mean

As stressed before, we will never estimate the exact value of a the population mean of Y using a random sample. However, we can compute confidence intervals for the population mean. In general, a confidence interval for a unkown parameter is a set of values that contains the true parameter with a prespecified probability, the confidence level. Confidence intervals are computed using the information available in the sample. Since this information is the result of a random process, confidence intervals are random variables themselves.

Key Concept 3.7 shows how to compute confidence intervals for the unknown population mean $E(Y)$.

Key Concept 3.6

Confidence Intervals for the Population Mean

A 95% confidence interval for μ_Y is a random variable that contains the true μ_Y in 95% of all possible random samples. When n is large we can use the normal approximation. Then, 99%, 95%, 90% confidence intervals are

$$99\% \text{ confidence interval for } \mu_Y = \left\{\overline{Y} \pm 2.58 \times SE(\overline{Y})\right\}. \tag{4.9}$$

$$95\% \text{ confidence interval for } \mu_Y = \left\{\overline{Y} \pm 1.96 \times SE(\overline{Y})\right\}. \tag{4.10}$$

$$90\% \text{ confidence interval for } \mu_Y = \left\{\overline{Y} \pm 1.64 \times SE(\overline{Y})\right\}. \tag{4.11}$$

These confidence intervals are sets of null hypotheses we cannot reject in a two-sided hypothesis test at the given level of confidence.

Now consider the following statements.

1. The interval
$$\left\{\overline{Y} \pm 1.96 \times SE(\overline{Y})\right\}$$
covers the true value of μ_Y with a probability of 95%.

2. We have computed $\overline{Y} = 5.1$ and $SE(\overline{Y} = 2.5$ so the interval
$$\{5.1 \pm 1.96 \times 2.5\} = [0.2, 10]$$
covers the true value of μ_Y with a probability of 95%.

While 1. is right (this is exactly in line with the definition above), 2. is completely wrong and none of Your lecturers wants to read such a sentence in a term paper, written exam or similar, believe us. The difference is that, while 1. is the definition of a random variable, 2. is one possible *outcome* of this random variable so there is no meaning in making any probabilistic statement about it. Either the computed interval *does cover* μ_Y or it *does not*!

In R, testing hypothesis about the mean of a population on the basis of a random sample is very easy due to functions like `t.test()` from the stats package. It procudes an object of type list. Luckily, one of the most simple ways to use `t.test()` is when You want to obtain a 95% confidence interval for some population mean. We start by generating some random data and calling `t.test()` in conjunction with `ls()` to obtain a breakdown of the output components.

```
# set random seed
set.seed(1)

# generate some sample data
sampledata <- rnorm(100,10,10)

# checke type
typeof(t.test(sampledata))
```

```
## [1] "list"
```

```
# display list elements produced by t.test
ls(
  t.test(sampledata)
)
```

```
## [1] "alternative" "conf.int"    "data.name"   "estimate"    "method"
## [6] "null.value"  "p.value"     "parameter"   "statistic"
```

Though we find that many items are reported, at the moment we are interested in computing a 95% confidence set for the mean.

```
t.test(sampledata)$"conf.int"
```

```
## [1]  9.306651 12.871096
## attr(,"conf.level")
## [1] 0.95
```

This tells us that the 95% confidence interval is

$$[9.31, 12.87]\,.$$

In this example, the computed interval does cover the true μ_Y which we know to be 10.

Let us have a look at the whole standard output produced by `t.test()`.

```
t.test(sampledata)
```

```
##
##  One Sample t-test
##
## data:  sampledata
## t = 12.346, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   9.306651 12.871096
## sample estimates:
## mean of x
##  11.08887
```

We see that `t.test()` not only computes a 95% confidence interval but automatically conducts a two-sided significance test of the hypothesis $H_0 : \mu_Y = 0$ at the level of 5% and reports relevant parameters thereof: the alternative hypothesis, the estimated mean, the resulting t-statistic, the degrees of freedom of the underlying t distribution (`t.test()` does not perform the normal approximation) and the corresponding p-value. Very convenient!

In this example, we come to the conclusion that the population mean *is not* significantly different from 0 at the level of 5% (which is correct), since $\mu_Y = 0$ is element of the 95% confidence interval

$$0 \in [-0.27, 0.12]\,.$$

We come to an equivalent result when using the p-value rejection rule:

$$p = 0.456 > 0.05$$

4.5 Comparing Means from Different Populations

Suppose You are interested in the means of two different populations, denote them μ_1 and μ_2. More specifically You are interested whether these population means are different from each other and plan an using a hypothesis test to verifiy this on the basis of independent sample data from both populations. A suitable pair of hypotheses then is

$$H_0 : \mu_1 - \mu_2 = d_0 \quad \text{vs.} \quad H_1 : \mu_1 - \mu_2 \neq d_0 \tag{4.12}$$

where d_0 denotes the hypothesized difference in means. The book teaches us that H_0 can be tested with the t-statistic

$$t = \frac{(\overline{Y}_1 - \overline{Y}_2) - d_0}{SE(\overline{Y}_1 - \overline{Y}_2)} \tag{4.13}$$

where

$$SE(\overline{Y}_1 - \overline{Y}_2) = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}. \tag{4.14}$$

This is called a two sample t-test. For large n_1 and n_2, (4.13) is standard normal distributed under the null hypothesis. Anlog to the simple t-test we can compute confidence intervals for the true difference in population means:

$$(\overline{Y}_1 - \overline{Y}_2) \pm 1.96 \times SE(\overline{Y}_1 - \overline{Y}_2)$$

is a 95% confidence interval for d. In R, Hypotheses as in (4.12) can be tested with `t.test()`, too. Note that `t.test()` chooses $d_0 = 0$ by default. This can be changed by setting the argument `mu` accordingly.

```
# set random seed
set.seed(1)

# draw data from two different populations with equal mean
sample_pop1 <- rnorm(100, 10, 10)
sample_pop2 <- rnorm(100, 10, 20)

# perform a two sample t-test
t.test(sample_pop1, sample_pop2)
```

```
##
##  Welch Two Sample t-test
##
## data:  sample_pop1 and sample_pop2
## t = 0.872, df = 140.52, p-value = 0.3847
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.338012  6.028083
## sample estimates:
## mean of x mean of y
## 11.088874  9.243838
```

We find that the two sample t-test does not reject the (true) null hypothesis that $d_0 = 0$.

4.6 An Application to the Gender Gap of Earnings

In this section discusses how to reproduce the results presented in the box '*The Gender Gap of Earnings of College Graduates in the United States*' in the book.

In order to reproduce table 3.1 You need to download the replication data which is hosted by Pearson and can be found and downloaded here. Download the data for chapter three as an excel spreadsheet (cps_ch3.xlsx). This data set contains data that ranges from 1992 to 2008 and earnings are reported in prices of 2008. There are several ways to import the .xlsx-files into R. Our suggestion is the function `read_excel()` from the readxl package. The package is not part of R's standard distribution and has to be installed manually.

```
# install and load the readxl package
## install.packages('readxl')
library(readxl)
```

You are now ready to import the data set. Make sure You use the correct path to the downloaded file! In our example, the file is saved in a subfolder (data) of the working directory. If You are not sure what Your current working directory is, use `getwd()`, see also `?getwd()`. This will give You the path that points to the place R is currently looking for files.

```
# import the data into R
cps <- read_excel(path = 'data/cps_ch3.xlsx')
```

Next, install and load the package dyplr. This package provides some handy functions that simplify data wrangling a lot. It makes use of the `%>%` operator.

```
# install and load the dplyr package
## install.packages('dplyr')
library('dplyr')
```

First, get an overview over the data set. Next, use `%>%` and some functions from the `dplyr` package to group the observations by gender and year and compute descriptive statistics for both groups.

```
# Get an overview of the data structure
head(cps)
```

```
## # A tibble: 6 x 3
##   a_sex  year ahe08
##   <dbl> <dbl> <dbl>
## 1    1. 1992.  17.2
## 2    1. 1992.  15.3
## 3    1. 1992.  22.9
## 4    2. 1992.  13.3
## 5    1. 1992.  22.1
## 6    2. 1992.  12.2
```

```
# group data by gender and year and compute the mean, standard deviation
# and number of observations for each group
avgs <- cps %>%
        group_by(a_sex, year) %>%
        summarise(mean(ahe08),
                  sd(ahe08),
                  n()
                  )

# print results to the console
print(avgs)
```

```
## # A tibble: 10 x 5
## # Groups:   a_sex [?]
##    a_sex  year `mean(ahe08)` `sd(ahe08)` `n()`
##    <dbl> <dbl>         <dbl>       <dbl> <int>
##  1    1. 1992.          23.3       10.2   1594
##  2    1. 1996.          22.5       10.1   1379
##  3    1. 2000.          24.9       11.6   1303
##  4    1. 2004.          25.1       12.0   1894
##  5    1. 2008.          25.0       11.8   1838
##  6    2. 1992.          20.0        7.87  1368
##  7    2. 1996.          19.0        7.95  1230
##  8    2. 2000.          20.7        9.36  1181
##  9    2. 2004.          21.0        9.36  1735
## 10    2. 2008.          20.9        9.66  1871
```

With the pipe operator `%>%` we simply chain different R functions that produce compatible input and ouput. In the code above, we take the dataset `cps` and use it as an input for the function `group_by()`. The output of `group_by` is subsequently used as an input for `summarise()` and so forth.

Now that we have computed the statistics of interest for both genders, we can investigate how the gap in earnings between both groups evolves over time.

```
# split the data set by gender
male <- avgs %>% filter(a_sex == 1)
female <- avgs %>% filter(a_sex == 2)

# Rename columns of both splits
colnames(male)   <- c("Sex", "Year", "Y_bar_m", "s_m", "n_m")
colnames(female) <- c("Sex", "Year", "Y_bar_f", "s_f", "n_f")

# Estimate Gender gaps, compute standard errors and confidence intervals for all dates
gap <- male$Y_bar_m - female$Y_bar_f

gap_se <- sqrt(male$s_m^2 / male$n_m + female$s_f^2 / female$n_f)

gap_ci_l <- gap - 1.96 * gap_se

gap_ci_u <- gap + 1.96 * gap_se

result <- cbind(male[,-1], female[,-(1:2)], gap, gap_se, gap_ci_l, gap_ci_u)

# print results to the console
print(result, digits = 3)
```

```
##   Year Y_bar_m  s_m  n_m Y_bar_f  s_f  n_f  gap gap_se gap_ci_l gap_ci_u
## 1 1992    23.3 10.2 1594    20.0 7.87 1368 3.23  0.332     2.58     3.88
## 2 1996    22.5 10.1 1379    19.0 7.95 1230 3.49  0.354     2.80     4.19
## 3 2000    24.9 11.6 1303    20.7 9.36 1181 4.14  0.421     3.32     4.97
## 4 2004    25.1 12.0 1894    21.0 9.36 1735 4.10  0.356     3.40     4.80
## 5 2008    25.0 11.8 1838    20.9 9.66 1871 4.10  0.354     3.41     4.80
```

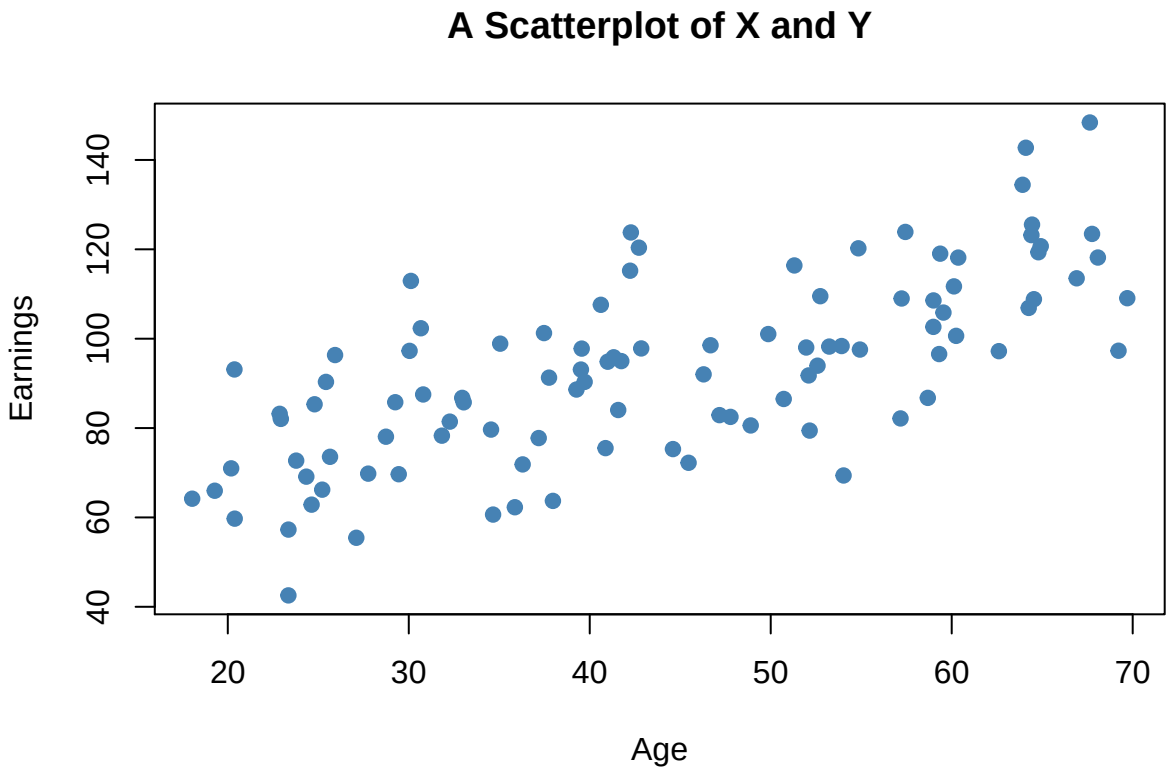
We observe virtually the same results as the ones presented in the book. the computed statistics suggest that there *is* a gender gap in earnings. Note that we can reject the null hypothesis that the gap is zero for all periodes. Further, estimates of the gap and bounds of the 95% confidence intervals indicate that the gap has been quite stable over the recent past.

4.7 Scatterplots, Sample Covariance and Sample Correlation

A scatterplot represents two dimensional data, for example n observation on X_i and Y_i, by points in a cartesian coordinate system. It is very easy to generate scatterplots using the `plot()` function in R. Let's generate some fictional data on age and earnings of workers and plot it.

```
# set random seed
set.seed(123)

# generate data set
X <- runif(n = 100,
           min = 18,
           max = 70
           )
Y <- X + rnorm(n=100, 50, 15)

# plot observations
plot(X,
     Y,
```

```
    type = "p",
    main = "A Scatterplot of X and Y",
    xlab = "Age",
    ylab = "Earnings",
    col = "steelblue",
    pch = 19
    )
```



The plot shows positive correlation between age and earnings. This in line with the assumption that older workers earn more than those that the joined the working population recently.

Sample Covariance and Correlation

By now You should be familiar with the concepts of variance and covariance. If not, we recommend You to work Your way through chapter 2 of the book (again).

As for the variance, covariance and correlation of two variables are properties that relate to the (unknown) joint probability distribution of these variable. Just as the individual population variances of both variables, we can estimate covariance and correlation by means of suitable estimators using a random sample (X_i, Y_i), $i = 1, \ldots, n$.

The sample covariance

$$s_{XY} = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})$$

is an estimator for the population variance of X and Y whereas the sample correlation

$$r_{XY} = \frac{s_{XY}}{s_X s_Y}$$

can be used to estimate the population correlation, a standardized measure for the strength of the linear relationship between X and Y. See chapter 3.7 in the book for a more detailed treatment of these estimators.

As for variance and standard deviation, these estimators are implemented as R functions in the stats package. We can use them to estimate population covariance and population correlation the fictional data on age and earnings.

```
# compute sample covariance of X and Y
cov(X,Y)
```

```
## [1] 213.934
```

```
# compute sample correlation between X and Y
cor(X,Y)
```

```
## [1] 0.706372
```

```
# equivalent way to compute the sample correlation
cov(X,Y)/(sd(X) * sd(Y))
```

```
## [1] 0.706372
```

The estimates indicate that X and Y are moderately correlated.

The next code chunk uses the function `mvnorm()` from package MASS to generate bivariate example data with different degree of correlation.

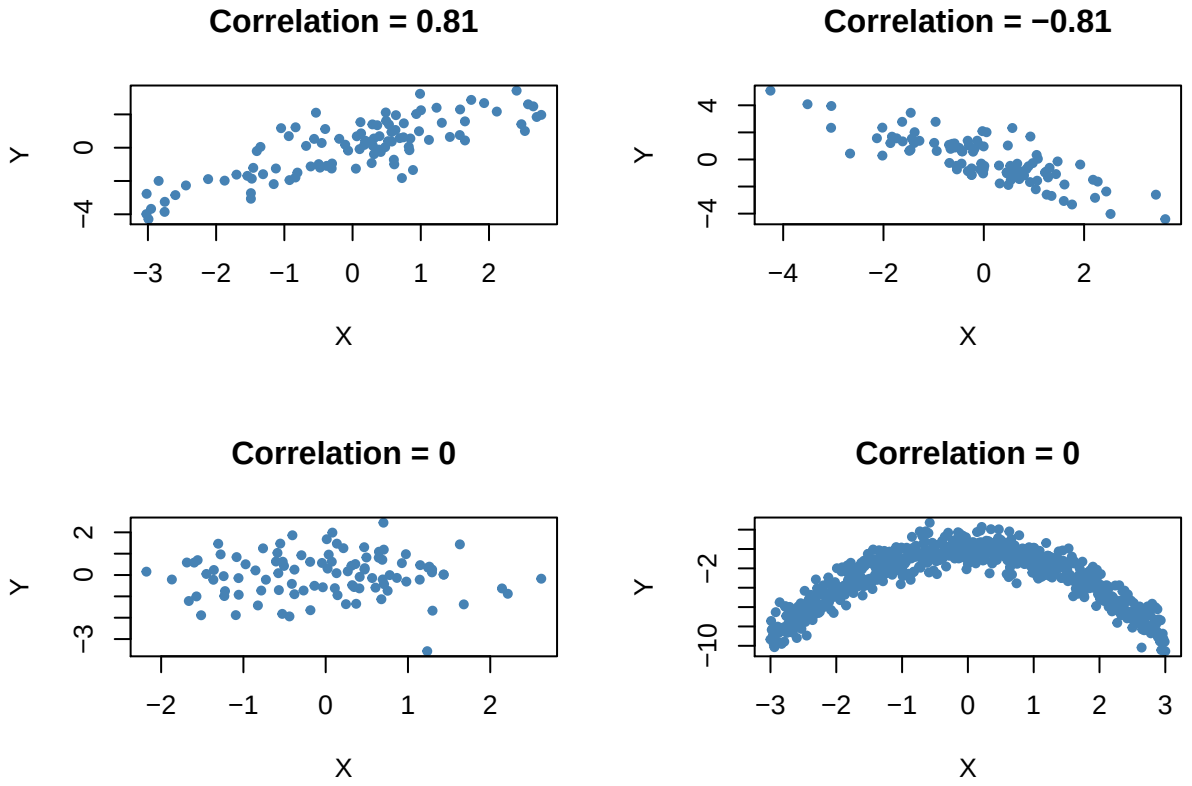
```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
# set random seed
set.seed(1)

# positive correlation (0.81)
example1 <- mvrnorm(100,
                    mu = c(0,0),
                    Sigma = matrix(c(2,2,2,3), ncol = 2),
                    empirical = TRUE
                    )

# negative correlation (-0.81)
example2 <- mvrnorm(100,
                    mu = c(0,0),
                    Sigma = matrix(c(2,-2,-2,3), ncol = 2),
                    empirical = TRUE
                    )

# no correlation
example3 <- mvrnorm(100,
                    mu = c(0,0),
                    Sigma = matrix(c(1,0,0,1), ncol = 2),
                    empirical = TRUE
                    )
```

```r
# no correlation (quadratic relationship)
X <- seq(-3,3,0.01)
Y <- -X^2 + rnorm(length(X))

example4 <- cbind(X,Y)

# estimate

## Plots

# divide plot area as 2-by-2 array
par(mfrow=c(2,2))

plot(example1, col='steelblue', pch=20, xlab = 'X', ylab = 'Y', main="Correlation = 0.81")
plot(example2, col='steelblue', pch=20, xlab = 'X', ylab = 'Y', main="Correlation = -0.81")
plot(example3, col='steelblue', pch=20, xlab = 'X', ylab = 'Y', main="Correlation = 0")
plot(example4, col='steelblue', pch=20, xlab = 'X', ylab = 'Y', main="Correlation = 0")
```



4.8 Exercises

1. Biased ...

Consider the following alternative estimator

$$\widetilde{Y} = \frac{1}{n-1}\sum_{i=1}^{n} Y_i$$

In this exercise we want to illustrate that this estimator is biased.

Instructions:

- Define a function for the estimator by yourself and name it Y_tilde.
- Randomly draw 5 observations from the $\mathcal{N}(10, 25)$ distribution and compute an estimate with Y_tilde(). Repeat this procedure 10000 times and store the results in est_biased.
- Plot a histogram of est_biased.
- Add a red vertical line at $\mu = 10$ using the function abline().

Define a function for the estimator

Compute repeatedly estimates and store the results in est_biased set.seed(123)

Plot a histogram of est_biased

Add a red vertical line at mu = 10

Define a function for the estimator Y_tilde <- function(x){sum(x)/(length(x)-1)}

Compute repeatedly estimates and store the results in est_biased set.seed(123) est_biased <- replicate(n = 10000, expr = Y_tilde(rnorm(5, 10, 5)))

Plot a histogram of est_biased hist(est_biased)

Add a red vertical line at mu = 10 abline(v = 10, col = "red")

test_function_definition('Y_tilde', function_test = { test_expression_result("Y_tilde(1:10)") test_expression_result("Y_t test_expression_result("Y_tilde(seq(0, 1, 0.1))") }) test_object('est_biased') test_function('hist', args = 'x') test_function('abline', args = c('v', 'col'))

Hints:

- To compute the sum of a vector you can use sum(), to get the length of a vector you can use length().
- Use the function replicate() to compute repeatedly estimates of random samples. With the arguments expr and n you can specify the operation and how often it has to be replicated.
- A histogram can be plotted with the function hist().
- The point on the x-axis as well as the color for the vertical line can be specified via the arguments v and col.

2. ... but consistent estimator

Consider again the estimator from the previous exercise which is available in your environment as Y_tilde(). Do the same procedure as in the previous exercise but now increase the number of observations to draw from 5 to 1000. What do you note? What can you say about this estimator?

Instructions:

- Randomly draw 1000 observations from the $\mathcal{N}(10, 25)$ distribution and compute an estimate with Y_tilde(). Repeat this procedure 10000 times and store the results in est_consistent.
- Plot a histogram of est_consistent.
- Add a red vertical line at $\mu = 10$ using the function abline().

Y_tilde <- function(x){sum(x)/(length(x)-1)}

Compute repeatedly estimates and store the results in est_consistent set.seed(123)

Plot a histogram of est_biased

Add a red vertical line at mu = 10

Compute repeatedly estimates and store the results in est_consistent set.seed(123) est_consistent <- replicate(n = 10000, expr = Y_tilde(rnorm(1000, 10, 5)))

Plot a histogram of est_consistent hist(est_consistent)

Add a red vertical line at mu = 10 abline(v = 10, col = "red")

test_object('est_consistent') test_function('hist', args = 'x') test_function('abline', args = c('v', 'col'))

Hints:

- Use the function replicate() to compute repeatedly estimates of random samples. With the arguments expr and n you can specify the operation and how often it has to be replicated.
- A histogram can be plotted with the function hist().
- The point on the x-axis as well as the color for the vertical line can be specified via the arguments v and col.

3. Efficiency of an estimator

In this exercise we want to illustrate the result that the sample mean

$$\hat{\mu}_Y = \sum_{i=1}^{n} a_i Y_i$$

with equal weighting scheme $a_i = \frac{1}{n}$ for $i = 1, ..., n$ is the best linear unbiased estimator (BLUE) of μ_Y.

As an alternative consider the estimator

$$\tilde{\mu}_Y = \sum_{i=1}^{n} b_i Y_i$$

where b_i gives the first $\frac{n}{2}$ observations a higher weighting than the second $\frac{n}{2}$ observations.

The respective weighting vector is already defined and available as w in your working environment.

Instructions:

- Verify that $\tilde{\mu}$ is unbiased.
- Define the alternative estimator as a function mu_tilde().
- Randomly draw 100 observations from the $\mathcal{N}(5, 10)$ distribution and compute an estimate with both estimators. Repeat this procedure 10000 times and store the results in est_bar and est_tilde.
- Compute the sample variances of est_bar and est_tilde. What can you say about both estimators?

Verify that the alternative estimator is unbiased n <- 100 w <- c(rep((1+0.5)/n, n/2), rep((1-0.5)/n, n/2))

Define the alternative estimator mu_tilde

Compute repeatedly estimates for both estimators and store the results in est_bar and est_tilde set.seed(123)

Compute the sample variances for est_bar and est_tilde

Verify that the alternative estimator is unbiased n <- 100 w <- c(rep((1+0.5)/n, n/2), rep((1-0.5)/n, n/2)) sum(w)

Define the alternative estimator mu_tilde mu_tilde <- function(x){sum(w*x)}

Compute repeatedly estimates for both estimators and store the results in est_bar and est_tilde set.seed(123) est_bar <- replicate(expr = mean(rnorm(100, 5, 10)), n = 10000) est_tilde <- replicate(expr = mu_tilde(rnorm(100, 5, 10)), n = 10000)

Compute the sample variances for est_bar and est_tilde var(est_bar) var(est_tilde)

test_function_result('sum') test_function_definition('mu_tilde', function_test = { test_expression_result("mu_tilde(1:100 test_expression_result("mu_tilde(2:101)") }) test_object('est_bar') test_object('est_tilde') test_function_result('var', index = 1) test_function_result('var', index = 2) success_msg('Correct! The sample mean is more efficient (that is, has a lower variance) than the alternative estimator.')

Hints:

- In order to be an unbiased estimator all weights have to sum up to 1.
- Use the function replicate() to compute repeatedly estimates of random samples. With the arguments expr and n you can specify the operation and how often it has to be replicated.
- To compute sample variances you can use var().

4. Hypothesis Test — t Statistic

Consider the CPS data set from Chapter 3.6 again which is available as cps in your working environment.

We suppose that the average hourly earnings (in prices of 2012) ahe12 exceed 23.50 $\frac{\$}{h}$ and wish to test this hypothesis at a significance level of $\alpha = 0.05$. For that reason please do the following:

Instructions:

- Compute the test statistic by hand and assign it to tstat.
- Use tstat to accept or reject the null hypothesis. Please do so using the normal approximation.

cps <- read.table("http://s3.amazonaws.com/assets.datacamp.com/production/course_1276/datasets/cps_ch3.csv", header = T, sep = ";")

Compute the t statistic by hand and assign it to tstat

Use tstat to accept or reject the null

Compute the t statistic by hand and assign it to tstat tstat <- (mean(cps$ahe12)−23.5)/(sd(cps$ahe12)/sqrt(length(cps$ahe1

Use tstat to accept or reject the null tstat > qnorm(0.95)

test_object('tstat') test_function_result('qnorm') test_student_typed('tstat', times = 2) test_or(test_output_contains('T' test_output_contains('F'))

Hints:

- We test $H_0 : \mu_{Y_{ahe}} \leq 23.5$ vs. $H_1 : \mu_{Y_{ahe}} > 23.5$. That is, we conduct a right-sided test.
- The t statistic is defined as $\frac{\bar{Y} - \mu_{Y,0}}{s_Y/\sqrt{n}}$ where s_Y denotes the sample variance.
- To decide whether the null hypothesis is accepted or rejected you can compare the t statistic with the respective quantile of the standard normal distribution. Use logical operators to check for this.

5. Hypothesis Test — p-value

Reconsider the test situation from previous exercise. cps as well as tstat are available in your working environment.

Instead of using the t statistic as decision criterion we can also use the respective p-value. For that reason please do the following:

Instructions:

- Compute the p-value by hand and assign it to pval.
- Use pval to accept or reject the null hypothesis.

cps <- read.table("http://s3.amazonaws.com/assets.datacamp.com/production/course_1276/datasets/cps_ch3.csv", header = T, sep = ";") tstat <- (mean(cps$ahe12) − 23.5)/(sd(cps$ahe12)/sqrt(length(cps$ahe12)))

Compute the p-value by hand and assign it to pval

Use pval to accept or reject the null

Compute the p-value by hand and assign it to pval pval <- 1-pnorm(tstat)

Use pval to accept or reject the null pval < 0.05

test_object('pval') test_student_typed('pval', times = 2) test_or(test_output_contains('T'), test_output_contains('F'))

Hints:

- The p-value for a right-sided test can be computed as $p = P(t > t^{act}|H_0)$.
- We reject the null if $p < \alpha$. Use logical operators to check for this.

6. Hypothesis Test — One Sample t-test

In the last two exercises we discovered two ways of conducting a hypothesis test. In practice these approaches seem very cumbersome and so R provides the function t.test() which does most of the work automatically for us. In fact it provides t statistics, p-values and even confidence intervals (more on the latter in later exercises). In addition it also uses the t instead of the normal distribution which becomes important especially for small sample sizes.

The data set cps as well as the variable pval from Exercise 3.4 are available in your working environment.

Instructions:

- Conduct the hypothesis test from previous exercises with the function t.test().
- Extract the t statistic and p-value from the list created by t.test(). Assign them to the variables tstat and pvalue.
- Verify that using the normal approximation here is valid as well by computing the difference between both p-values.

cps <- read.table("http://s3.amazonaws.com/assets.datacamp.com/production/course_1276/datasets/cps_ch3.csv", header = T, sep = ";") tstat <- (mean(cps$ahe12) − 23.5)/(sd(cps$ahe12)/sqrt(length(cps$ahe12))) pval <- 1-pnorm(tstat)

Conduct the hypothesis test from previous exercises with t.test()

Extract t statistic and p-value from list created by t.test()

Verify that using the normal approximation here is valid as well

Conduct the hypothesis test from previous exercises with t.test() t.test(cps$ahe12, alternative = "greater", mu = 23.5)

Extract t statistic and p-value from list created by t.test() tstat <- t.test(cps$ahe12, alternative = greater, mu = 23.5)statistic pvalue <- t.test(cps$ahe12, alternative = greater, mu = 23.5)p.value

Verify that using the normal approximation here is valid as well pvalue - pval

test_function_result('t.test') test_object('tstat') test_object('pvalue') test_or(test_student_typed('pvalue - pval'), test_student_typed('pval - pvalue'))

Hints:

- The type of the test as well as the null hypothesis can be specified via the arguments alternative and mu.
- t statistic and p-value can be obtained via $statistic < /tt > and < tt >$p.value, respectively.

7. Hypothesis Test — Two Sample t-test

Consider the annual maximum sea levels at Port Pirie (Southern Australia) and Fremantle (Western Australia) for the last 30 years.

The observations are available in the variables portpirie and fremantle in your working environment.

Instructions:

- Test whether there is a significant difference in the annual maximum sea levels at a significance level of $\alpha = 0.05$.

set.seed(123) portpirie <- runif(30, 3.6, 4.6) fremantle <- runif(30, 1.2, 1.8)

Conduct a two sample t-test

Conduct a two sample t-test t.test(portpirie, fremantle)

test_or(test_output_contains('t.test(portpirie, fremantle)'), test_output_contains('t.test(fremantle, portpirie)'))

Hints:

- We test $H_0 : \mu_P - \mu_F = 0$ vs. $H_1 : \mu_P - \mu_F \neq 0$. That is, we conduct a two sample t-test.
- For a two sample t-test t.test() expects two vectors containing the data.

8. Confidence Interval

Reconsider the test situation concerning the annual maximum sea levels at Port Pirie and Fremantle.

The variables portpirie and fremantle are again available in your working environment.

Instructions:

- Construct a 95%-confidence interval for the difference in the sea levels using t.test().

set.seed(123) portpirie <- runif(30, 3.6, 4.6) fremantle <- runif(30, 1.2, 1.8)

Construct a 95%-confidence interval using t.test()

Construct a 95%-confidence interval using t.test() t.test(portpirie, fremantle)$conf.int

test_or(test_output_contains('t.test(portpirie, fremantle)$conf.int'), $test_output_contains('t.test(fremantle, portpirie)$conf.in

Hint:

- The function t.test() computes by default a confidence interval which is accessible via $conf.int.

4.8.0.1 9. (Co)variance and Correlation I

Consider a random sample (X_i, Y_i) for $i = 1, ..., 100$.

The respective vectors X and Y are already available in your working environment as X and Y.

Instructions:

- Compute the variance of X using the function cov().
- Compute the covariance of X and Y.
- Compute the correlation between X and Y.

custom_seed(123) X <- runif(100, 900, 1000) Y <- 3*X+rnorm(100, 0, 100)

Compute the variance of X

Compute the covariance of X and Y

Compute the correlation between X and Y

Compute the variance of X with cov() cov(X, X)

Compute the covariance of X and Y cov(X, Y)

Compute the correlation between X and Y cor(X, Y)

test_function_result("cov", index = 1) test_function_result("cov", index = 2) test_function_result("cor")

Hints:

- The variance is just a special case of the covariance.
- cov() as well as cor() expect a vector for each variable.

4.8.0.2 10. (Co)variance and Correlation II

In this exercise we want to examine the limitations of the correlation as a dependency measure.

Once the session has initialized you will see the plot of 100 realizations from two random variables X and Y.

The respective observations are available in the vectors X and Y in your working environment.

Instructions:

- Compute the correlation between X and Y. Interpret your result critically.

custom_seed(123) X <- runif(100, 0, 3) Y <- exp(-X) + rnorm(100, 0, .05) plot(Y ~ X)

Compute the correlation between X and Y

Compute the correlation between x and y cor(X, Y) # The correlation is only able to quantify linear relationships which is not the case here. test_function_result("cor") success_msg('Correct! The correlation is able to capture the negative relationship between both variables. However it only measures linear dependencies, whereas the dependency here is clearly nonlinear (exponential).')

Hint:

- cor() expects a vector for each variable.

Chapter 5

Linear Regression with One Regressor

This chapter introduces the basics in linear regression and shows how to perform regression analysis in R. In linear regression, the aim is to model the relationship between a dependent variable Y and one or more explanatory variables denoted as $X_1, X_2, \ldots, X_k$. Following the book we will focus on the concept of simple linear regression throughout the whole chapter. In simple linear regression, there is just one explanatory variable X_1. If for example a school cuts the class sizes by hiring new teachers, that is the school lowers the student-teacher ratios of their classes, X_1, how would this affect the performance of the students involved in a standardized test,Y? With linear regression we can not only examine whether the student-teacher ratio *does have* an impact on the test results but we can also learn about the *direction* and the *strength* of this effect.

To start with an easy example, consider the following combinations of average test score and the average student-teacher ratio in some fictional school disctrics.

1

2

3

4

5

6

7

TestScore

680

640

670

660

630

660.0

635

STR

15

17

19

20

22

23.5

25

To work with these data in R we begin by creating two vectors: one for the student-teacher ratios (`STR`) and one for test scores (`TestScore`), both containing the data from the table above.

```
# Create sample data
STR <- c(15, 17, 19, 20, 22, 23.5, 25)
TestScore <- c(680, 640, 670, 660, 630, 660, 635)

# Print out sample data
STR
```

```
## [1] 15.0 17.0 19.0 20.0 22.0 23.5 25.0
```

```
TestScore
```

```
## [1] 680 640 670 660 630 660 635
```

If we use a simple linear regression model, we assume that the true relationship between both variables can be represented by a straight line, formally

$$Y = b \cdot X + a.$$

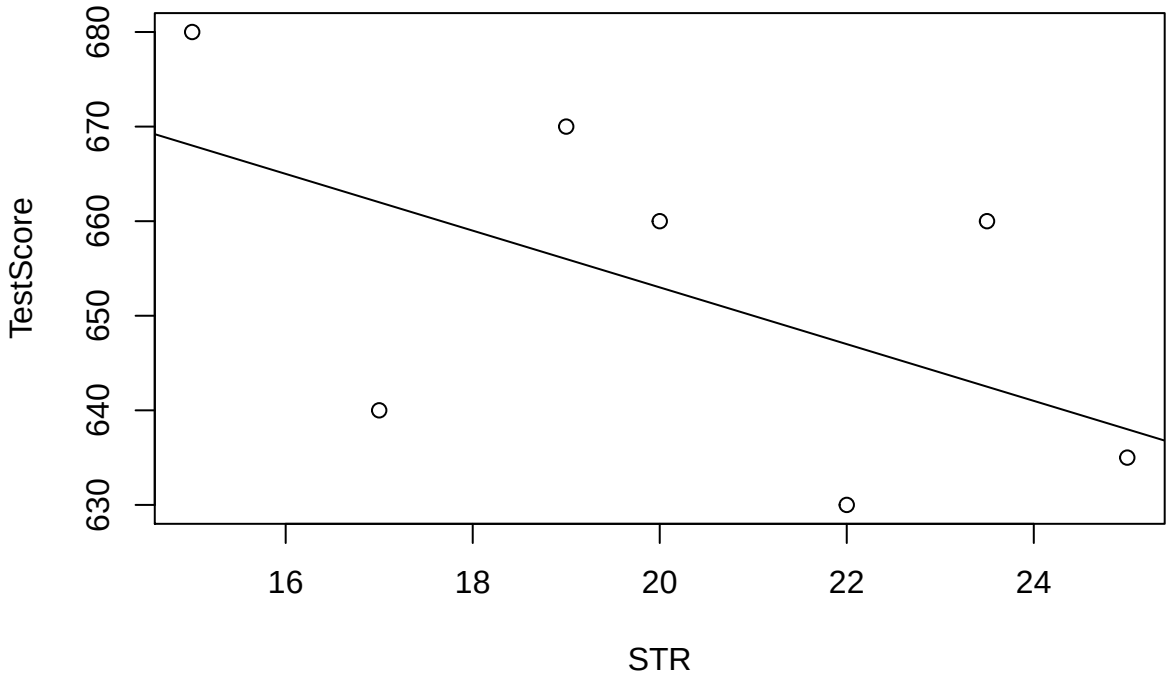
For now, let us suppose that the true function which relates test score and student-teacher ratio to each other is

$$TestScore = 713 - 3 \times STR.$$

If possible, it is always a good idea to visualize the data You work with in an appropriate way. For our purpose it is suitable to use the function `plot()` to produce a scatterplot with `STR` on the X-axis and `TestScore` on the Y axis. An easy way to do so is to call `plot(y_variable ~ x_variable)` whereby `y_variable` and `x_variable` are placeholders for the vectors of observations we want to plot. Furthermore, we might want to add the true relationship to the plot. To draw a straight line, R provides the function `abline()`. We just have to call this function with arguments `a` (representing the intercept) and `b` (representing the slope) after executing `plot()` in order to add the line to our scatterplot.

The following code reproduces figure 4.1 from the textbook.

```
# create a scatter plot of the data
plot(TestScore ~ STR)
# add the true relationship to the plot
abline(a = 713, b = -3)
```



We find that our line does not touch any of the points although we claimed that it represents the true relationship. The reason for this is the core problem of statistics, *randomness*. Most of the time there are influences which cannot be explained in a purely deterministic fashion and thus exacerbate finding the true relationship.

In order to account for these differences between observed data and the true relationship, we extend our model from above by an *error term* u which covers these random effects. Put differently, u accounts for all the differences between the true regression line and the actual observed data. Beside pure randomness, these deviations could also arise from measerment errors or, as will be discussed later, could be the consequence of leaving out other factors that are relevant in explaining the dependent variable. Which other factor are plausible in our example? For one thing, the test scores might be driven by the teachers quality and the background of the students. It is also imaginable that in some classes, the students were lucky on the test days and thus achieved higher scores. For now, we will summarize such influences by an additive component:

$$TestScore = \beta_0 + \beta_1 \times STR + \text{other factors}$$

Of course this idea is very general as it can be easily extented to other situations that can be described with a linear model. The basic linear regression function we will work with hence is

$$Y_i = \beta_0 + \beta_1 X_i + u_i.$$

Key Concept 4.1 summarizes the terminology of the simple linear regression model.

Key Concept 4.1

Terminology for the Linear Regression Model with a Single Regressor

The linear regression model is

$$Y_i = \beta_0 + \beta_1 X_1 + u_i$$

where

- the subscript i runs over the observations, $i = 1$, ..., n

- Y_i is the *dependent variable*, the *regressand*, or simply the *left-hand variable*
- X_i is the *independent variable*, the *regressor*, or simply the *right-hand variable*
- $Y = \beta_0 + \beta_1 X$ is the *population regression line* also called the *population regression function*
- β_0 is the *intercept* of the population regression line
- β_1 is the *slope* of the population regression line
- u_i is the *error term*

5.1 Estimating the Coefficients of the Linear Regression Model

In practice, the intercept β_0 and slope $\beta_1$of the population regression line are unknown. Therefore, we must employ data to estimate both unknown parameters. In the following a real world example will be used to demonstrate how this is achieved. We want to relate test scores to student-teacher ratios measured in californian schools. The test score is the district-wide average of reading and math scores for fifth graders. Again, the class size is measured as the number of students divided by the number of teachers (the student-teacher ratio). As for the data, the California School dataset (`CASchools`) comes with a R package called AER, an acronym for Applied Econometrics with R. After installing the package with `install.packages("AER")` and attaching it with `library("AER")` the dataset can be loaded using the `data` function.

```
# install the AER package (once)
install.packages("AER")

# load the AER package
library(AER)

# load the the data set in the workspace
data(CASchools)
```

Note that once a package has been installed it is available for use at further occasions when invoked with `library()` — there is no need to run `install.packages("...")` again!

For several reasons it is interesting to know what kind of object we are dealing with. `class(object_name)` returns the type (class) of an object. Depending on the class of an object some functions (such as `plot()` and `summary()`) behave differently.

Let us check the class of the object `CASchools`.

```
class(CASchools)
```

```
## [1] "data.frame"
```

It turns out that `CASchools` is of class `data.frame` which is a convienient format to work with.

With help of the function `head()` we get a first overview of our data. This function shows only the first 6 rows of the data set which prevents an overcrowded console output.

> Press ctrl + L to clear the console. This command deletes any code that has been typed in and executed by You or printed to the console by R functions. Good news is: anything else is left untouched. You neither loose defined variables and alike nor the code history. It is still possible to recall previously executed R commands using the up and down keys. If You are working in RStudio, press ctrl + Up on Your keyboard (CMD + Up on a mac) to review a list of previously entered commands.

```
head(CASchools)
```

```
##   district                        school  county grades students
```

```
## 1    75119              Sunol Glen Unified Alameda  KK-08      195
## 2    61499            Manzanita Elementary   Butte  KK-08      240
## 3    61549     Thermalito Union Elementary   Butte  KK-08     1550
## 4    61457 Golden Feather Union Elementary   Butte  KK-08      243
## 5    61523        Palermo Union Elementary   Butte  KK-08     1335
## 6    62042         Burrel Union Elementary  Fresno  KK-08      137
##   teachers calworks   lunch computer expenditure    income   english  read
## 1    10.90   0.5102  2.0408       67    6384.911 22.690001  0.000000 691.6
## 2    11.15  15.4167 47.9167      101    5099.381  9.824000  4.583333 660.5
## 3    82.90  55.0323 76.3226      169    5501.955  8.978000 30.000002 636.3
## 4    14.00  36.4754 77.0492       85    7101.831  8.978000  0.000000 651.9
## 5    71.50  33.1086 78.4270      171    5235.988  9.080333 13.857677 641.8
## 6     6.40  12.3188 86.9565       25    5580.147 10.415000 12.408759 605.7
##    math
## 1 690.0
## 2 661.9
## 3 650.9
## 4 643.5
## 5 639.9
## 6 605.4
```

We find that the dataset consists of plenty of variables and most of them are numeric.

By the way: an alternative to `class()` and `head()` is `str()` which is deduced from 'structure' and gives a comprehensive overview of the object. Try this!

Turning back to `CASchools`, the two variables we are intersted in (i.e. average test score and the student-teacher ratio) are *not* included. However, it is possible to calculate both from the provided data. To obtain the student-teacher ratios, we simply divide the number of students by the number of teachers. The avarage test score is the arithmetic mean of the test score for reading and the score of the math test. The next code chunk shows how the two variables can be constructed and how they are appended to `CASchools` which is a data.frame.

```
# compute STR and append it to CASchools
CASchools$STR <- CASchools$students/CASchools$teachers

# compute TestScore and append it to CASchools
CASchools$score <- (CASchools$read + CASchools$math)/2
```

If we ran `head(CASchools)` again we would find the two variables of interest as additional columns named `STR` and `score` (check this!).

Table 4.1 from the text book summarizes the distribution of test scores and student-teacher ratios. There are several functions which can be used to produce similar results within R:

- `mean()` (computes the arithmetic mean of the provided numbers)
- `sd()` (computes the sample standard deviation)
- `quantile()` (returns a vector of the specified quantiles for the data)

The next code chunk shows how to achieve this. First, we compute summary statistics on the coloumns `STR` and `score` of `CASchools`. In order to have a nice display format we gather the computed measures in a `data.frame` object named `DistributionSummary`.

```
# compute sample averages of STR and score
avg_STR <- mean(CASchools$STR)
avg_score <- mean(CASchools$score)
```

```
# compute sample standard deviations of STR and score
sd_STR <- sd(CASchools$STR)
sd_score <- sd(CASchools$score)

# set up a vector of percentiles and compute the quantiles
quantiles <- c(0.10, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9)
quant_STR <- quantile(CASchools$STR, quantiles)
quant_score <- quantile(CASchools$score, quantiles)

# gather everything in a data.frame
DistributionSummary <- data.frame(
                                  Average = c(avg_STR, avg_score),
                                  StandardDeviation = c(sd_STR, sd_score),
                                  quantile = rbind(quant_STR, quant_score)
                                  )

# print the summary to the console
DistributionSummary
```

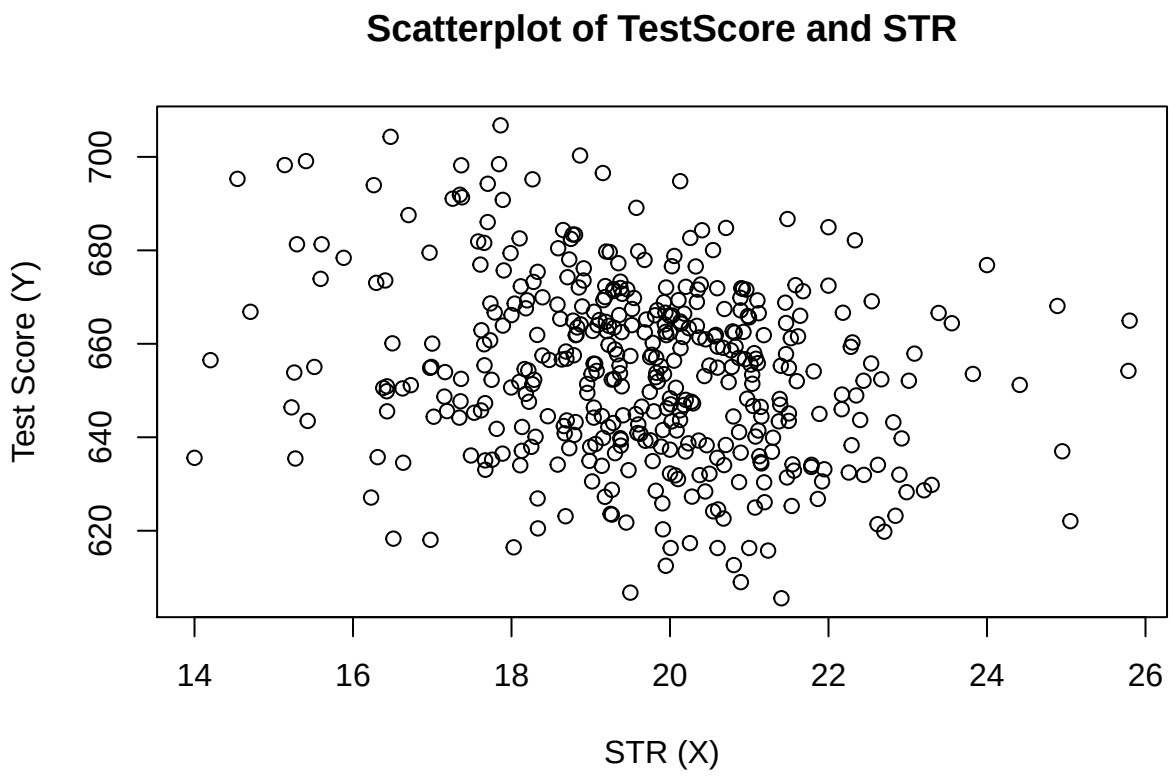
```
##               Average StandardDeviation quantile.10. quantile.25.
## quant_STR    19.64043          1.891812      17.3486     18.58236
## quant_score 654.15655         19.053347     630.3950    640.05000
##             quantile.40. quantile.50. quantile.60. quantile.75.
## quant_STR       19.26618     19.72321      20.0783     20.87181
## quant_score    649.06999    654.45000     659.4000    666.66249
##             quantile.90.
## quant_STR       21.86741
## quant_score    678.85999
```

The standard distribution (the Base R package) of R already contains a `summary` function which can be applied to objects of class `data.frame`. Type and execute `summary(STR)`!

As done for the sample data, we use `plot()` for a visual survey. This allows us to detect specific characteristics of our data, such as outliers which are hard to discover by looking at mere numbers. This time we add some additional arguments to the `plot()` function.

The first argument in our call of `plot()`, `score ~ STR`, is again a formula that states the dependent variable and the regressor. However, this time the two variables are not saved in seperate vectors but are columns of `CASchools`. Therefore, R would not find the variables without the argument `data` beeing correctly specified. `data` must be in accordance with the name of the `data.frame` to which the variables belong, in this case `CASchools`. Further arguments are used to change the appearance of the plot: while `main` adds a title, `xlab` and `ylab` are adding custom labels to both axes.

```
plot(score ~ STR,
     data = CASchools,
     main = "Scatterplot of TestScore and STR",
     xlab = "STR (X)",
     ylab = "Test Score (Y)"
)
```



The plot (figure 4.2 in the book) shows the scatterplot of all observations on student-teacher ratio and Test score. We see that the points are strongly scatterd and an apparent relationship cannot be detected by only looking at them. Yet it can be assumed that both variables are negatively correlated, that is we expect to observe lower test scores in bigger classes.

The function `cor()` (type and execute `?cor` for further info), can be used to compute the correlation between 2 *numerical* vectors.

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

```
## [1] -0.2263627
```

As the scatterplot already suggests, the correlation is negative but rather weak.

The task we are facing now is to find a line which fits best to the data. Of course we could simply stick with graphical inspection and correlation analysis and then select the best fitting line by eyeballing. However, this is pretty unscientific and prone to subjective perception: different students would draw different regression lines. On this account, we are interested in techniques that are more sophisticated. Such a technique is ordinary least squares (OLS) estimation.

The Ordinary Least Squares Estimator

The OLS estimator chooses the regression coefficients such that the estimated regression line is as close as possible to the observed data points. Thereby closeness is measured by the sum of the squared mistakes made in predicting Y given X. Let b_0 and b_1 be some estimators of β_0 and β_1. Then the sum of squared estimation mistakes can be expressed as

$$\sum_{i=1}^{n}(Y_i - b_0 - b_1 X_i)^2.$$

The OLS estimator in the simple regression model is the pair of estimators for intercept and slope which minimizes the expression above. The derivation of the OLS estimators for both parameters are presented in Appendix 4.1 of the book. The results are summarized in Key Concept 4.2.

Key Concept 4.2

The OLS Estimator, Predicted Values, and Residuals

The OLS estimators of the slope β_1 and the intercept β_0 in the simple linear regression model are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \tag{5.1}$$

$$\tag{5.2}$$

$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{X} \tag{5.3}$$

The OLS predicted values $\widehat{Y}_i$ and residuals $\hat{u}_i$ are

$$\widehat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i, \tag{5.4}$$

$$\tag{5.5}$$

$$\hat{u}_i = Y_i - \widehat{Y}_i. \tag{5.6}$$

The estimated intercept $\hat{\beta}_0$, the slope parameter $\hat{\beta}_1$, and the residuals ($\hat{u}_i$) are computed from a sample of n observations of X_i and Y_i, i, ..., n. These are *estimates* of the unkown true population intercept (β_0), slope (β_1), and error term (u_i).

We are aware that the results presented in Key Concept 4.2 are not very intuitive at first glance. The following interactice application aims to help You understand the mechanics of OLS. You can add observations by clicking into the coordinate system where the data are represented by points. If two or more observations are available, the application computes a regression line using OLS and some statistics which are displayed in the right panel. The results are updated as You add further observations to the left panel. A double-click resets the application i.e. all data are removed.

There are many possible ways to compute $\hat{\beta}_0$ and $\hat{\beta}_1$ in R. For example, we could implement the formulas presented in Key Concept 4.2 with two of R's most basic functions: `mean()` and `sum()`.

```
attach(CASchools) #allows to use the variables contained in CASchools directly

# compute beta_1
beta_1 <- sum((STR - mean(STR))*(score - mean(score))) / sum((STR - mean(STR))^2)

# compute beta_0
beta_0 <- mean(score) - beta_1 * mean(STR)

# print the results to the console
beta_1
```

```
## [1] -2.279808
```

```
beta_0
```

```
## [1] 698.9329
```

Of course there are also other and even more manual ways to do the same tasks. Luckily, OLS is one of the most widely-used estimation techniques. Being a statistical programming language, R already contains a built-in function named `lm()` (**l**inear **m**odel) which can be used to carry out regression analysis.

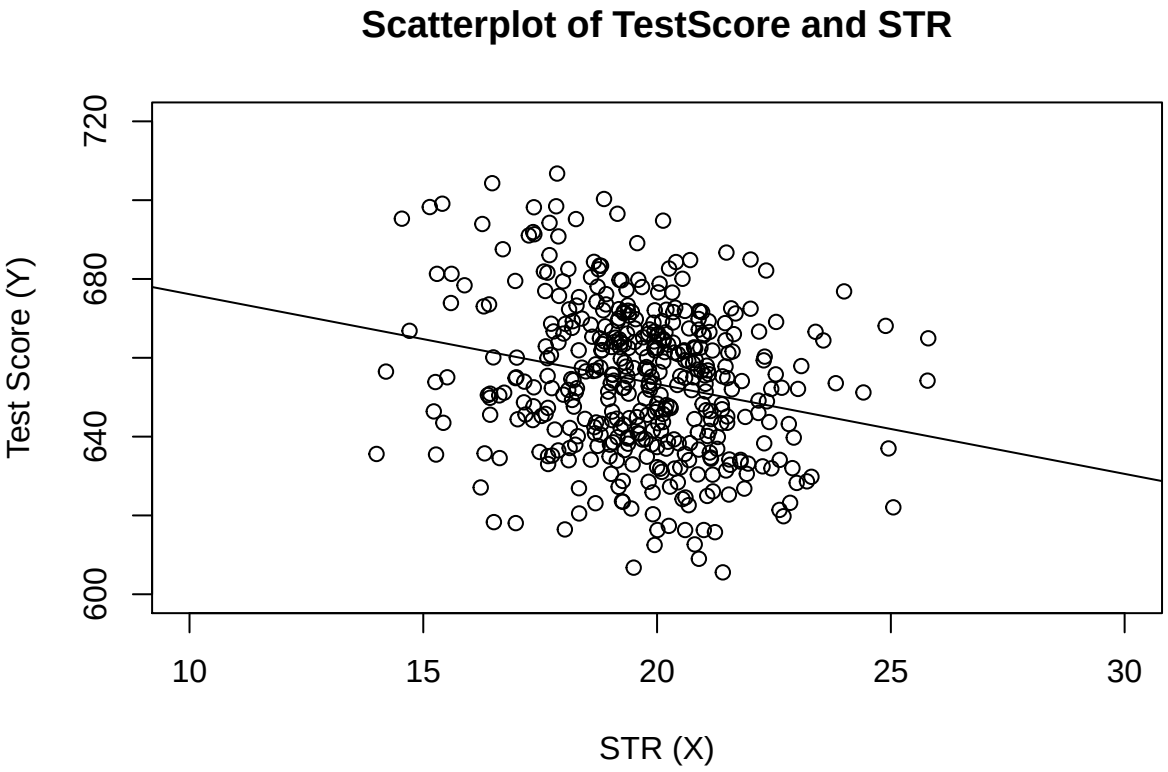
The first argument of the function to be specified is, similar as in `plot()`, the regression formula with the basic syntax `y ~ x` where `y` is the dependent variable and `x` the explanatory variable. The argument `data` sets the data set to be used in the regression. We now revisit the example from the book where the relationship between the test scores and the class sizes is analysed. The following code uses `lm()` to replicate the results presented in figure 4.3 in the book.

```
# estimate the model and assign the result to linear_model
linear_model <- lm(score ~ STR, data = CASchools)

# Print the standard output of the estimated lm object to the console
linear_model
```

```
##
## Call:
## lm(formula = score ~ STR, data = CASchools)
##
## Coefficients:
## (Intercept)          STR
##      698.93        -2.28
```

Let us add the estimated regression line to the plot. This time we also enlarge ranges of both axes by setting the arguments `xlim` and `ylim`.

```
# plot the data
plot(score ~ STR,
     data = CASchools,
     main = "Scatterplot of TestScore and STR",
     xlab = "STR (X)",
     ylab = "Test Score (Y)",
     xlim = c(10, 30),
     ylim = c(600, 720)
     )

# add the regression line
abline(linear_model)
```



Did you notice that this time, we did not pass the intercept and slope parameters to `abline`? If you call `abline` on an object of class `lm` that only contains a single regressor variable, R draws the regression line automatically!

5.2 Measures of Fit

After estimating a linear regression, the question occurs how well that regression line describes the data. Are the observations tightly clustered arround the regression line, or are they spread out? Both, the R^2 and the *standard error of the regression* (SER) measure how well the OLS Regression line fits the data.

The R^2

The R^2 is the fraction of sample variance of Y_i that is explained by X_i. Mathemethically, the R^2 can be written as the ratio of the explained sum of squares to the total sum of squares. The *explained sum of squares* (ESS) is the sum of squared deviations of the predicted values, $\hat{Y}_i$, from the average of the Y_i. The *total sum of squares* (TSS) is the sum of squared deviations of the Y_i from their average.

$$ESS = \sum_{i=1}^{n} \left(\hat{Y}_i - \overline{Y} \right)^2 \tag{5.7}$$

$$\tag{5.8}$$

$$TSS = \sum_{i=1}^{n} \left(Y_i - \overline{Y} \right)^2 \tag{5.9}$$

$$\tag{5.10}$$

$$R^2 = \frac{ESS}{TSS} \tag{5.11}$$

Since $TSS = ESS + SSR$ we can also write

$$R^2 = 1 - \frac{SSR}{TSS}$$

where SSR is the sum of squared residuals, a measure for the errors made when predicting the Y by X. The SSR is defined as

$$SSR = \sum_{i=1}^{n} \hat{u}_i^2.$$

R^2 lies between 0 and 1. It is easy to see that a perfect fit, i.e. no errors made when fitting the regression line, implies $R^2 = 1$ since then we have $SSR = 0$. On the contrary, if our estimated regression line does not explain any variation in the Y_i, we have $ESS = 0$ and consequently $R^2 = 0$.

Standard Error of the Regression

The *Standard Error of the Regression* (SER) is an estimator of the standard deviation of the regression error $\hat{u}_i$. As such it measure the magnitude of a typical deviation from the regression, i.e. the magnitude of a typical regression error.

$$SER = s_{\hat{u}} = \sqrt{s_{\hat{u}}^2} \quad \text{where} \quad s_{\hat{u}}^2 = \frac{1}{n-2} \sum_{i=1}^{n} \hat{u}_i^2 = \frac{SSR}{n-2}$$

Remember that the u_i are *unobserved*. That is why we use their estimated counterparts, the residuals $\hat{u}_i$ instead. See chapter 4.3 of the book for a more detailed comment on the SER.

Application to the Test Score Data

Both measures of fit can be obtained by using the function `summary()` with the `lm` object provided as the only argument. Whereas the function `lm()` only prints out the estimated coefficients to the console, `summary` provides additional predefined information such as the regression's R^2 and the SER.

```
mod_summary <- summary(linear_model)
mod_summary
```

```
##
## Call:
## lm(formula = score ~ STR, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.727 -14.251   0.483  12.822  48.540
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 698.9329     9.4675  73.825  < 2e-16 ***
## STR          -2.2798     0.4798  -4.751 2.78e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.58 on 418 degrees of freedom
## Multiple R-squared:  0.05124,    Adjusted R-squared:  0.04897
## F-statistic: 22.58 on 1 and 418 DF,  p-value: 2.783e-06
```

The R^2 in the output is called 'Multiple R-squared' and has the value 0.051. Hence, 5.1% of the variance of the dependent variable *score* is explained by the explanatory variable STR. That is the regression explains some of the variance but much of the variation in test scores remains unexplained (cf. figure 4.3 in the book).

The SER is called 'Residual standard error' and takes the value 18.58. The unit of the SER is the same as the unit of the dependent variable. In our context we can interpret the value as follows: on average the deviation of the actual achieved test score and the regression line is 18.58 points.

Now, let us check whether the `summary()` function uses the same definitions for R^2 and SER as we do by computing them manually.

```
# compute R^2 manually
SSR <- sum(mod_summary$residuals^2)
TSS <- sum((score - mean(score))^2)
R2 <- 1 - SSR/TSS

# print the value to the console
R2
```

```
## [1] 0.05124009
```

```
# compute SER manually
n <- nrow(CASchools)
SER <- sqrt(SSR / (n-2))

# print the value to the console
SER
```

```
## [1] 18.58097
```

We find that the results coincide. Note that the values provided by `summary()` are rounded to two decimal places. Can you do so using R?

5.3 The Least Squares Assumptions

OLS performs well under a quite broad variety of different circumstances. However, there are some assumptions which are posed on the data which need to be satisfied in order to achieve reliable results.

Key Concept 4.3

The Least Squares Assumptions

$$Y_i = \beta_0 + \beta_1 X_i + u_i, \ i = 1, ..., n$$

where

1. The error term u_i has conditional mean zero given X_i: $E(u_i|X_i) = 0$
2. $(X_i, Y_i), i = 1, ..., n$ are independent and identically distributed (i.i.d.) draws from their joint distribution
3. Large outliers are unlikely: X_i and Y_i have nonzero finite fourth moments

Assumption #1: The Error Term has Conditional Mean of Zero

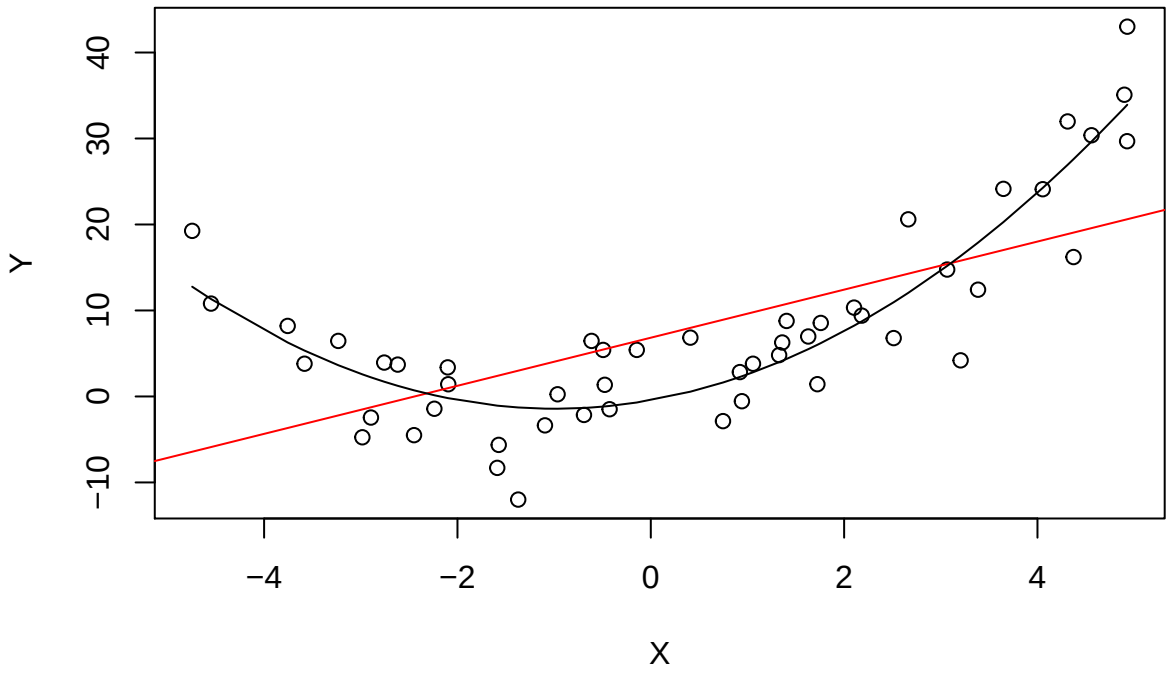
This means that no matter which value we choose for X, the error term u must not show any systematic pattern and must have a mean of 0. Consider the case that $E(u) = 0$ but for low and high values of X, the error term tends to be positive and for midrange values of X the error tends to be negative. We can use R to construct such an example. To do so we generate our own data using R's build in random number generators.

We will use the following functions You should be familiar with:

- `runif()` (generates uniformly distributed random numbers)
- `rnorm()` (generates nomally distributed random numbers)
- `predict()` (does predictions based on the results of model fitting functions like `lm()`)
- `lines()` (adds line segments to an existing plot)

We start by creating a vector containing values that are randomly scattered on the domain $[-5, 5]$. For our example we decide to generate uniformly distributed random numbers. This can be done with the function `runif()`. We also need to simulate the error term. For this we generate normally distributed random numbers with a mean equal to 0 and a variance of 1 using `rnorm()`. The Y values are obtained as a quadratic function of the X values and the error. After generating the data we estimate both a simple regression model and a quadratic model that also includes the regressor X^2. Finally, we plot the simulated data and add a the estimated regression line of a simple regression model as well as the predictions made with a quadratic model to compare the fit graphically.

```
# set a random seed to make the results reproducible
set.seed(321)

# simulate the data
X <- runif(50, min = -5, max = 5)
u <- rnorm(50, sd = 5)
## the true relation
Y <- X^2 + 2*X + u

# estimate a simple regression model
mod_simple <- lm(Y ~ X)

# predict using a quadratic model
prediction <- predict(lm(Y ~ X +  I(X^2)), data.frame(X = sort(X)))

# plot the results
plot(Y ~ X)
abline(mod_simple, col = "red")
lines(sort(X), prediction)
```



This shows what is meant by $E(u_i|X_i) = 0$:

Using the quadratic model (represented by the black curve) we see that there are no systematic deviations of the observation from the predicted relation. It is credible that the assumption is not violated when such a model is employed. However, using a simple linear regression model we see that the assumption is probably violated as $E(u_i|X_i)$ varies with the X_i.

Assumption #2: All (X_i, Y_i) are Independently and Identically Distributed

Most common sampling schemes used when collecting data from populations produce i.i.d. samples. For example, we could use R's random number generator to randomly select student IDs from a university's enrollment list and record age X and earnings Y of the corresponding students. This is a typical example of simple random sampling and ensures that all the (X_i, Y_i) are drawn randomly from the same population.

A prominent example where the i.i.d. assumption is not fulfilled is time series data where we have observations on the same unit over time. For example, take X as the number of workers employed by a production company over the course of time. Due to technological change, the company makes job cuts periodically but there are also some non-deterministic influences that relate to economics, politics and alike. Using R we can simulate such a process and plot it.

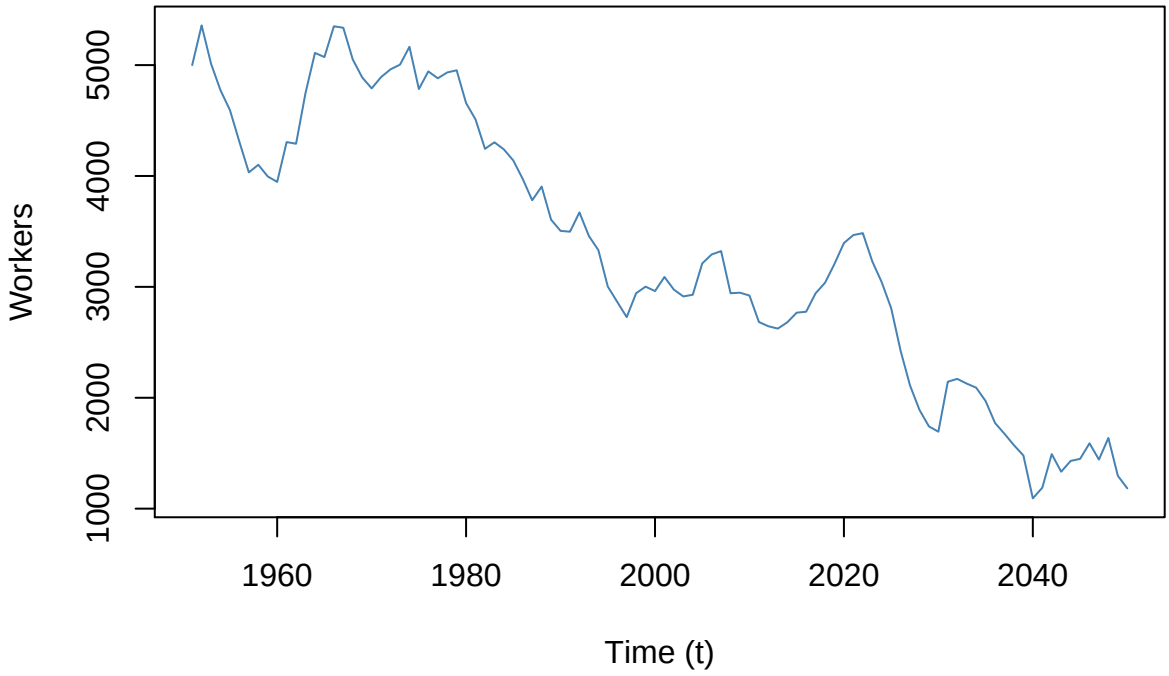
We start the series with a total of 5000 workers and simulate the reduction of employment with a simple autoregressive process that exhibits a downward trend and has normal distributed errors:[1]

$$employment_t = 0.98 \cdot employment_{t-1} + u_t$$

```
# set random seed
set.seed(7)

# initialize the employment vector
X <- c(5000,rep(NA,99))
```

[1] See chapter 14 in the book for more on autoregressive processes and time series analysis in general.

```r
# generate a date vector
Date <- seq(as.Date("1951/1/1"), as.Date("2050/1/1"), "years")

# generate time series observations with random influences
for (i in 2:100) X[i] <- 0.98*X[i-1] + rnorm(1, sd=200)

#plot the results
plot(Date, X, type = "l", col="steelblue", ylab = "Workers", xlab="Time (t)")
```



It is evident that the observations on X cannot be independnet in this example: the level of today's employment is correlated with tomorrows employment level. Thus, the i.i.d. assumption is violated for X.

Assumption #3: Large outliers are unlikely

It is easy to come up with situations where extreme observations, i.e. observations that deviate considerably from the usual range of the data, may occur. Such observations are called outliers. Technically speaking, assumption #3 requires that X and Y have a finite kurtosis.[2]

Common cases where we want to exclude or (if possible) correct such outliers is when they are apperently typos, conversion errors or measurement errors. Even if it seems that extreme observations have been recorded correctly, it is advisable to exclude them before estimating a model since OLS suffers from *sensitivity to outliers*.

What does this mean? One can show that extreme observation receive heavy weighting in the computation done with OLS. Therefore, outliers can lead to strongly distorted estimates of regression coefficient. To get a better impression of this, consider the following application where we have placed some sample data on X and Y which are highly correlated. The relation between X and Y seems to be explained pretty good by the plotted regression line: all of the blue dots lie close to the red line and we have $R^2 = 0.92$.

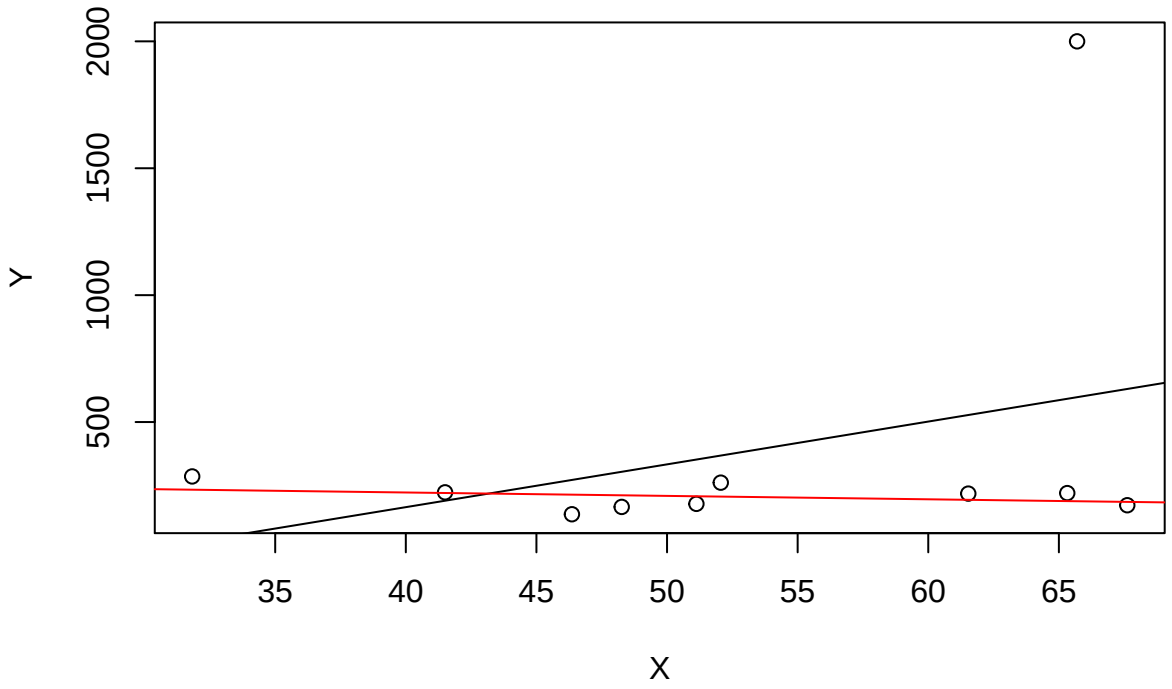
Now go ahead and add a further observation at, say, $(18, 2)$. This clearly is an outlier. The result is quite striking: the estimated regression line differs greatly from the one we adjudged to fit the data well. The slope

[2] See chapter 4.4 in the book.

is heavily downward biased and R^2 decreased to a mere 29%! Double-click inside the coordinate system to reset the app. Feel free to experiment. Choose different coordinates for the outlier or add additional ones.

The following code roughly reproduces what is shown in figure 4.5 in the book. As done above we use sample data generated using R's random number functions `rnorm()` and `runif()`. We estimate simple regression models based on the original data set and a modified set where one observation is change to be an outlier and plot the results. In order to understand the complete code You should be familiar with the function `sort()` which sorts the entries of a numeric vector in ascending order.

```
# set random seed
set.seed(123)

# generate the data
X <- sort(runif(10, min = 30, max = 70 ))
Y <- rnorm(10 , mean = 200, sd = 50)
Y[9] <- 2000

# fit model with outlier
fit <- lm(Y ~ X)

# fit model without outlier
fitWithoutOutlier <- lm(Y[-9] ~ X[-9])

# plot the results
plot(Y ~ X)
abline(fit)
abline(fitWithoutOutlier, col = "red")
```



5.4 The Sampling Distribution of the OLS Estimator

Because the OLS estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ are computed from a randomly drawn sample, the estimators themselves are random variables with a probability distribution — the so-called sampling distribution of the estimators — which describes the values they could take over different random samples. Although the sampling distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$ can be complicated when the sample size is small and generally differs with the number of observation, n, it is possible to make certain statements about it that hold for all n. In particular

$$E(\hat{\beta}_0) = \beta_0 \quad \text{and} \quad E(\hat{\beta}_1) = \beta_1,$$

that is, $\hat{\beta}_0$ and $\hat{\beta}_1$ are unbiased estimators of β_0 and β_1, the true parameters. If the sample is sufficiently large, by the central limit theorem the *joint* sampling distribution of the estimators is well approximated by the bivariate normal distribution (2.1). This implies that the marginal distributions are also normal in large samples. Core facts on the large-sample distribution of β_0 and β_1 are presented in Key Concept 4.4.

Key Concept 4.4

Large Sample Distribution of $\hat{\beta}_0$ and $\hat{\beta}_1$

If the least squares assumptions in Key Concept 4.3 hold, then in large samples $\hat{\beta}_0$ and $\hat{\beta}_1$ have a jointly normal sampling distribution. The large sample normal distribution of $\hat{\beta}_1$ is $N(\beta_1, \sigma^2_{\hat{\beta}_1})$, where the variance of the distribution, $\sigma^2_{\hat{\beta}_1}$, is

$$\sigma^2_{\hat{\beta}_1} = \frac{1}{n}\frac{Var\left[(X_i - \mu_X)\, u_i\right]}{\left[Var\left(X_i\right)\right]^2}. \tag{4.1}$$

The large sample normal distribution of $\hat{\beta}_0$ is $N(\beta_0, \sigma^2_{\hat{\beta}_0})$, where

$$\sigma^2_{\hat{\beta}_0} = \frac{1}{n}\frac{Var\left(H_i u_i\right)}{\left[E\left(H_i^2\right)\right]^2}, \text{ where } H_i = 1 - \left[\frac{\mu_X}{E\left(X_i^2\right)}\right] X_i. \tag{4.2}$$

R Simulation Study 1

Whether Key Koncept 4.4 really holds can be verified using R. First we build our own population of 100000 observations in total. To do this we need values for our independent variable X, for the error term u, and the regression parameters β_0 and β_1. With all this combined in a simple regression model, we can compute our dependent variable Y. In our example we generate the numbers X_i, $i = 1, \dots, 100000$ by drawing a random sample from a uniform distribution on the interval $[0, 20]$. The realisations of the error terms u_i are drawn from a standard normal distribution with parameters $\mu = 0$ and $\sigma^2 = 100$ (note that `rnorm()` requires σ as input for the argument `sd`, see `?rnorm`). Furthermore we chose $\beta_0 = -2$ and $\beta_1 = 3.5$ so the true model is

$$Y_i = -2 + 3.5 \cdot X_i.$$

Finally, we store the results in a data.frame.

```
# simulate data
N <- 100000
X <- runif(N, min = 0, max = 20)
u <- rnorm(N, sd = 10)

# population regression
Y <- -2 + 3.5 * X + u
population <- data.frame(X, Y)
```

From now on we will consider the previously generated data as the true population (which of course would be *unknown* in a real world application, otherwise there would not be a reason to do draw a random sample in the first place). The knowledge about the true population and the true relationship between Y and X can be used to verify the statements made in Key Concept 4.4.

First, let us calculate the true variances $\sigma^2_{\hat{\beta}_0}$ and $\sigma^2_{\hat{\beta}_1}$ for a randomly drawn sample of size $n = 100$.

```
# set sample size
n <- 100

# compute the variance of hat_beta_0
H_i <- 1 - mean(X) / mean(X^2) * X
var_b0 <- var(H_i * u) / (n * mean(H_i^2)^2 )

# compute the variance of hat_beta_1
var_b1 <- var( ( X - mean(X) ) * u ) / (100 * var(X)^2)
```

```
# print variances to the console
var_b0
```

```
## [1] 4.045066
```

```
var_b1
```

```
## [1] 0.03018694
```

Now let us assume that we do not know the true values of β_0 and β_1 and that it is not possible to observe the whole population. However, we can observe a random sample of n observations. Then, it would not be possible to compute the true parameters but we could obtain estimates of β_0 and β_1 from the sample data using OLS. However, we know that these estimates are outcomes of random variables themselves since the observations are randomly sampled from the population. Key Concept 4.4. describes their distributions for large n. When drawing a single sample of size n it is not possible to make any statement about these distributions. Things change if we repeat the sampling scheme many times and compute the estimates for each sample: using such a procedure we simulate outcomes of the respective distributions.

To achieve this in R, we employ the following approach:

- We assign the number of repetitions, say 10000, to `reps`. Then we initialize a matrix `fit` were the estimates obtained in each sampling iteration shall be stored row-wise. Thus `fit` has to be an array of dimensions `reps`$\times 2$.
- In the next step we draw `reps` random sample of size `n` from the population and obtain the OLS estimates for each sample. The results are stored as row entries in the outcome matrix `fit`. This is done using a `for()` loop.
- At last, we estimate variances of both coefficient estimators using the sampled outcomes and plot histograms of the latter. We also add plot of the density functions belonging to the distributions that follow from Key Concept 4.4. The function `bquote()` is used to obtain math expressions in the titels and labels of both plots. See `?bquote`.

```
# set repetitions and sample size
n <- 100
reps <- 10000

# initialize the matrix of outcomes
fit <- matrix(ncol = 2, nrow = reps)

# loop sampling and estimating of the coefficients
for (i in 1:reps){
 sample <- population[sample(1:N, n),]
```

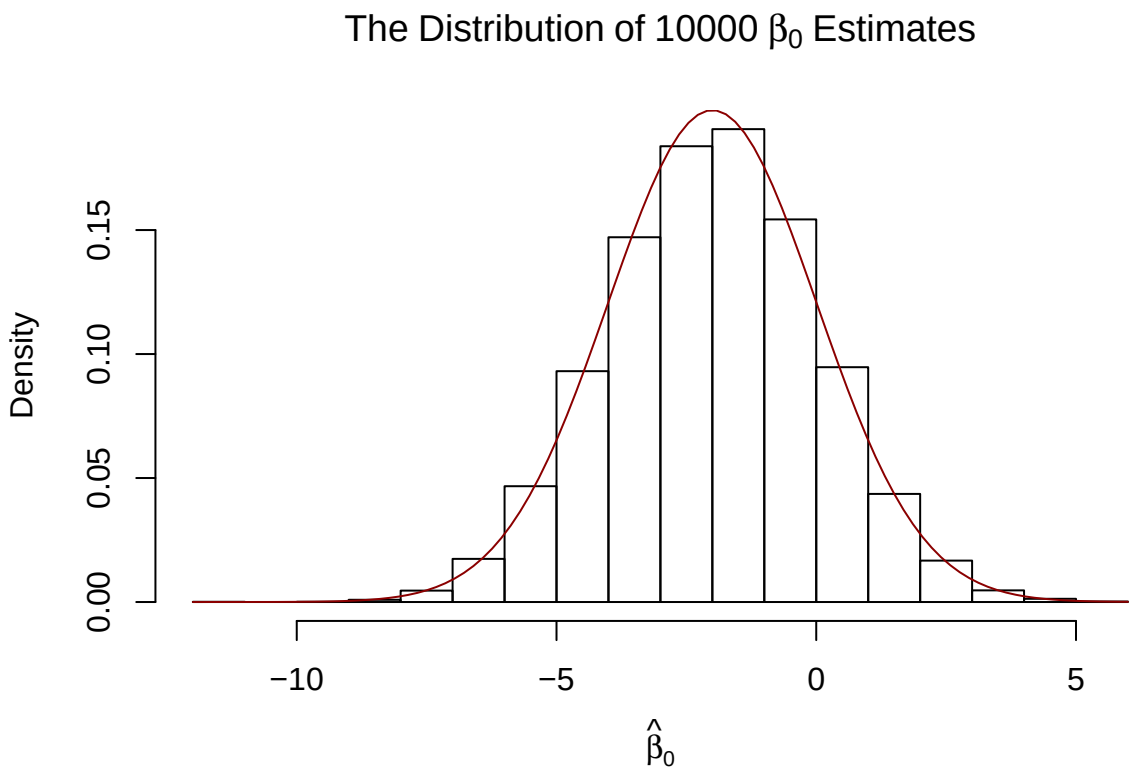
```
 fit[i, ] <- lm(Y ~ X, data = sample)$coefficients
}

# compute variance estimates using outcomes
var(fit[ ,1])
```

```
## [1] 4.057089
```

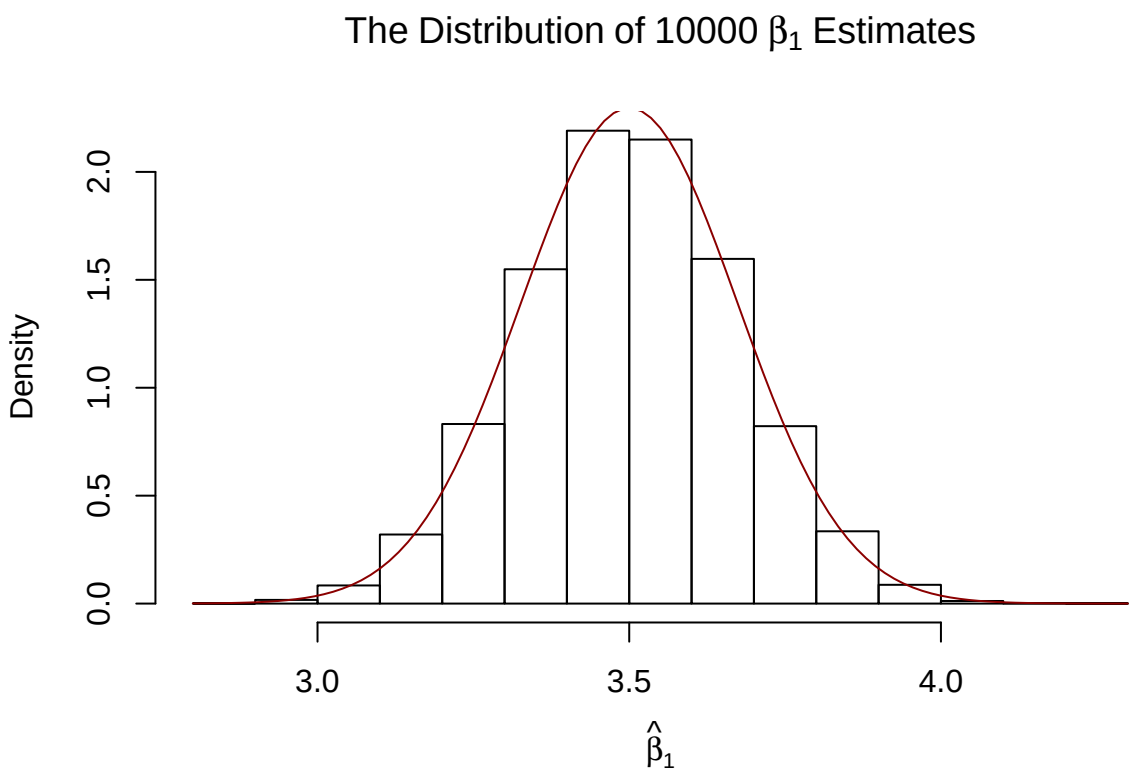
```
var(fit[ ,2])
```

```
## [1] 0.03021784
```

```
# plot histograms of beta_0 estimates
hist(fit[ ,1],
     main = bquote(The ~ Distribution  ~ of ~ 10000 ~ beta[0] ~ Estimates),
     xlab = bquote(hat(beta)[0]),
     freq = F)
# add true distribution to plot
curve(dnorm(x,-2,sqrt(var_b0)), add = T, col="darkred")
```




```
# plot histograms of beta_1 estimates
hist(fit[ ,2],
     main = bquote(The ~ Distribution  ~ of ~ 10000 ~ beta[1] ~ Estimates),
     xlab = bquote(hat(beta)[1]),
     freq = F)

# add true distribution to plot
curve(dnorm(x,3.5,sqrt(var_b1)), add = T, col="darkred")
```



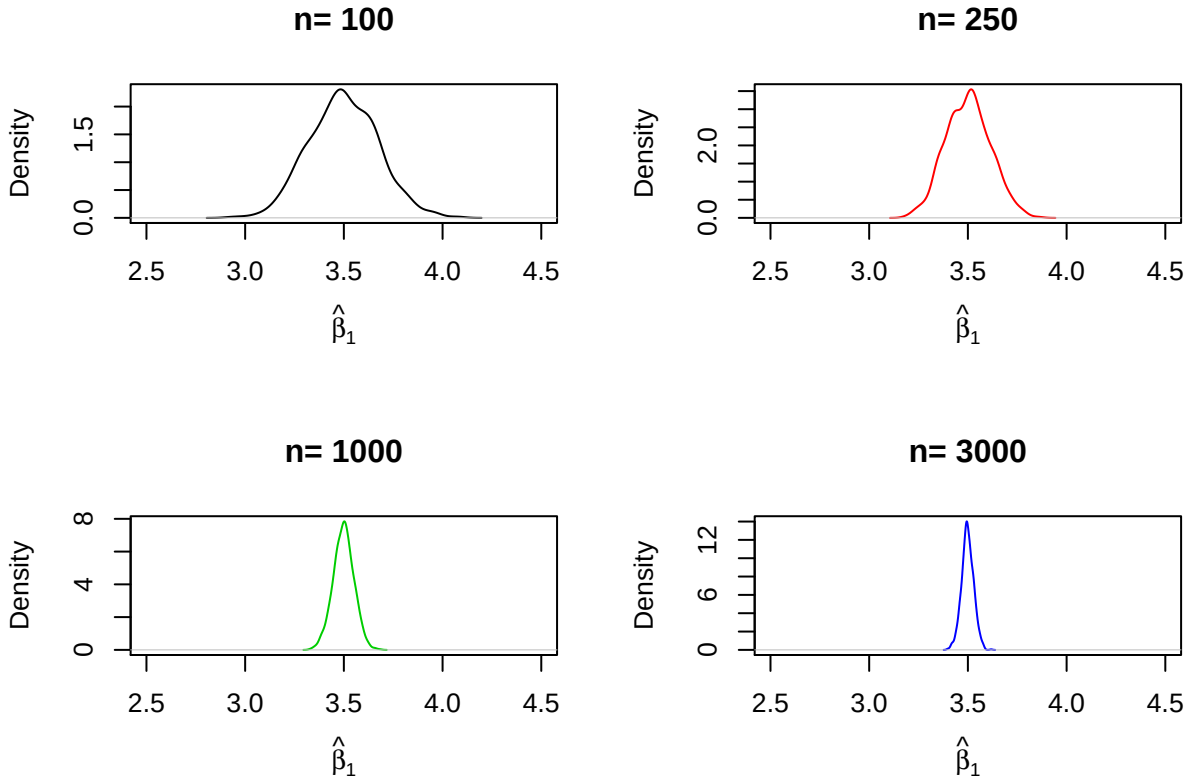
We are now able to say the following: first, our variance estimates are in favour of the claims made in Key Concept 4.4 since they come close to the computed theoretical values. Second, the histograms suggest that the estimators distributions indeed follow normal distributions which can be fairly approximated by the respective normal distributions stated in Key Concept 4.4.

R Simulation Study 2

A further result implied by Key Concept 4.4 is that both estimators are consistent i.e. they converge in probability to their true value. This is since their variances converge to 0 as n increases. We can check this by repeating the simulation above for an increasing sequence of sample sizes. This means we no langer assign the sample size but a *vector* of sample sizes: `n <- c(...)`. Let us look at the distributions of β_1. The idea here is to add an additional call of `for()` to the code. This is done in order to loop over the vector of sample sizes `n`. For each of the sample sizes we carry out the same simulation as before but plot a density estimate for the outcomes of each iteration over `n`. Notice that we have to change `n` to `n[j]` in the inner loop to ensure that the `j`th element of `n` is used. In the simulation, we use sample sizes $100, 250, 1000$ and 3000. Consequently we have a total of four distinct simulations using different sample sizes.

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

# set repetitions and the vector of sample sizes
reps <- 1000
n <- c(100, 250, 1000, 3000)

# initialize the matrix of outcomes
fit <- matrix(ncol = 2, nrow = reps)

# devide the plot panel in a 2-by-2 array
par(mfrow = c(2,2))
```

```
#### Loop sampling and plotting ####

# outer loop over n
for (j in 1:length(n)) {

  # inner loop: sampling and estimating of the coefficients
  for (i in 1:reps){
    sample <- population[sample(1:N, n[j]), ]
    fit[i, ] <- lm(Y ~ X, data = sample)$coefficients
  }

  # draw density estimates
  plot(density(fit[,2]), xlim=c(2.5,4.5), col=j,
       main = paste("n=", n[j]), xlab = bquote(hat(beta)[1]))

}
```



We find that, as n increases, the distribution of $\hat{\beta}_1$ concentrates around its mean, i.e. its variance decreases. Put differently, the likelihood of observering estimates close to the true value of $\beta_1 = 3.5$ grows as we increase the sample size. The same behaviour could be observed if we would analyze the distribution of $\hat{\beta}_0$ instead.

R Simulation Study 3

Furthermore, (4.1) reveals that the variance of the OLS estimator for β_1 decreases as the variance of the X_i increases. In other words, as we increase the amount of information provided by the regressor, that is increasing $Var(X)$, which is used to estimate β_1, we are more confident that the estimate is close to the true value (i.e. $Var(\hat{\beta}_1)$ decreases). We can visualize this by reproducing figure 4.6 from the book. To do this, we sample 100 observations (X, Y) from a bivariate normal distribution with

$$E(X) = E(Y) = 5,$$

$$Var(X) = Var(Y) = 5$$

and

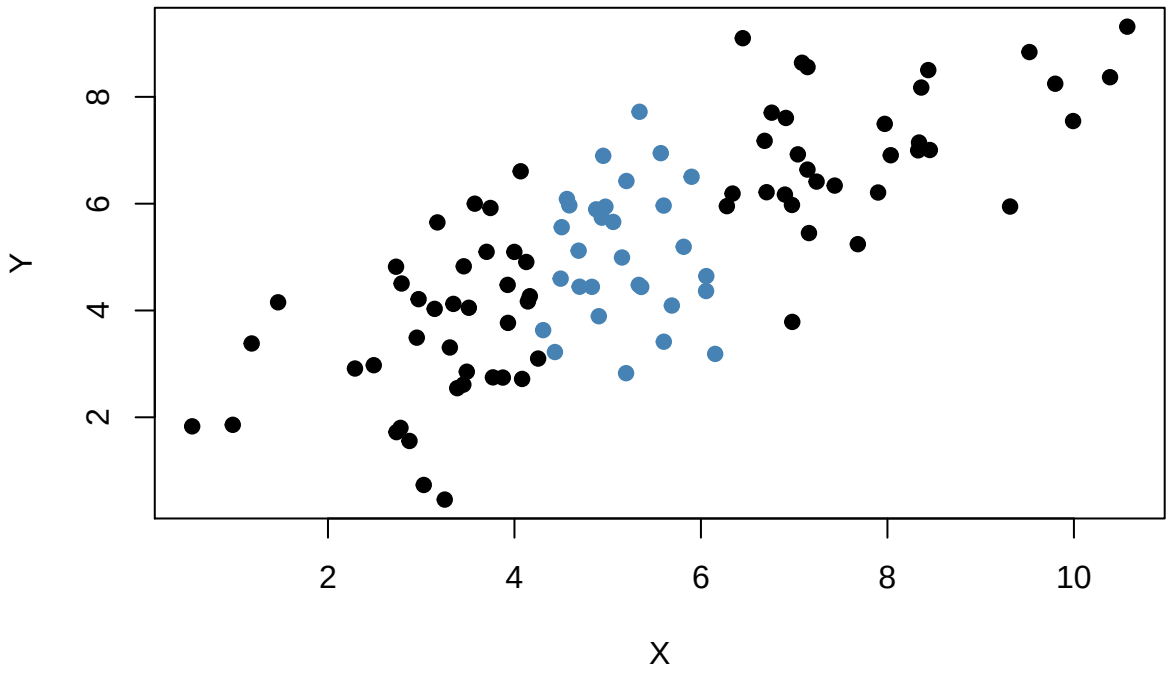
$$Cov(X, Y) = 4.$$

Formally, this is written down as

$$\begin{pmatrix} X \\ Y \end{pmatrix} \overset{i.i.d.}{\sim} \mathcal{N}\left[\begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix} \right]. \tag{4.3}$$

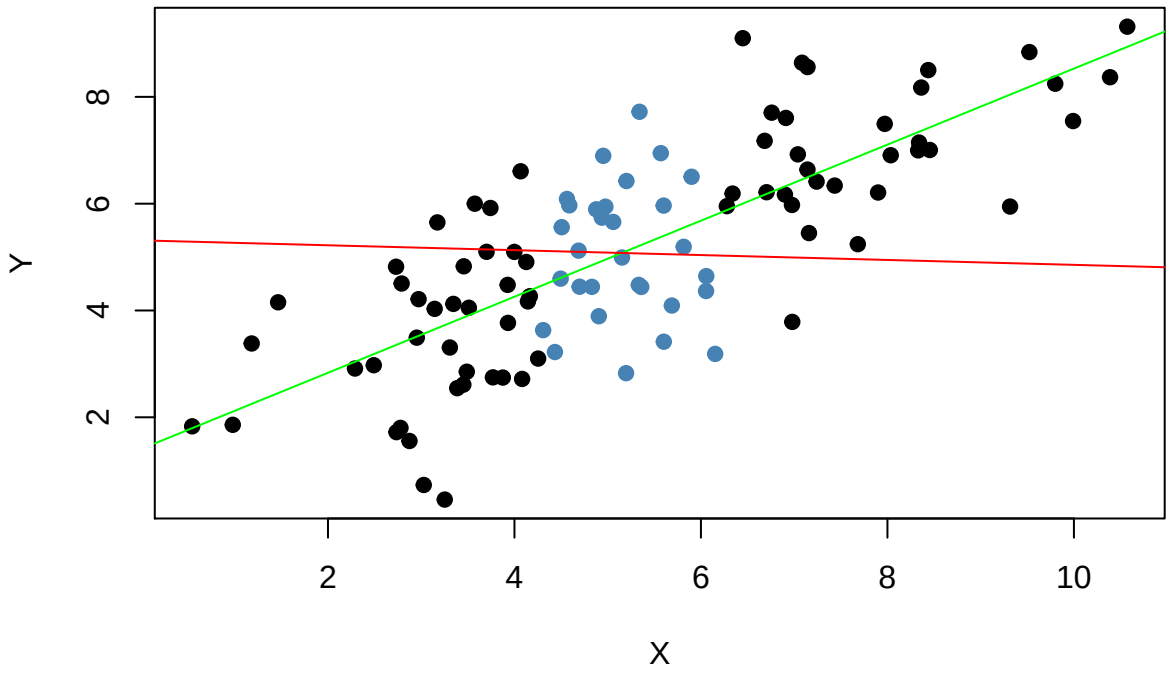
To carry out the random sampling, we make use of the function `mvtnorm()` from the package `MASS` which allows to draw random samples from multivariate normal distributions, see `?mvtnorm`. Next, we use the `subset()` function to split the sample into two subsets such that the first set, `set1`, consists of observations that fulfill the condition $|X - \overline{X}| > 1$ and the second set, `set2`, includes the remainder of the sample. We then plot both sets and use different colors to make them distinguishable.

```
# load the MASS package
library(MASS)

# set random seed for reproducibility
set.seed(4)

# simulate bivarite normal data
bvndata <- mvrnorm(100,
                   mu = c(5,5),
                   Sigma = cbind(c(5,4),c(4,5))
                   )

# assign column names / convert to data.frame
colnames(bvndata) <- c("X","Y")
bvndata <- as.data.frame(bvndata)

# subset the data
set1 <- subset(bvndata, abs(mean(X) - X) > 1)
set2 <- subset(bvndata, abs(mean(X) - X) <= 1)

# plot both data sets
plot(set1, xlab = "X", ylab = "Y", pch = 19)
points(set2, col = "steelblue", pch = 19)
```



It is clear that observations that are close to the sample average of the X_i have less variance than those that are farther away. Now, if we were to draw a line as accurately as possible through either of the two sets it is obvious that choosing the observations indicated by the black dots, i.e. using the set of observations which has larger variance than the blue ones, would result in a more precise line. Now, let us use OLS to estimate and draw the regression lines for both sets of observations.

```
# estimate both regression lines
lm.set1 <- lm(Y ~ X, data = set1)
lm.set2 <- lm(Y ~ X, data = set2)

# add both lines to the plot
abline(lm.set1, col="green")
abline(lm.set2, col="red")
```



Evidently, the green regression line does far better in describing data sampled from the bivariate normal distribution stated in (4.3) than the red line. This is a nice example why we are interested in a high variance of the regressor X: more variance in the X_i means more information from which the precision of the estimation benefits.

5.5 Exercises

1. Class Sizes and Test Scores

A researcher wants to analyse the relationship between class size (measured by the student-teacher ratio) and the pupils' average test score. Therefore he measures both variables in 10 different classes and ends up with the following results.

```
<tr>
  <td><b>Class Size</b></td>
  <td>23</td>
  <td>19</td>
  <td>30</td>
  <td>22</td>
  <td>23</td>
  <td>29</td>
  <td>35</td>
  <td>36</td>
  <td>33</td>
  <td>25</td>
</tr>
<tr>
  <td><b>Test Score</b></td>
  <td>430</td>
  <td>430</td>
  <td>333</td>
```

```
    <td>410</td>
    <td>390</td>
    <td>377</td>
    <td>325</td>
    <td>310</td>
    <td>328</td>
    <td>375</td>
  </tr>
</table>
```

Instructions:

- Create the vectors cs (the class sizea) and ts (the test scorea), containing the obeservations above.
- Draw a scatter plot of the results using plot().

<code data-type="sample-code">

Create both vectors

Draw the scatter plot

Compute mean, median, variance & standard deviation of test score

Compute the covariance and the correlation coefficient

<code data-type="solution">

Create both vectors cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375)

Draw the scatter plot plot(cs,ts)

<code data-type="sct">

test_object(“cs”) test_object(“ts”)

test_function(“plot”)

2. Mean, Variance, Covariance and Correlation

The vectors cs and ts are available in the working environment (you can check this: type their names to the console and then press enter).

Instructions:

- Compute the mean, the sample variance and the sample standard deviation of ts.
- Compute the covariance and the correlation coefficient for ts and cs.

Use the R functions presented in this chapter: mean(), sd(), cov(), cor() and var()

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375)

<code data-type="sample-code">

Compute mean, variance and standard deviation of test scores

Compute the covariance and the correlation coefficient

<code data-type="solution">

mean(ts) sd(ts) var(ts) cov(ts,cs) cor(ts,cs)

<code data-type="sct">

test_predefined_objects(“cs”) test_predefined_objects(“ts”)

test_function(“mean”, args = “x”)

test_function(“sd”, args = “x”)

test_output_contains(“cov(ts,cs)”)

test_output_contains(“cor(ts,cs)”)

3. Simple Linear Regression

The vectors cs and ts are available in the working environment.

Instructions:

- The function lm() is part of the package AER. Attach it using library().
- Use lm() to estimate the regression model

$$TestScore_i = \beta_0 + \beta_1 STR_i + u_i.$$

- Obtain a stastistical summary of the model.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) # attach the package AER

estimate the model

model summary

library(AER) lm(cs ~ ts) summary(lm(cs ~ ts))

test_predefined_objects(c(“ts”,“cs”)) test_student_typed(“library(AER)”) test_output_contains(“lm(ts ~ cs)”) test_output_contains(“summary(lm(ts ~ cs))”)

4. The Model Object

Let’s see how a model object is structured.

The vectors cs and ts as well as the model object mod from the previous exercise are available in your workspace.

- Use class() to learn about the class of the object mod
- mod is an object of type list with named entries. Check this using the function is.list().
- See what information you can obtain from mod using names().
- Read out an arbitrary entry of the object mod using the $ operator.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # check the class of ‘mod’

```
# use 'is.list()'


# check entry names of mod using 'names()'


# use the operator '$' on 'mod'
```

class(mod) is.list(mod) mod$fitted.values < /code >< codedata−type = sct > test_predefined_objects(mod)test_function(clas
x)test_function(is.list, args = x)test_student_typed(mod“)

5. Plotting the Regression Line

Now add a regression line to the scatter plot from a few exercises before. You are provided with the code for the scatter plot in script.R

The object mod is available in your working environment.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # Add the regression line the the scatterplot plot(cs,ts) plot(cs,ts) abline(mod) test_predefined_objects("mod") test_function("plot") test_function("abline")

6. Summary of a Model Object

Now read out and store some of the information that is contained in the output produced by summary():

- Assign the output of summary(mod) to the variable 's'
- Check entry names of 's'
- Create a new variable R2 storing the regression's R^2.

The object mod is available in your working environment.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # Assign the model summary to the variable 's'

```
# Check names of entries in 's'


# Store the regression's R^2 in the variable 'R2'
```

s <- summary(mod) names(s) R2 <- s$r.squared test_predefined_objects("mod") test_function("names", args = "x") test_object("s") test_object("R2")

7. Estimated Coefficients

summary() provides information on the statistical significance of the estimated coefficients.

Extract the named 2×4 matrix with estimated coefficients, standard errors, t-statistics and corresponding p-values from the model summary s. Save this matrix in an object named coefs.

The objects mod and s are available in your working environment.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) s <- summary(mod) # Save the coefficient matrix to 'coefs'

coefs <- s$coefficients test_predefined_objects("s") test_object("coefs")

8. Dropping the Intercept

So far, we have done regressions where the model consisted of an intercept and a single regressor. In this exercise you will learn ways to specify and to estimate regression models excluding the intercept.

Note that excluding the intercept from a regression model might be a dodgy practice in some models as this imposes the conditional expectation function of the dependent variable to be zero if the regressor is zero.

- Use the help function on lm (?lm) to find out how to specify the formula argument for a regression of ts solely on cs, i.e. a regression without intercept.
- Estimate the regression model without intercept and store the result in mod_ni.

The vectors cs, ts and the model object mod from previous exercises are availabe in the working environment.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) s <- summary(mod) # Regress ts solely and cs, store the result in 'mod_ni'

mod_ni <- lm(ts ~ cs - 1) # or: lm(ts ~ cs + 0) test_predefined_objects("mod")

```
test_or(
  {
  test_student_typed("ts ~ cs + 0")
},{
  test_student_typed("ts ~ cs - 1")
}
)

test_function("lm")

test_object("mod_ni", eval=F)
```

9. Regression Output: No Constant Case

You have estimated a model without intercept. The estimated regression function is

$$\widehat{TestScore} = \underset{(1.36)}{12.65} \times STR.$$

Convince yourself that everything is as stated above: extract the coefficient matrix from the summary of mod_ni and store it in a variable named coef.

The vectors cs, ts as well as the model object mod_ni from the previous exercise are availabe in your working environment.

Remember: an entry of a named list can be accessed with the $ operator.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) mod_ni <- lm(ts ~ cs - 1) # Extract the coefficient matrix from the model's summary and save it to 'coef'

coef <- summary(mod_ni)$coefficients test_predefined_objects("mod") test_predefined_objects("mod_ni") test_object("coef") success_msg("Right! Note that there is only one coefficient estimate (the coefficient on cs) reported by summary(mod_ni)")

10. Regression Output: No Constant Case — Ctd.

You have estimated a model without intercept. The estimated regression function is

$$\widehat{TestScore}_i = \underset{(1.36)}{12.65} \times STR_i.$$

The coefficient matrix coef contains the estimated coefficient on STR, its standard error, the t-statistic of the significance test and the corresponding p-value.

- Print the contents of coef to the console.
- Convince yourself that the reported t-statistic is correct: use the entries of coef to compute the t-statistic and save it to t_stat.

The matrix coef from the previous exercise is availabe in your working environment.

Hints: X[a,b] returns the [a,b] element of a matrix X. The t-statistic for this significance test is computed as

$$t = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)}.$$

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) coef <- summary(mod)$coef # print the contents of 'coef' to the console

compute the t-statistic manually

print(coef) t_stat <- coef[1,1]/coef[1,2] test_predefined_objects("coef") test_output_contains("coef") test_object("t_stat")

11. Two Regressions, One Plot

The two estimated regression equations are

$$\widehat{TestScore}_i = \underset{(1.36)}{12.65} \times STR_i$$

and

$$\widehat{TestScore}_i = \underset{(23.9606)}{567.4272} \underset{(0.8536)}{-7.1501} \times STR_i.$$

You are provided with the code line plot(cs,ts) which creates a scatter plot of ts and cs. It must be executed before calling abline()! You may color the regression lines by using e.g. col = "red" or col = "blue" as an additional argument to abline() for better distinguishability.

The vectors cs and ts as well as the list objects mod and mod_ni from previous exercises are availabe in your working environment.

Generate a scatter plot of the ts and cs and add the estimated regression lines of mod and mod_ni.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) mod_ni <- lm(ts ~ cs - 1) # Plot regression lines for both models plot(cs, ts)

plot(cs,ts) abline(mod, col="blue") abline(mod_ni, col="red") test_predefined_objects("mod") test_predefined_objects("mod_ni") test_function("abline", index=1) test_function("abline", index=2) success_msg("Yep, that looks good!")

12. TSS and SSR

If graphical inspection does not help, researchers resort to analytic techniques in order to detect if a model fits the data at hand good or better than another model.

Let us go back to the simple regression model including an intercept. The estimated regression line for mod was

$$\widehat{TestScore}_i = 567.43 - 7.15 \times STR_i,\ R^2 = 0.8976,\ SER = 15.19.$$

You can check this as mod and the vectors cs and ts are available in your working environment.

Now, please do the following:

- Compute SSR, the sum of squared residuals, and save it to ssr
- Compute TSS, the total sum of squares, and save it to tss

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # Compute the SSR and save it to ssr

Compute the TSS and save it to tss

ssr <- sum(mod$residuals^2) tss <- 9*var(ts) # Note: var() computes the unbiased sample variance! => Corrected for by multiplying with (n-1) = 9 test_predefined_objects("mod") test_object("ssr") test_object("tss")

13. The R^2 of a Regression Model

The R^2 for the regression model stored in mod is 0.8976. You can check this by executing summary(mod)$r.squared in the console below.

Remember the formula for R^2:

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{SSR}{TSS}$$

The objects mod, tss and ssr from the previous exercise are available in your working environment.

- Use ssr and tss to compute R^2 manually. *Round* the result to four decimal places and save it to R2.
- Use the logical expression == to check whether your result equals the value mentioned above.

You can round numeric values using the function round(). See ?round.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) ssr <- sum(mod$residuals^2) tss <- 9*var(ts) # Compute R^2, round it and save it to 'R2'

Check whether your result is correct using the '==' operator

R2 <- round(1-ssr/tss,4) R2 == 0.8976 test_predefined_objects("ssr") test_predefined_objects("tss") test_predefined_objects("mod") test_object("R2") test_function("round", args="digits", eq_condition="equal") test_student_typed("R2 == 0.8976", not_typed_msg = "Something's wrong. Make sure you type `R2 ==` followed by the value mentioned above.") success_msg("Great!")

14. The Standard Error of The Regression

The standard error of the Regression in the simple regression model is

$$SER = \frac{1}{n-2}\sum_{i=1}^{n}\widehat{u}_i^2 = \sqrt{\frac{SSR}{n-2}}.$$

It mesaures the size of an average residual which is an estimate of the magnitude of a typical regression error.

- Use summary() to obtain the SER for the regression of $TestScore$ on STR stored in the model object mod. Save the value in the variable SER.
- Use SER to compute the model's SSR and store it in SSR.
- Check that SSR is indeed the model's SSR by comparing SSR to the result of sum(mod$residuals^2)

The model object mod and the vectors cs and ts are available in your workspace.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # Obtain the SER using 'summary()' and save the value to 'SER'

Compute the model's SSR and save it to 'SSR'

Do the comparison

SER <- summary(mod)$sigma SSR <- -SER^2 * (length(ts) - 2) SSR == sum(mod$residuals^2)

test_object("SER") test_object("SSR") test_student_typed("SSR == sum(mod$residuals^2)") success_msg("Nice!")

15. The Estimated Covariance Matrix

As has been discussed in Chapter 4.4, the OLS estimators $\widehat{\beta}_0$ and $\widehat{\beta}_1$ can be considered random variables since they are functions of the random error term. For two or more random variables, the covariances and variances are summarized by their covariance matrix. Taking the square root of the diagonal elements of the model's estimated covariance matrix obtains the standard errors of $\widehat{\beta}_0$ and $\widehat{\beta}_1$.

summary() computes an estimate of this matrix. The respective entry is called cov.unscaled.

- Use summary() to obtain the covariance matrix estimate for the regression of $TestScore$ on STR stored in the model object mod. Save the matrix to cov_matrix.
- Obtain the diagonal elements of cov_matrix, compute their square root and assign the result to the variable SEs.

The model object mod is available in your workspace.

Hint: diag(A) returns the diagonal elements of the matrix A.

cs <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) ts <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(ts ~ cs) # obtain the matrix and save it to 'cov_matrix'

compute the standard errors and assign them to the vector 'SEs'

cov_matrix <- summary(mod)$cov.unscaled SEs <- sqrt(diag(cov_matrix)) test_object("cov_matrix") test_object("SEs")

Chapter 6

Hypothesis Tests and Confidence Intervals in the Simple Linear Regression Model

In this chapter, we continue with the treatment of the simple linear regression model. The following subsction discuss how we may use our knowledge about the sampling distribution of the OLS estimator in order to make statements regarding its uncertainty. These subsections cover the following topics:

- Testing Hypotheses about regression coefficients
- Confidence intervals for regression coefficients
- Regression when X is a dummy variable
- Heteroskedasticity and Homoskedasticity

6.1 Testing Two-Sided Hypotheses Concerning β_1

Using the fact that $\hat{\beta}_1$ is approximately normal distributed in large samples (see Key Concept 4.4), testing hypotheses about the true value β_1 can be done with the same approach as discussed in chapter 3.2.

Key Concept 5.1

General Form of the t-Statistic

Remember from chapter 3 that a general t-statistic has the form

$$t = \frac{\text{estimated value} - \text{hypothesized value}}{\text{standard error of the estimator}}.$$

Key Concept 5.2

Testing Hyotheses about β_1

For testing the hypothesis $H_0 : \beta_1 = \beta_{1,0}$, we need to perform the following steps:

1. Compute the standard error of $\hat{\beta}_1$, $SE(\hat{\beta}_1)$

$$SE(\hat{\beta}_1) = \sqrt{\hat{\sigma}^2_{\hat{\beta}_1}} \quad , \quad \hat{\sigma}^2_{\hat{\beta}_1} = \frac{1}{n} \times \frac{\frac{1}{n-2}\sum_{i=1}^{n}(X_i - \overline{X})^2 \hat{u}_i^2}{\left[\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right]^2}.$$

2. Compute the t-statistic

$$t = \frac{\hat{\beta}_1 - \beta_{1,0}}{SE(\hat{\beta}_1)}.$$

3. Now, given a two sided alternative ($H_1 : \beta_1 \neq \beta_{1,0}$) we reject at the 5% level if $|t^{act}| > 1.96$ or, equivalently, if the p-value is less than 0.05.
   Recall the definition of the p-value:

$$\begin{aligned} p\text{-value} &= \Pr_{H_0}\left[\left|\frac{\hat{\beta}_1 - \beta_{1,0}}{SE(\hat{\beta}_1)}\right| > \left|\frac{\hat{\beta}_1^{act} - \beta_{1,0}}{SE(\hat{\beta}_1)}\right|\right] && (6.1)\\ &= \Pr_{H_0}(|t| > |t^{act}|) && (6.2)\\ &= 2 \cdot \Phi(-|t^{act}|) && (6.3) \end{aligned}$$

   The last equality holds due to the normal approximation for large samples.

Consider again the OLS regression stored in `linear_model` from Chapter 4 that gave us the regression line

$$\widehat{TestScore} = \underset{(9.47)}{698.9} - \underset{(0.49)}{2.28} \times STR \ , \ R^2 = 0.051 \ , \ SER = 18.6.$$

For testing a hypothesis about the slope parameter (the coefficient on STR), we need $SE(\hat{\beta}_1)$, the standard error of the respective point estimator. As common in the literature, standard errors are presented in parantheses below the point estimates.

As can be witnessed in Key Concept 5.1 it is rather cumbersome to compute the standard error and thus the t-statistic by hand. The question You should be asking Yourself right now is obvious: can we obtain these values with minimum effort using R? Yes, we can. Let us first use `summary()` to get a summary on the estimated coefficients in `linear_model`.

```
# print the summary of coefficients to the console
summary(linear_model)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 698.932949  9.4674911 73.824516 6.569846e-242
## STR          -2.279808  0.4798255 -4.751327  2.783308e-06
```

When looking at the second column of the coefficients' summary, we discover values for $SE(\hat{\beta}_0)$ and $SE(\hat{\beta}_1)$. Also, in the third column, named `t value`, we find t-statistics t^{act} suitable for tests of the individual hypotheses $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 0$. Furthermore, the output provides us with p-values corresponding to both tests against the two-sided alternatives $H_1 : \beta_0 \neq 0$ respectively $H_1 : \beta_1 \neq 0$ in the fourth column of the table.

Let us have a closer look at the test of

$$H_0 : \beta_1 = 0 \quad vs. \quad H_1 : \beta_1 \neq 0.$$

Using our revisited knowledge about t-statistics we find that

$$t^{act} = \frac{-2.279808 - 0}{0.4798255} \approx -4.75.$$

What does this tell us about the significance of the estimated coefficient? We reject the null hypothesis at the 5% level of significance since $|t^{act}| > 1.96$ that is the observed test statistic falls into the region of rejection. Or, alternatively and leading to the same result, we have p-value $= 2.78 * 10^{-6} < 0.05$. We conclude that the coefficient is significantly different from zero. With other words, our analysis provides evidence that the clase size *has an influence* on the students test scores. We say that β_1 is significantly different from 0 at the level of 5%.

Note that, although the difference is negligible in the present case as we will see later, `summary()` does not perform the normal approximation but calculates p-values using the appropriate t-distribution instead. Generally, the degrees of freedom are determined in the following manner:

$$\text{DF} = n - k - 1$$

where n is the number of observations used to estimate the model and k is the number of regressors, excluding the intercept. In our case, we have $n = 420$ observations and the only regressor is STR so $k = 1$. A sleek way to determine the model degress of freedom using R is

```
# determine degrees of freedom
linear_model$df.residual
```

```
## [1] 418
```

Hence, for the sampling distribution of $\hat{\beta}_1$ we have

$$\hat{\beta}_1 \sim t_{418}$$

such that the p-value for a two-sided significance test can be obtained by executing the following code:

```
2 * pt(-4.751327, df = 418)
```

```
## [1] 2.78331e-06
```

The result is very close to the value provided by `summary`. However since n is sufficiently large one could just as well use the standard normal density to compute the p-value:

```
2 * pnorm(-4.751327)
```

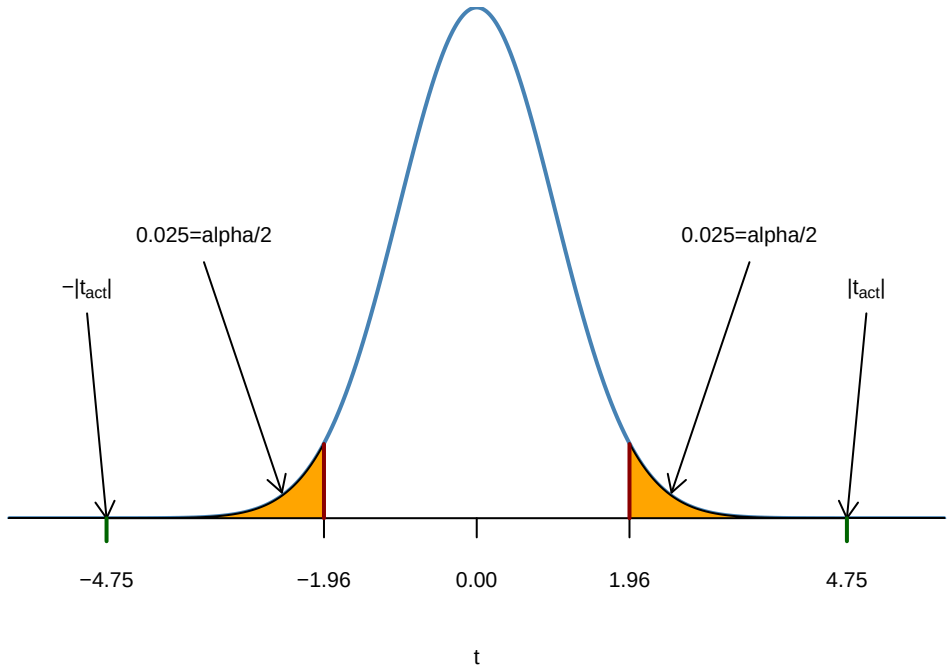
```
## [1] 2.02086e-06
```

The difference is indeed negligible. These findings tell us that, if $H_0 : \beta_1 = 0$ is true and we were to repeat the whole process of gathering observations and estimating the model, chances of observing a $\hat{\beta}_1 \geq |-4.75|$ are roughly $1 : 359285$ — so higher chances than winning the lottory next saturday but still very unlikely!

Using R we may visualise how such a statement is made when using the normal approximation. This reflects the principles depicted in figure 5.1 in the book. Do not let the following code chunk deter You: the code is somewhat longer than the usual examples and looks unappealing but there is **a lot** of repetition since color shadings and annotations are added on both tails of the normal distribution. We recommend You to execute the code step by step in order to see how the graph is augmented with the annotations.

```
# Plot the standard normal on the domain [-6,6]
t <- seq(-6,6,0.01)

plot(x = t,
     y = dnorm(t, 0, 1),
     type = "l",
```

```
     col = "steelblue",
     lwd = 2,
     yaxs = "i",
     axes = F,
     ylab = "",
     main = "Calculating the p-Value of a Two-Sided Test When t^act = -4.75",
     cex.lab = 0.7
     )

tact <- -4.75

axis(1, at = c(0,-1.96,1.96,-tact,tact), cex.axis=0.7)

# Shade the critical regions using polygon()

## critical region in left tail
polygon(x = c(-6, seq(-6,-1.96,0.01),-1.96),
        y = c(0, dnorm(seq(-6,-1.96,0.01)),0),
        col = 'orange'
        )

## critical region in right tail

polygon(x = c(1.96, seq(1.96, 6, 0.01), 6),
        y = c(0, dnorm(seq(1.96, 6, 0.01)), 0),
        col = 'orange'
        )

# Add arrows and texts indicating critical regions and the p-value
arrows(-3.5, 0.2, -2.5, 0.02, length = 0.1)
arrows(3.5, 0.2, 2.5, 0.02, length = 0.1)

arrows(-5, 0.16, -4.75, 0, length = 0.1)
arrows(5, 0.16, 4.75, 0, length = 0.1)

text(-3.5,0.22, labels = paste("0.025=",expression(alpha),"/2",sep = ""), cex = 0.7)
text(3.5,0.22, labels = paste("0.025=",expression(alpha),"/2",sep = ""), cex = 0.7)

text(-5,0.18, labels = expression(paste("-|",t[act],"|")), cex = 0.7)
text(5,0.18, labels = expression(paste("|",t[act],"|")), cex = 0.7)

# Add ticks indicating critical values at the 0.05-level, t^act and -t^act
rug(c(-1.96,1.96), ticksize  = 0.145, lwd = 2, col = "darkred")
rug(c(-tact,tact), ticksize  = -0.0451, lwd = 2, col = "darkgreen")
```



The p-Value is the area under the curve to left of -4.75 plus the area under the curve to the right of 4.75. As we already know from the calculations above, this value is very small.

6.2 Confidence Intervals for Regression Coefficients

As we already know, estimates of the regression coefficients β_0 and β_1 are afflicted with sampling uncertainty, see chapter 4. Therefore, we will *never* estimate the exact true value of these parameters from sample data in an empirical application. However, we may construct confidence intervals for the intercept and the slope parameter.

A 95% confidence interval for β_i has two equivalent definitions:

- The interval is the set of values for which a hypothesis test to the level of 5% cannot be rejected.
- The interval has a probability of 95% to contain the true value of β_i. So in 95% of all samples that could be drawn, the confidence interval will cover the true value of β_i.

We also say that the interval has a confidence level of 95%. The idea is summarized in Key Concept 5.3.

Key Concept 5.3

A Confidence Interval for β_i

Imagine You could draw all possible random samples of given size. The interval that contains the true value β_i in 95% of all samples is given by the expression

$$\text{KI}_{0.95}^{\beta_i} = \left[\hat{\beta}_i - 1.96 \times SE(\hat{\beta}_i)\,,\, \hat{\beta}_i + 1.96 \times SE(\hat{\beta}_i)\right].$$

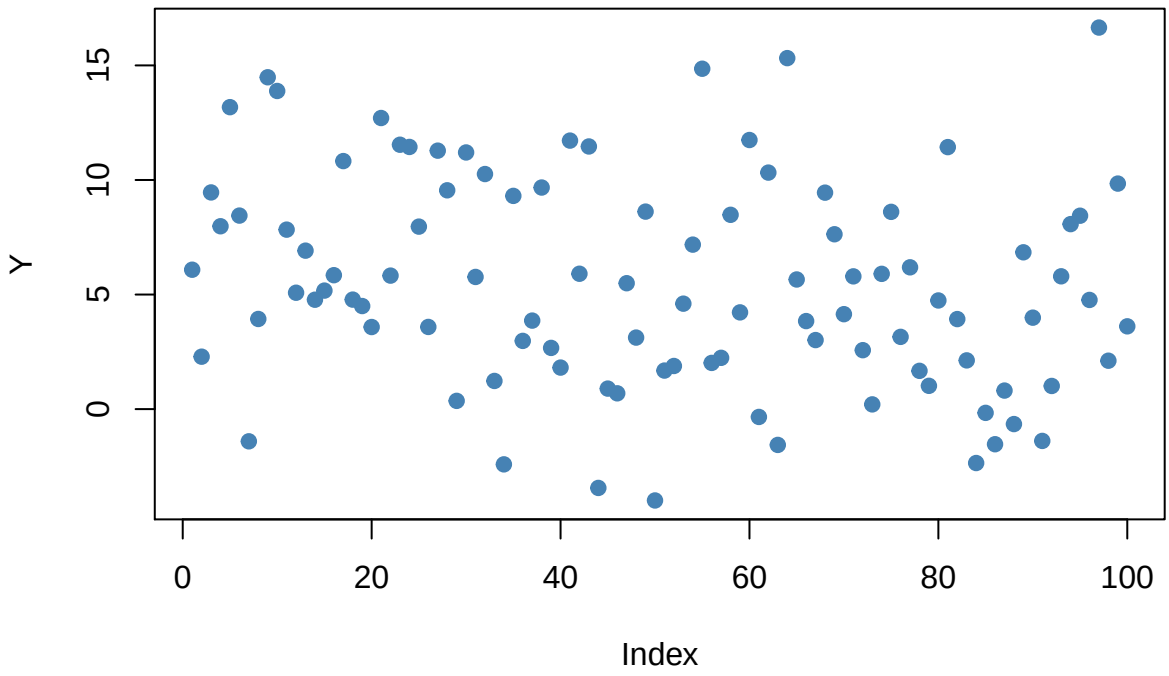
Equivalently, this interval can be seen as the set of null hypotheses for which a 5% two-sided hypothesis test does not reject.

R Simulation Study 5.1

To get a better understanding of confidence intervals we will conduct another simulation study. For now, assume that we are confronted with the following sample of $n = 100$ observations on a single variable Y where

$$Y_i \overset{i.i.d}{\sim} N(5, 25) \quad \forall\ i = 1, \ldots, 100.$$

```
# set random seed for reproducibility
set.seed(4)

# generate and plot the sample data
Y <- rnorm(n = 100,
           mean = 5,
           sd =5
           )

plot(Y,
     pch=19,
     col = "steelblue"
     )
```



We assume that the data is generated by the model

$$Y_i = \mu + \epsilon_i$$

where μ is the unknown constant and we know that $\epsilon_i \overset{i.i.d.}{\sim} N(0, 25)$. In this model, the OLS estimator for μ is given by

$$\hat{\mu} = \overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

(try to verify this!) i.e. the sample average of the Y_i. It further holds that

$$SE(\hat{\mu}) = \frac{\sigma_\epsilon}{\sqrt{n}} = \frac{5}{\sqrt{100}}.$$

A large sample 95% confidence interval for μ is then given by

$$KI^{\mu}_{0.95} = \left[\hat{\mu} - 1.96 \times \frac{5}{\sqrt{100}} \ , \ \hat{\mu} + 1.96 \times \frac{5}{\sqrt{100}}\right]. \tag{6.4}$$

It is fairly easy to compute this interval in R by hand. The following code chunck generates a named vector containing the interval bounds:

```
cbind(
  CIlower = mean(Y) - 1.96 * 5/10,
  CIupper = mean(Y) + 1.96 * 5/10
)
```

```
##       CIlower  CIupper
## [1,] 4.502625 6.462625
```

Nowing that $\mu = 5$ we see that our example covers the true value for the present sample. As opposed to real world examples, we can use R to get a better understanding of confidence intervals by repeatedly sampling data, estimating μ and computing the confidence interval for μ as in (6.4).

The procedure is as follows:

- We initialize the vectors `lower` and `upper` in which the simulated interval boundaries are to be saved. We want to simulate 10000 intervals so both vectors are set to have this length.
- We use a `for()` loop to sample 100 observations from the $N(5, 25)$ distribution and compute $\hat{\mu}$ as well as the boundaries of the confidence interval in every iteration of the loop.
- At last we join `lower` and `upper` in an array.

```
# set random seed
set.seed(1)

# initialize vectors of lower and upper interval boundaries
lower <- numeric(10000)
upper <- numeric(10000)

# loop sampling / estimation / CI
for(i in 1:10000) {
  Y <- rnorm(100, mean = 5, sd =5)
  lower[i] <- mean(Y) - 1.96 * 5/10
  upper[i] <- mean(Y) + 1.96 * 5/10
}

# join vectors of interval boundaries
CIs <- cbind(lower, upper)
```

According to Key Concept 5.3 we expect that the fraction of the 10000 simulated intervals saved in the array `CIs` that contain the true value $\mu = 5$ should be roughly 95%. We can check this using logical operators.

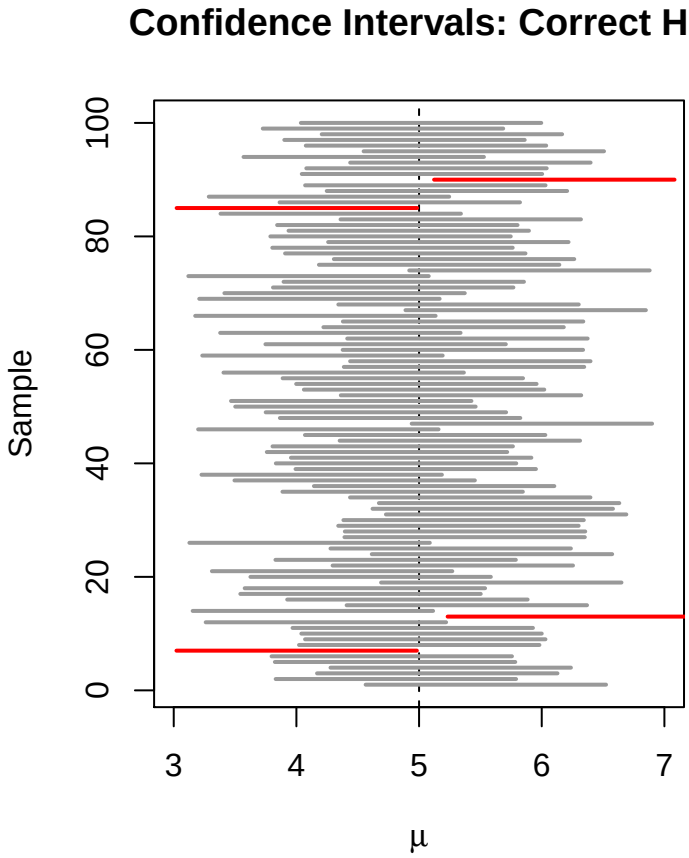
```
sum(CIs[,1] <= 5 & 5 <= CIs[,2])/10000
```

```
## [1] 0.9487
```

The simulation shows that the fraction of intervals covering $\mu = 5$, i.e. those intervals for which $H_0 : \mu = 5$ cannot be rejected is close to the theoretical value of 95%.

Let us draw a plot of the first 100 simulated confidence intervals and indicate those which *do not* cover the true value of μ. We do this by adding horizonal lines representing the confidence intervals on top of each other.

```
# identify intervals not covering mu
# (4 intervals out of 100)
ID <- which(!(CIs[1:100,1] <= 5 & 5 <= CIs[1:100,2]))

# initialize the plot
plot(0,
     xlim = c(3,7),
     ylim = c(1,100),
     ylab = "Sample",
     xlab = expression(mu),
     main = "Confidence Intervals: Correct H0")

# setup color vector
colors <- rep(gray(0.6), 100)
colors[ID] <- "red"

# draw reference line at mu=5
abline(v=5, lty=2)

# add horizontal bars representing the CIs
for(j in 1:100) {
  lines(c(CIs[j,1], CIs[j,2]),
        c(j,j),
        col = colors[j],
        lwd=2)
}
```



We find that for the first 100 samples, the true null hypthesis is rejected in four cases so these intervals do not cover $\mu = 5$. We have indicated the intervals which lead to a rejection of the true null hypothesis by red color.

Let us now turn back to the example of test scores and class sizes.The regression model from chapter 4 is stored in `linear_model`. An easy way to get 95% confidence intervals for β_0 and β_1, the coefficients on `(intercept)` and `STR`, is to use the function `confint()`. We only have to provide a fitted model object as the argument `object` to this function. The confidence level is set to 95% by default but can be modified by setting the argument `level`, see `?confint`.

```
confint(object = linear_model)
```

```
##                 2.5 %     97.5 %
## (Intercept) 680.32312 717.542775
## STR          -3.22298  -1.336636
```

Let us check if the calculation is done as we expect it to be. For β_1, that is the coefficient on `STR`, according to the formula presented above the interval borders are computed as

$$-2.279808 \pm 1.96 \times 0.4798255 \;\Rightarrow\; \text{KI}_{0.95}^{\beta_1} = [-3.22, -1.34]$$

so this actually leads to the same interval. Obviously, this interval *does not* contain the value zero what, as we have already seen in the previous section, leads to rejection of the null hypothesis $\beta_{1,0} = 0$.

6.3 Regression when X is a Binary Variable

Instead of using a continuous regressor X, we might be interested in running the regression

$$Y_i = \beta_0 + \beta_1 D_i + u_i \tag{5.2}$$

where D_i is binary variable, a so-called *dummy variable*. For example, we may define D_i in the following way:

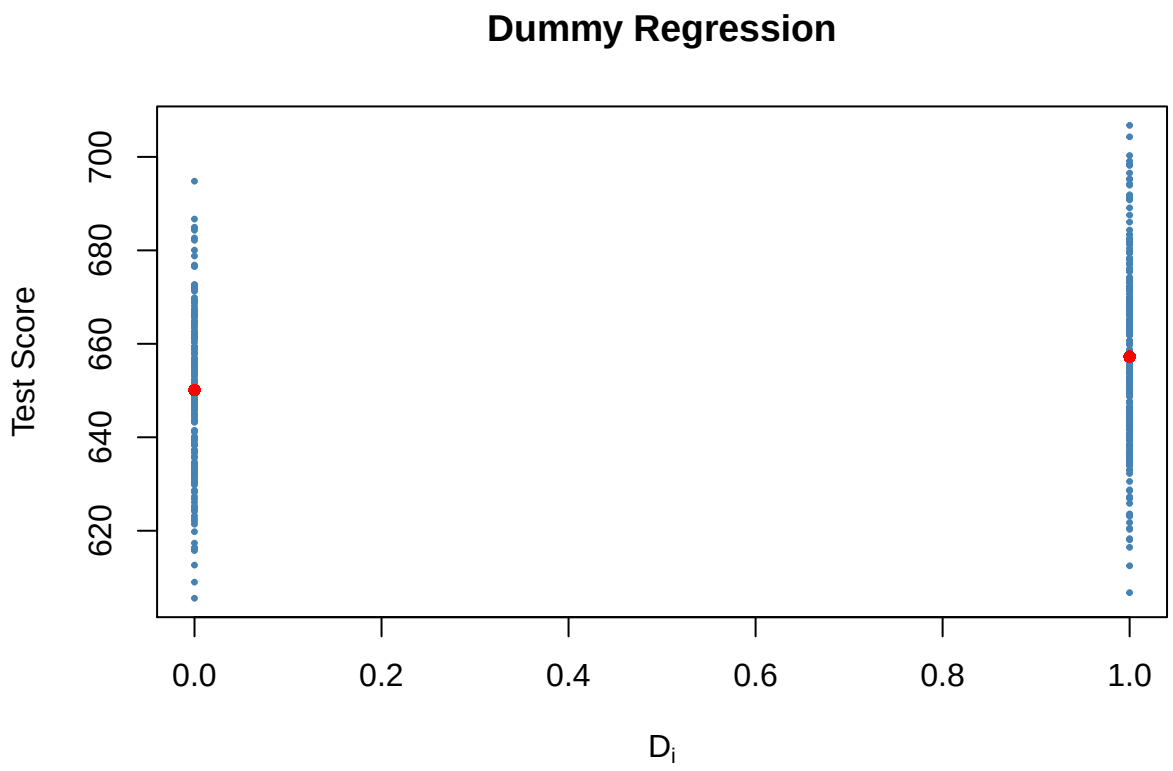
$$D_i = \begin{cases} 1 & \text{if } STR \text{ in } i^{th} \text{ school district} < 20 \\ 0 & \text{if } STR \text{ in } i^{th} \text{ school district} \geq 20 \end{cases} \tag{5.3}$$

The regression model now is

$$TestScore_i = \beta_0 + \beta_1 D_i + u_i. \tag{5.4}$$

Let us see how these data look like in a scatter plot:

```
# Create the dummy variable as defined above using a for loop
for (i in 1:nrow(CASchools)) {
  if (CASchools$STR[i] < 20) {
    CASchools$D[i] <- 1
    } else {
      CASchools$D[i] <- 0
    }
  }

# Plot the data
plot(CASchools$D, CASchools$score,          # provide the data to be ploted
     pch=20,                                # use filled circles as plot symbols
     cex=0.5,                               # set size of plot symbols to 0.5
     col="Steelblue",                       # set the symbols' color to "Steelblue"
     xlab=expression(D[i]),                 # Set title and axis names
     ylab="Test Score",
     main = "Dummy Regression"
     )
```



We see that with D as the regressor, it is not useful to think of β_1 as a slope parameter since $D_i \in \{0, 1\}$, i.e. we only observe two discrete values instead of a continuum of regressor values lying (in some range) on the real line. Simply put, there is no continuous line depicting the conditional expectation function $E(TestScore_i|D_i)$ since this function is solely defined for X-positions 0 and 1.

Therefore, the interpretation of the coefficients in our regression model is as follows:

- $E(Y_i|D_i = 0) = \beta_0$ so β_0 is the expected test score in districts where $D_i = 0$ i.e. where STR is below 20.
- $E(Y_i|D_i = 1) = \beta_0 + \beta_1$ or, using the result above, $\beta_1 = E(Y_i|D_i = 1) - E(Y_i|D_i = 0)$. Thus, β_1 is the difference in group specific expectations, i.e. the difference in expected test score between districts with $STR < 20$ and those with $STR \geq 20$.

We will now use R to estimate the dummy regression model as defined by equations (5.2) - (5.3) .

```
# estimate the dummy regression model
dummy_model <- lm(score ~ D, data = CASchools)
summary(dummy_model)
```

```
##
## Call:
## lm(formula = score ~ D, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.496 -14.029  -0.346  12.884  49.504
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  650.077      1.393 466.666  < 2e-16 ***
## D              7.169      1.847   3.882  0.00012 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.74 on 418 degrees of freedom
## Multiple R-squared:  0.0348, Adjusted R-squared:  0.0325
## F-statistic: 15.07 on 1 and 418 DF,  p-value: 0.0001202
```

One can see that the expected test score in districts with $STR < 20$ ($D_i = 1$) is predicted to be $650.1+7.17 = 657.27$ while districts with $STR \geq 20$ ($D_i = 0$) are expected to have an average test score of only 650.1.

Group specific predictions can be added to the plot by execution of the following code chunk:

```
# add group specific predictions to the plot
points(x = CASchools$D,
       y = predict(dummy_model),
       col = "red",
       pch = 20
       )
```

Here we use the function `predict()` to obtain estimates of the group specific means. The red dots represent these sample group averages. Accordingly, $\hat{\beta}_1 = 7.17$ can be seen as the difference in group averages.

By inspection of the output generated with `summary(dummy_model)` we may also find an answer to the question whether there is a statistically significant difference in group means. This in turn would support the hypothesis that students perform differently when they are taught in small classes rather than in large groups. We can assess this by a two-tailed test of the hypothesis $H_0 : \beta_1 = 0$. Conviniently the t-statistic and the corresponding p-value for this test are computed defaultly by `summary()`!

Since `t value` $= 3.88 > 1.96$ we reject the null hypothesis at the 5% level of significance. The same conclusion can be made when using the p-value which reports significance to the 0.00012% level.

As done with `linear_model`, we may alternatively use the `confint()` function to compute a 95% confidence interval for the true difference in means and see if the hypothesised value is an element of this confidence set.

```
# confidence intervals for coefficients in the dummy regression
confint(dummy_model)
```

```
##                 2.5 %    97.5 %
## (Intercept) 647.338594 652.81500
## D             3.539562  10.79931
```

We reject the hypothesis that there is no difference between group means at the 5% significance level since $\beta_{1,0} = 0$ lies outside of $[3.54, 10.8]$, the 95% confidence interval for the coefficient on D.

6.4 Heteroskedasticity and Homoskedasticity

All inference made in the previous chapters relies on the assumption that the error variance does not vary as regressor values change. But this will not necessarily be the case in most empirical applications.

Key Concept 5.4

Heteroskedasticity and Homoskedasticity

- We say that the error term of our regression model is homoskedastic if the variance of the conditional distribution of u_i given X_i, $Var(u_i|X_i = x)$, is constant *for all* observations in our sample

$$\text{Var}(u_i|X_i = x) = \sigma^2 \ \forall \ i = 1, \dots, n.$$

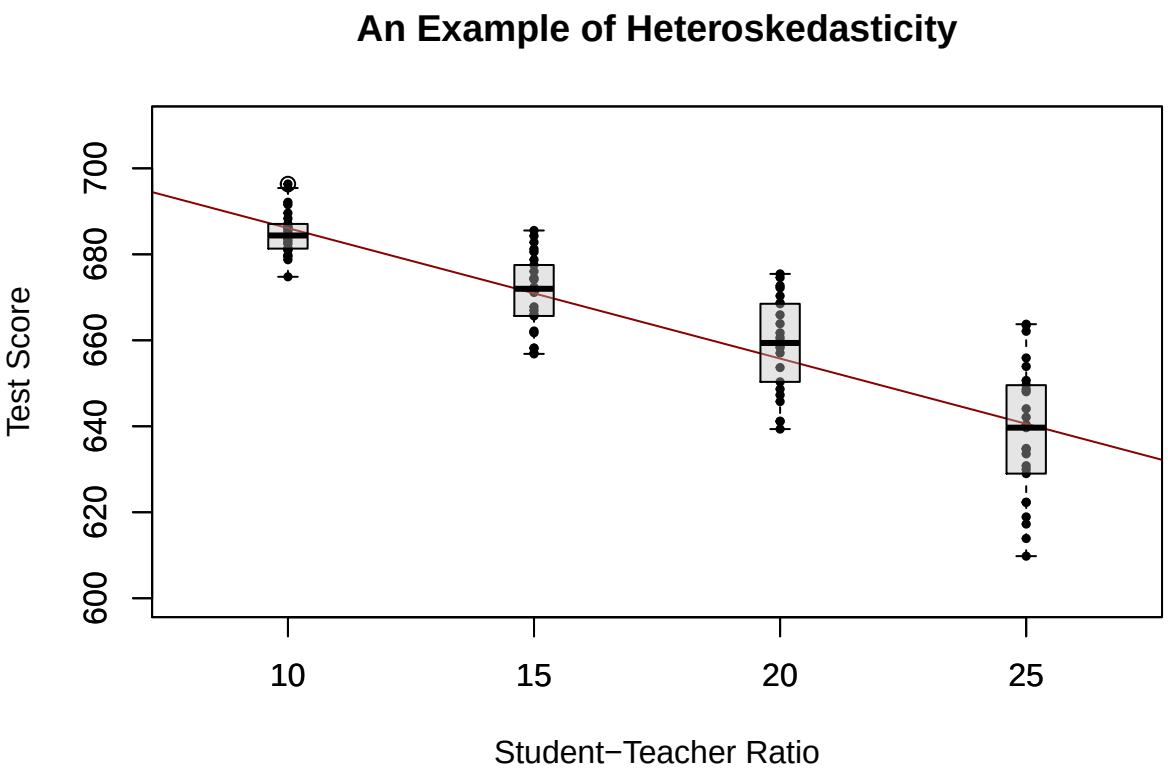
- If instead there is dependence of the conditional variance of u_i on X_i, the error term is said to be heteroskedastic. We then write

$$\text{Var}(u_i|X_i = x) = \sigma_i^2 \ \forall \ i = 1, \ldots, n.$$

- Homoskedasticity is a *special case* of heteroskedasticity.

For a better understanding of heteroskedasticity, we generate some bivariate heteroskedastic data, estimate a linear regression model and then use boxplots to depict the conditional distributions of the residuals.

```
# load scales package for custom color opacities
library(scales)

# Genrate some heteroskedastic data
set.seed(123)
x <- rep(c(10,15,20,25),each=25)
e <- rnorm(100, sd=12)
i <- order(runif(100, max=dnorm(e, sd=12)))
y <- 720 - 3.3 * x + e[rev(i)]

# Estimate the model
mod <- lm(y ~ x)

# Plot the data
plot(x=x,
     y=y,
     main="An Example of Heteroskedasticity",
     xlab = "Student-Teacher Ratio",
     ylab = "Test Score",
     cex = 0.5,
     pch = 19,
     xlim = c(8,27),
     ylim = c(600,710)
     )

# Add the regression line to the plot
abline(mod, col="darkred")

# Add boxplots to the plot
boxplot(y ~ x,
        add = TRUE,
        at = c(10,15,20,25),
        col = alpha("gray", 0.4),
        border = "black"
        )
```



For this artificial data it is straightforward to see that we face unequal conditional error variances. Specifically, we observe that the variance in test scores (and therefore the variance of the errors committed) *increases* with the student teacher ratio.

A Real-World Example for Heteroskedasticity

Think about the economic value of education: if there would not be an expected economic value-added to receiving education at university, You probably would not be reading this script right now. A starting point to empirical verification of such a relation exists is to have data on individuals that are in an employment relationship. More precisely, we need data on wages and education in order to work with a model like

$$wage_i = \beta_0 + \beta_1 \cdot education_i + u_i.$$

What can be presumed about this relation? It is likely that, on average, higher educated workers earn more money than workers with less education so we expect to estimate an upward sloping regression line. Also it seems plausible that workers with better education are more likely to meet the requirements for the well-paid jobs. However, workers with low education will have no shot at those well-paid jobs. Therefore it seems plausible that the distribution of earnings spreads out as education increases. In other words: we expect that there is heteroskedasticity!

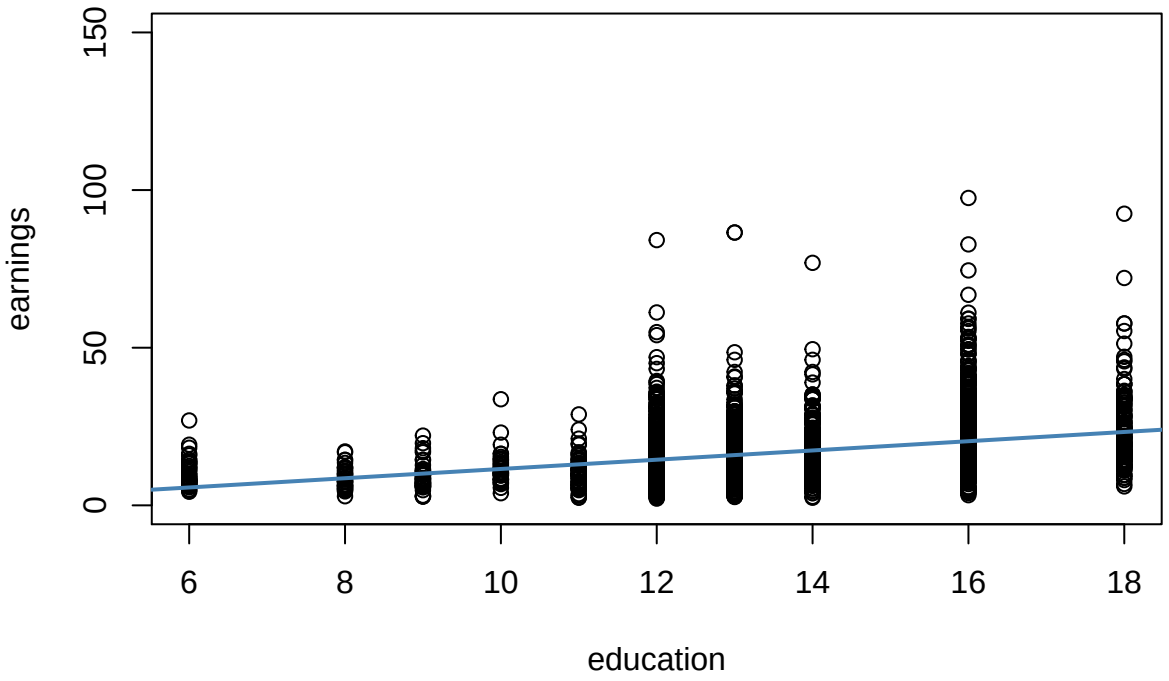
To verify this empirically we may use real data on hourly earnings and the number of years of education of employees. Such data can be found in `CPSSWEducation`. This data set is part of the package `AER` and stems from the Current Population Survey (CPS) which is conducted periodically by the Bureau of Labor Statistics in the US.

The subsequent code chunks demonstrate how to load the data into R and how to produce a plot in the fashion of figure 5.3 in the book.

```
# load package and attach data
library(AER)
data("CPSSWEducation")
attach(CPSSWEducation)

# get an overview
summary(CPSSWEducation)
```

```
##       age          gender        earnings        education
##  Min.   :29.0   female:1202   Min.   : 2.137   Min.   : 6.00
##  1st Qu.:29.0   male  :1748   1st Qu.:10.577   1st Qu.:12.00
##  Median :29.0                 Median :14.615   Median :13.00
##  Mean   :29.5                 Mean   :16.743   Mean   :13.55
##  3rd Qu.:30.0                 3rd Qu.:20.192   3rd Qu.:16.00
##  Max.   :30.0                 Max.   :97.500   Max.   :18.00
```

```
# estimate a simple regression model
labor_model <- lm(earnings ~ education)

# plot observations and add the regression line
plot(education,
     earnings,
     ylim = c(0,150)
     )

abline(labor_model, col="steelblue", lwd=2)
```



From inspecting the plot we can tell that the mean of the distribution of earnings increases with the level of education. This is also suggested by formal analysis: the estimated regression model stored in `labor_mod` asserts that there is a positive relation between years of education and earnings.

```
labor_model
```

```
##
```

```
## Call:
## lm(formula = earnings ~ education)
##
## Coefficients:
## (Intercept)    education
##      -3.134        1.467
```

The estimated regression equation states that, on average, an additional year of education increases a workers hourly earnings by about \$1.47. Once more we use `confint()` to obtain a 95% confidence interval for both regression coefficients.

```
confint(labor_model)
```

```
##                 2.5 %    97.5 %
## (Intercept) -5.015248 -1.253495
## education    1.330098  1.603753
```

Since the interval is $[1.33, 1.60]$ we can reject the hypothesis that the coefficient on `education` is zero at the 5% level.

What is more, the plot indicates that there is heteroskedasticity: if we assume the regression line to be a reasonably good representation of the conditional mean function $E(earnings_i|education_i)$, the dispersion of hourly earnings around that function cleary increases with the level of education, i.e. the variance of the distribution of earnings increases. In other words: the variance of the residuals increases with the years of education so that the regression errors are heteroskedastic. This example makes a case that it is doubtful to assume homoskedasticity in many economic applications.

Should We Care About Heteroskedasticity?

To answer this question, let us see how the variance of $\hat{\beta}_1$ is computed under the assumption of homoskedasticity. In this case we have

$$\sigma^2_{\hat{\beta}_1} = \frac{\sigma^2_u}{n \cdot \sigma^2_X} \tag{5.5}$$

which is a simplified version of the general equation (4.1) presented in Key Concept 4.4. See Appendix 5.1 of the book for details on the derivation. The `summary()` function in R estimates (5.5) by

$$\tilde{\sigma}^2_{\hat{\beta}_1} = \frac{SER^2}{\sum_{i=1}^n (X_i - \overline{X})^2} \quad \text{where} \quad SER = \frac{1}{n-2}\sum_{i=1}^n \hat{u}_i^2.$$

Thus `summary()` estimates the *homoskedasticity-only* standard error

$$\sqrt{\tilde{\sigma}^2_{\hat{\beta}_1}} = \sqrt{\frac{SER^2}{\sum (X_i - \overline{X})^2}}.$$

This in fact is an estimator for the standard deviation of the estimator $\hat{\beta}_1$ that is *inconsistent* for the true value $\sigma^2_{\hat{\beta}_1}$ when there is heteroskedasticity. The implication is that t-statistics computed in the manner of Key Concept 5.1 do not have a standard normal distribution, even in large samples. This issue may invalidate inference drawn when using the previously treated tools for hypothesis testing: we should be cautious when making statements about the significance of regression coefficients on the basis of t-statistics as computed by `summary()` or confidence intervals produced by `confint()` if it is doubtful for the assumption of homoskedasticity to hold!

We will now use R to compute the homoskedasticity-only standard error estimate for $\hat{\beta}_1$ in the test score regression model `linear_model` by hand and see if it matches the value produced by `summary()`.

```
# Store model summary in 'mod'
model <- summary(linear_model)

# Extract the standard error of the regression from model summary
SER <- model$sigma

# Compute the variation in 'size'
V <- (nrow(CASchools)-1) * var(CASchools$STR)

# Compute the standard error of the slope parameter's estimator and print it
SE.beta_1.hat <- sqrt(SER^2/V)
SE.beta_1.hat
```

```
## [1] 0.4798255
```

```
# Use logical operators to see if the value computed by hand matches the one provided # in mod$coeffici

round(model$coefficients[2,2], 4) == round(SE.beta_1.hat, 4)
```

```
## [1] TRUE
```

Indeed, the estimated values are equal.

Computation of Heteroskedasticity-Robust Standard Errors

Cosistent estimation of $\sigma_{\hat{\beta}_1}$ under heteroskedasticity is granted when the following *robust* estimator is used.

$$SE(\hat{\beta}_1) = \sqrt{\frac{\frac{1}{n-2}\sum_{i=1}^{n}(X_i - \overline{X})^2\hat{u}_i^2}{\left[\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right]^2}} \tag{5.6}$$

Standard error estimates computed this way are also referred to as Eicker-Huber-White standard errors. It can be quite cumbersome to do this calculation by hand. Luckily, there are R function for that purpose. A convenient one, named `vcovHC()` is part of the `sandwich` package. This function can compute a variety of standard error estimators. The one brought forward in (5.6) is computed when the argument `type` is set to `"HC0"`.

Let us now compute robust standard error estimates for the coefficients in `linear_model`.

```
# load the sandwich package
library(sandwich)

# compute robust standard error estimates
vcov <- vcovHC(linear_model, type = "HC0")
vcov
```

```
##             (Intercept)        STR
## (Intercept)  106.908469 -5.3383689
## STR           -5.338369  0.2685841
```

The output of `vcovHC()` is the variance-covariance matrix of coefficient estimates. We are interested in the square root of the diagonal elements of this matrix since these values are the standard error estimates we seek.

When we have k > 1 regressors, writing down the equations for a regression model becomes very messy. A more convinient way to denote and estimate so-called multiple regression models is matrix algebra. This is why functions like vcovHC() produce matrices. In the simple linear regression model, the variances and covariances of the coefficient estimators can be gathered in the variance-covariance matrix

$$\text{Var}\begin{pmatrix}\hat{\beta}_0 \\ \hat{\beta}_1\end{pmatrix} = \begin{pmatrix}\text{Var}(\hat{\beta}_0) & \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \\ \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) & \text{Var}(\hat{\beta}_1)\end{pmatrix} \tag{6.5}$$

which is a symmetric matrix. So vcovHC() gives us $\widehat{\text{Var}}(\hat{\beta}_0)$, $\widehat{\text{Var}}(\hat{\beta}_1)$ and $\widehat{\text{Cov}}(\hat{\beta}_0, \hat{\beta}_1)$ but most of the time we are interested in the diagonal elements of the estimated matrix.

```
# compute the square root of the diagonal elements in vcov
robust_se <- sqrt(diag(vcov))
robust_se
```

```
## (Intercept)         STR
##   10.339655    0.518251
```

Now assume we want to generate a coefficient summary as provided by `summary()` but with *robust* standard error estimates for the coefficient estimators, robust t-statistics and corresponding p-values for the regression model `linear_model`. This can be done using `coeftest()` from the package `lmtest`, see `?coeftest`. Further we specify in the argument `vcov.` that `vcov`, the Eicker-Huber-White estimate of the variance matrix we have computed before should be used.

```
# We invoke the function `coeftest()` on our model
coeftest(linear_model, vcov. = vcov)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 698.93295   10.33966  67.597 < 2.2e-16 ***
## STR          -2.27981    0.51825  -4.399 1.382e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that values reported in the column `Std. Error` equal the ones received using `sqrt(diag(vcov))`.

How severe are the implications of using homoskedasticity-only standard errors in the presence of heteroskedasticity? The answer is: it depends. As mentioned above we may face the risk of drawing wrong conclusions when conducting significance tests. Let us illustrate this by generating another example of a heteroskedastic data set and use it to estimate a simple regression model. We take

$$Y_i = \beta_1 \cdot X_i + u_i \quad , \quad u_i \overset{i.i.d.}{\sim} N(0, 0.36 \cdot X_i^2)$$

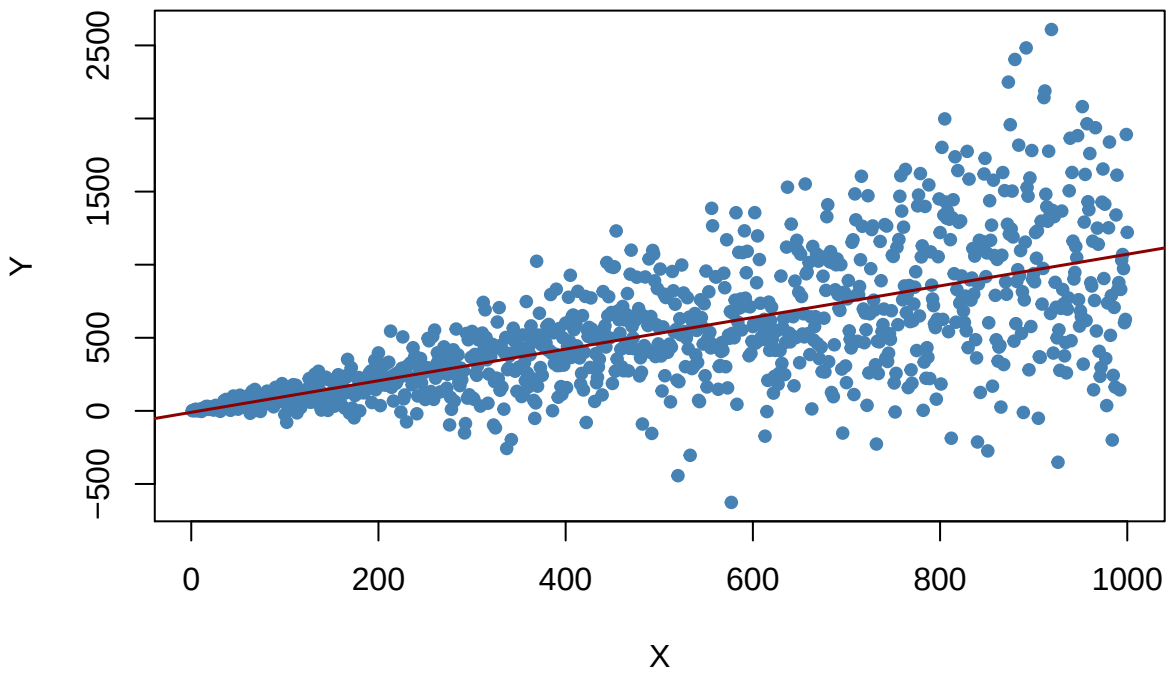
with $\beta_1 = 1$ as the data generating process. The assumption of homoskedasticity is violated since the variance of the errors is a non-linear increasing function of X_i but the errors have zero mean and are i.i.d. such that the assumptions made in Key Concept 4.3 are not violated. As before, the true conditional mean function we are interested in estimating is

$$E(Y_i|X_i) = \beta_1 X_i.$$

```
# set random seed
set.seed(21)

# generate heteroskedastic data
X <- 1:1000
Y <- rnorm(n = 1000, mean = X, sd = 0.6*X)

# estimate a simple regression model
reg <- lm(Y ~ X)
```

We plot the data and add the regression line.

```
# plot the data
plot(X, Y, pch = 19, col="steelblue", cex = 0.8)

# add the regression line to the plot
abline(reg, col = "darkred", lwd = 1.5)
```



The plot clearly shows that the data are heteroskedastic as the variance of Y grows with X. We continue by conducting a significance test of the (true) null hypothesis $H_0 : \beta_1 = 1$ twice, once using the homoskedasticity-only standard error formula and once with the robust version (5.6). An idiomatic way to do this in R is the function `linearHypothesis()` from the package `car`, see `?linearHypothesis`. It allows to test linear hypotheses about parameters in linear models in a similar way as done with a t-statistic and offers various robust covariance matrix estimators. We test by comparing the tests' p-values to the significance level of 5%.

linearHypothesis() computes a test statistic that follows an F distribution under the null hypothesis. We will not loose too much words on the theory behind it at this time. In general, the core idea of the F test is to compare the fit of different models. When testing a hypothesis about a *single* coefficient using a F test, one can show that the test statistic is simply the square of the corresponding t-statistic:

$$F = t^2 = \frac{\hat{\beta}_i - \beta_{i,0}}{SE(\hat{\beta}_i)} \sim F_{1,n-k-1}$$

In linearHypothesis(), the hypothesis must be provided as a *string*. The function returns an object of class anova which contains further information on the test that can be accessed using the $ operator. For example, we can obtain the test's p-value by adding $'Pr(>F)' right behind the function call.

```
# test using default standard error
linearHypothesis(reg, hypothesis.matrix = "X = 1")$'Pr(>F)'[2] < 0.05
```

```
## [1] TRUE
```

```
# test using robust standard error
linearHypothesis(reg, hypothesis.matrix = "X = 1", white.adjust = "hc0")$'Pr(>F)'[2] < 0.05
```

```
## [1] FALSE
```

This is a good example of what can go wrong if we do not care for heteroskedasticity: for the data set at hand the default method rejects the null hypothesis $\beta_1 = 1$ although it is true. Using the robust standard error though the test does not reject the null. Of course we could argue that this is just a coincidence and both tests are equally well in maintaining the type I error rate of 5%. This can be further investigated by computing Monte Carlo estimates of the rejection frequencies of both tests on the basis of a large number of random samples. We proceed as follows:

- initialize vectors `t` and `t.rob` as type `numeric` with length 10000.
- Using a `for()` loop, we generate 10000 heteroskedastic random samples of size 1000, estimate the regression model and check whether the tests wrongly reject the null at the level of 5% using comparison operators. The results are stored in the respective vectors `t` and `t.rob`.
- After the simulation, we compute the fraction of rejections for both tests.

```
# initialize vectors t and t.rob
t <- numeric(10000)
t.rob <- numeric(10000)

# loop sampling and estimation
for (i in 1:10000) {

  # sample data
  X <- 1:1000
  Y <- rnorm(n = 1000, mean = X, sd = 0.6*X)

  # estimate regression model
  reg <- lm(Y ~ X)

  # homoskedasdicity-only significance test
  t[i] <- linearHypothesis(reg, "X = 1")$'Pr(>F)'[2] < 0.05

  # robust significance test
  t.rob[i] <- linearHypothesis(reg, "X = 1", white.adjust = "hc0")$'Pr(>F)'[2] < 0.05
```

```
}

# compute fraction of rejections
cbind(t = sum(t), t.rob = sum(t.rob)) / 10000
```

```
##           t  t.rob
## [1,] 0.0762 0.0524
```

The results show that we face an increased risk of falsely rejecting the null using the homoskedasticity-only standard error for the testing problem at hand: with the common standard error estimator, 7.62% of all tests reject the null hypothesis falsely. In contrast, with the robust test statistic we are close to the nominal level of 5%.

6.5 The Gauss-Markov Theorem

When estimating regression models, we know that the results of the estimation procedure are outcomes of a random process. However, when using unbiased estimators, at least on average, we estimate the true parameter. When comparing different unbiased estimators, it is therefore interesting to know which one has the highest precision: being aware that the likelihood of estimating the *exact* value of the parameter of interest is 0 in an empirical application, we want to make sure that the likelihood of obtaining an estimate very close to the true value is as high as possible. We want to use the estimator with the lowest variance of all unbiased estimators. The Gauss-Markov theorem states that the OLS estimator has this property under certain conditions.

Key Concept 5.5

The Gauss-Markov Theorem for $\hat{\beta}_1$

Suppose that the assumptions made in Key Concept 4.3 hold *and* that the errors are *homoskedastic*. The OLS estimator is the best (in the sense of smallest variance) linear conditionally unbiased estimator (BLUE) in this setting.

Let us have a closer look at what this means:

- Estimators of β_1 that are linear functions of the $Y_1, \dots, Y_n$ and that are unbiased conditionally on the regressor $X_1, \dots, X_n$ can be written as
$$\widetilde{\beta}_1 = \sum_{i=1}^{n} a_i Y_i$$
where the a_i are weights that are allowed to depend on the X_i but *not* on the Y_i.
- We already know that $\widetilde{\beta}_1$ has a sampling distribution: $\widetilde{\beta}_1$ is a linear function of the Y_i which are random variables. If now
$$E(\widetilde{\beta}_1 | X_1, \dots, X_n) = \beta_1$$
we say that $\widetilde{\beta}_1$ is a linear unbiased estimator of β_1, conditionally on the $X_1, \dots, X_n$.
- We may ask if $\widetilde{\beta}_1$ is also the *best* estimator in this class, i.e. the most efficient one of all linear unbiased estimators where "most efficient" means smallest variance. The weights a_i play an important role here and it turns out that OLS uses just the right weights to have the BLUE property.

R Simulation Study: BLUE Estimator

Consider the case of a regression of $Y_i, \dots, Y_n$ only on a constant. Here, the Y_i are assumed to be a random sample from a population with mean μ and variance σ^2. We know that the OLS estimator in this model is

simply the sample mean:

$$\hat{\beta}_1 = \overline{\beta}_1 = \sum_{i=1}^{n} \underbrace{\frac{1}{n}}_{=a_i} Y_i \tag{6.6}$$

Clearly, each observation is weighted by

$$a_i = \frac{1}{n}.$$

and We also know that $\text{Var}(\hat{\beta}_1) = \text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{n}$.

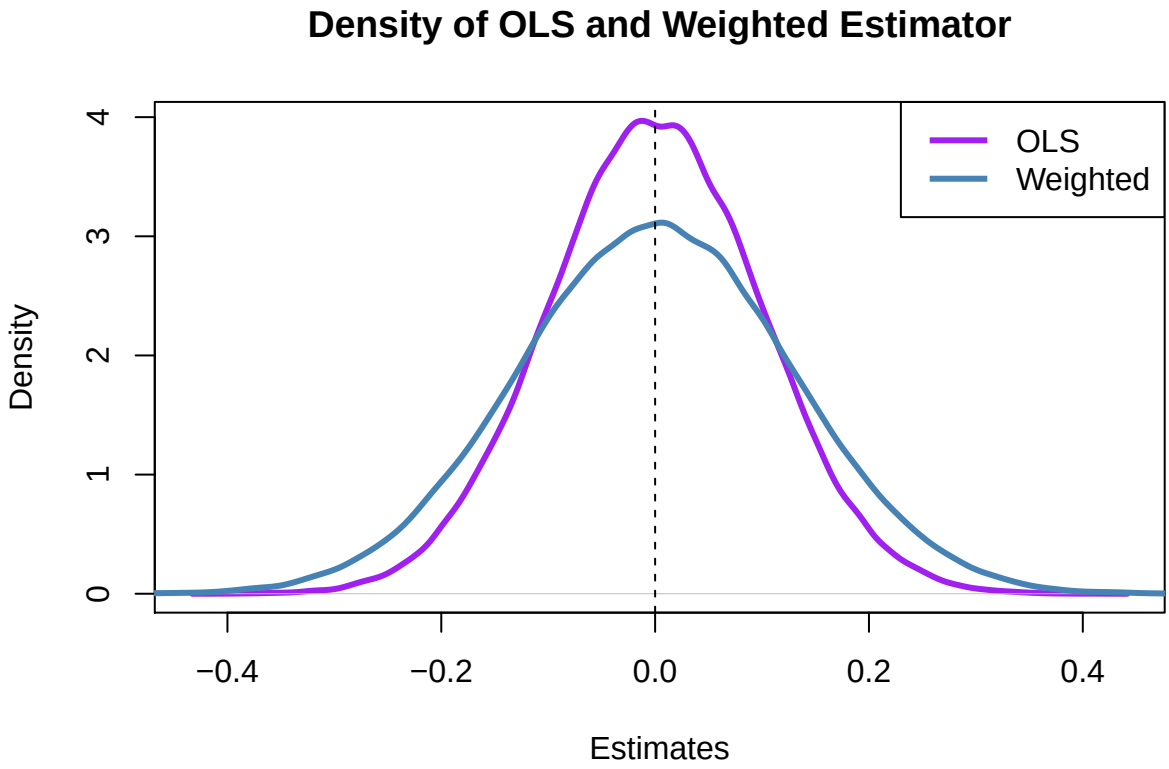
We will now use R for a simulation study that illustrates what happens to the variance of (6.6) if different weights

$$w_i = \frac{1 \pm \epsilon}{n}$$

are assigned to either half of the sample $Y_1, \ldots, Y_n$ instead of using $\frac{1}{n}$, the weights implied by OLS.

```
# Set sample size and number of repitititons
n <- 100
reps <- 1e5

# Choose epsilon and create a vector of weights as defined above
epsilon <- 0.8
w <- c(rep((1+epsilon)/n,n/2),
       rep((1-epsilon)/n,n/2)
      )

# Draw a random sample y_1,...,y_n from the standard normal distribution,
# use both estimators 1e5 times and store the result in thr vectors ols and
# weightedestimator

ols <- rep(NA, reps)
weightedestimator <- rep(NA, reps)

for (i in 1:reps) {
  y <- rnorm(n)
  ols[i] <- mean(y)
  weightedestimator[i] <- crossprod(w,y)
}

# Plot kernel density estimates of the estimators' distributions

# OLS
plot(density(ols),
     col = "purple",
     lwd = 3,
     main = "Density of OLS and Weighted Estimator",
     xlab = "Estimates"
     )

# Weighted
lines(density(weightedestimator),
```

```
      col = "steelblue",
      lwd = 3
      )

# Add a dashed line at 0 and a legend to the plot
abline(v = 0, lty = 2)
legend('topright',
       c("OLS","Weighted"),
       col = c("purple","steelblue"),
       lwd = 3
       )
```



What conclusion can we draw from the result?

- Both estimators seem to be unbiased: the means of their estimated distributions are zero.
- The `weightedestimator` is less efficient than the `ols` estimator: there is higher dispersion when weights are $w_i = \frac{1 \pm 0.8}{100}$ instead of $w_i = \frac{1}{100}$ as required by the OLS solution.

Hence, our simulation results confirm what is stated by the Gauss-Markov Theorem.

6.6 Using the t-Statistic in Regression When the Sample Size Is Small

The three OLS assumptions discussed in chapter 4 (see Key Concept 4.3) are the foundation results on the large sample distribution of the OLS estimators in the simple regression model. What can be said about the distribution of the estimators and their t-statistics when the sample size is small and the population distribution of the data is unkown? Provided that the three least squares assumptions hold and the errors are normally distributed and homoskedastic (we refer to these conditions as the homoskedastic normal regression

assumptions), we have normally distributed estimators and t-distributed test tatistics. Recall the definition of a t-distributed variable

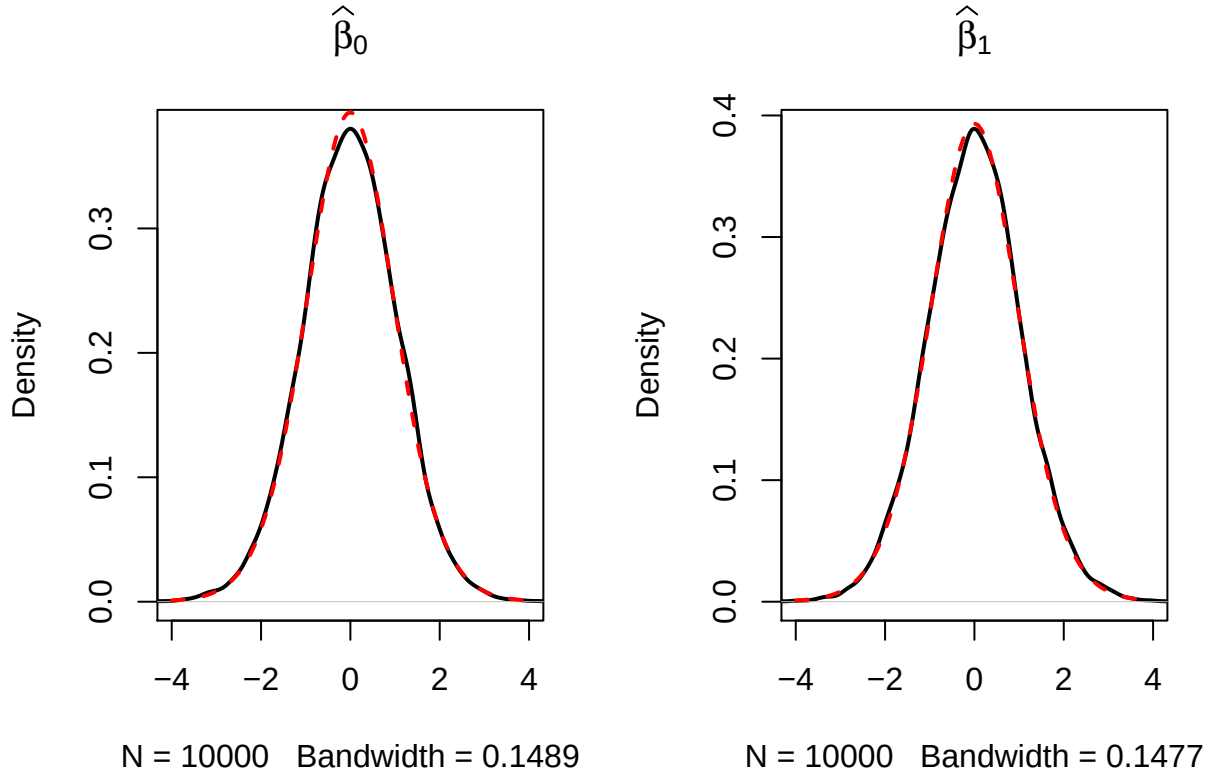
$$\frac{Z}{\sqrt{W/m}} \sim t_m$$

where Z is a standard normal random variable, W is χ^2 distributed with m degrees of freedom and Z and W are independent. See section 5.6 in the book for a more detailed discussion of the small sample distribution of t-statistics in regression.

Let us simulate the distribution of regression t-statistics based on a large number of small random samples ($n = 20$) and compare their simulated distributions to their theoretical distribution which sould be t_{18}, the t-distribution with 18 degrees of freedom (recall that DF $= n - k - 1$).

```
# initialize vectors
beta_0 <- numeric(10000)
beta_1 <- numeric(10000)

# loop sampling / estimation / t statistics
for (i in 1:10000) {

  X <- runif(20,0,20)
  Y <- rnorm(n = 20, mean = X)
  reg <- summary(lm(Y ~ X))
  beta_0[i] <- (reg$coefficients[1,1] - 0)/(reg$coefficients[1,2])
  beta_1[i] <- (reg$coefficients[2,1] - 1)/(reg$coefficients[2,2])

}

# plot distributions and compare with t_18 density function
par(mfrow = c(1,2))

# plot simulated density of beta_0
plot(density(beta_0),
     lwd= 2 ,
     main = expression(widehat(beta)[0]),
     xlim = c(-4, 4)
    )

# add t_18 density to the plot
curve(dt(x, df = 18),
      add = T,
      col = "red",
      lwd = 2,
      lty = 2
      )

# plot simulated density of beta_1
plot(density(beta_1),
     lwd = 2,
     main = expression(widehat(beta)[1]), xlim=c(-4,4)
     )

# add t_18 density to the plot
curve(dt(x, df = 18),
```

```
      add = T,
      col = "red",
      lwd = 2,
      lty = 2
      )
```



The outcomes are consistent with our expectations: the empirical distributions of both estimators seem to track the t_{18} distribution quite closely.

6.7 Exercises

1. Testing Two Null Hypotheses Individually

Consider the estimated regression line

$$\widehat{TestScore} = \underset{(23.96)}{567.43} - \underset{(0.85)}{7.15} \times STR, \; R^2 = 0.8976, \; SER = 15.19$$

with standard errors in parentheses.

Instructions:

- Compute the p-value for a t-test of the hypothesis that the intercept is zero. Save the result to p_int
- Compute the p-value for a t-test of the hypothesis that the coefficient of STR is zero. Save the result to p_STR

Hint: both hypotheses can be tested individually using a two-sided test. Use pnorm() to obtain cumulated probabilities for standard normally distributed outcomes, see ?pnorm

Compute the p-value for the first significance test and save it to p_int

Compute the p-value for the second significance test and save it to p_cs

t_int <- 567.43/23.9606 p_int <- 2*(1-pnorm(abs(t_int)))* t_STR <- *7.15/0.8536 p_STR <- 2*(1-pnorm(abs(t_STR))) test_object("p_int") test_object("p_STR")

2. Two Null Hypotheses You Can't Reject, Can You?

$$\widehat{TestScore} = \underset{(23.96)}{567.43} - \underset{(0.85)}{7.15} \times STR,\ R^2 = 0.8976,\ SER = 15.19$$

Can you reject the null hypotheses discussed in the previous code exercise using individual t-tests at the 5% significance level?

The variables t_int and t_STR are the t-statistics. Both are available in your working environment.

Instructions

- Gather t_int and t_STR in a vector test and use logical operators to check whether the corresponding rejection rule applies.

Hints:

- Both tests are two-sided t-tests. Key Concept 5.2 recaps how a two-sided t-test is conducted.
- Use qnorm() to obtain normal critical values.

t_int <- 567.43/23.9606 t_STR <- 7.15/0.8536 # check wether the t-tests reject

test <- c(t_int, t_STR) # entry is 'TRUE' if a test rejects abs(test) >= qnorm(0.95)

test_or({ test <- c(t_int,t_STR) },{ test <- c(t_STR, t_int) }) test_or({ test_student_typed("abs(test) >= qnorm(0.95)") },{ test_student_typed("abs(test) <= qnorm(0.95)") })

3. Confidence Intervals

mod, an object of class lm which contains the estimated model

$$\widehat{TestScore} = \underset{(23.96)}{567.43} - \underset{(0.85)}{7.15} \times STR,\ R^2 = 0.8976,\ SER = 15.19$$

is available in your working environment.

Instructions

- Compute 90% confidence intervals for both coefficients.

Hints:

Use the function confint(), see ?confint. The argument level sets the confidence level to be used.

STR <- c(23, 19, 30, 22, 23, 29, 35, 36, 33, 25) TS <- c(430, 430, 333, 410, 390, 377, 325, 310, 328, 375) mod <- lm(TS ~ STR) # compute 90% confidence intervals for the model coefficients

confint(mod, level = 0.9)

test_function("confint", args = c("object","level"))

4. A Confidence Interval for the Mean I

Consider the model

$$Y_i = \beta_1 + u_i$$

where $Y_i \sim \mathcal{N}(\mu, \sigma^2)$. Following the discussion preceding equation (6.4), a 95% confidence interval for the mean of the Y_i can be computed as

$$KI^{\mu}_{0.95} = \left[\hat{\mu} - 1.96 \times \frac{\sigma}{\sqrt{n}} \ , \ \hat{\mu} + 1.96 \times \frac{\sigma}{\sqrt{n}}\right].$$

Instructions

- Sample $n = 100$ observations from a normal distribution with variance 100 and mean 10.
- Use the sample to estimate β_1. Save the estimate in mu_hat.
- Assume that $\sigma^2 = 100$ is known. Replace the zeros in the code below to obtain a 95% confidence interval for the mean of the Y_i.

Hints:

Use the function confint(), see ?confint. The argument level sets the confidence level to be used.

random seed for reproducibility set.seed(1)

sample the observations

estimate the mean, assign the estimate to 'mu_hat'

compute the 95% confidence interval using the formula above CI <- c("lower" = 0, "upper" = 0)

set.seed(1) mu_hat <- mean(rnorm(100, mean = 10, sd = 10)) CI <- c("lower" = mu_hat - 1.96, "upper" = mu_hat + 1.96) test_object("CI") test_object("mu_hat") test_object("CI")

5. A Confidence Interval for the Mean II

For historical reasons, some R functions that we use to obtain statistical inference on model parameters, among them confint() and summary(), rely on the t-distribution instead of using the large-sample normal approximation. This is why for small sample sizes (and hence small degrees of freedom), p-values and confidence intervals reported by these functions deviate from those computed using critical values or cumulative probabilities of the standard normal distribution.

The 95% confidence interval for the mean in the previous exercise is $[9.1289, 13.0489]$.

100 observations sampled from a normal distribution with $\mu = 10$ and $\sigma^2 = 100$ have been assigned to the vector s and are available in your workspace.

Instructions

- Set up a suitable regression model to estimate the mean of the observations in s. Then use confint() to compute a 95% confidence interval for the mean.

(Check that the result is different from the interval reported above.)

set.seed(1); s <- rnorm(100, mean = 10, sd = 10) # use a regression to obtain an estimate of the mean

compute the 95% CI

set.seed(1) confint(lm(s ~ 1)) test_predefined_objects("s") test_output_contains("confint(lm(s ~ 1))")

6. Regression on a Dummy I

Chapter 5.3 discusses regression when X is a dummy variables where we have used a for() loop to generate a binary variable indicating whether a schooling district in the CASchools data set has a student-teacher ration below 20. Though it is instructive to use a loop for this, there are ways to achieve the same with less code.

A dataframe DF with 100 observations of a variable X is available in your working environment.

Instructions

- Use ifelse() to generate a binary vector dummy indicating whether the observations in X are positive.
- Append dummy to the dataframe DF.

set.seed(1) DF <- data.frame("X" = rnorm(100)) # generate the dummy vector 'dummy' using 'ifelse()'

append 'dummy' to 'DF'

dummy <- ifelse(DF$X > 0, 1, 0)$DF$dummy <- dummy test_object("dummy") test_function("ifelse") test_object("DF")

7. Regression on a Dummy II

A dataframe DF with 100 observations on Y and the binary variable D from the previous exercise are available in your working environment.

Instructions

- Compute the group-specific sample means of the observations in Y. Save the mean of observations in Y where dummy == 1 to mu_Y_D1 and assign the mean of those observations with D == 0 to mu_Y_D0.
- Use lm() to regress Y on D i.e. estimate the coefficients in the model

$$\widehat{Y}_i = \beta_0 + \beta_1 \times D_i + u_i.$$

Check that the estimates of the coefficients β_0 and β_1 reflect specific sample means. Can you tell which?

set.seed(1) DF <- data.frame("Y" = rnorm(100)) DF$D <- -ifelse(DF$Y > 0, 1, 0) # compute group-specific sample means of 'Y'

Regress 'Y' on 'D'

mu_Y_D1 <- mean(DF$Y[DF$D ==1]) mu_Y_D0 <- mean(DF$Y[DF$D ==0]) lm(Y ~ dummy, data = DF) test_object("mu_Y_D1") test_object("mu_Y_D1") test_or({ test_function("lm", args = "formula") },{ ex() %>% override_solution("lm(DF$Y ~ DF$dummy)") %>% check_function("lm") %>% check_arg("formula") })

8. Regression on a Dummy III

In this exercise, you have to visualize some of the results from the dummy regression model

$$\widehat{Y}_i = -0.66 + 1.43 \times D_i$$

estimated in the previous exercise.

A dataframe DF with 100 observations on X and the binary variable dummy as well as the model object dummy_mod from the previous exercise are available in your working environment.

Instructions

- Start by drawing a visually appealing plot of the observations on Y and D based on the code chunk provided in Script.R. Replace the ??? by the correct expressions!
- Add the regression line to the plot.

set.seed(1) DF <- data.frame("Y" = rnorm(100)) DF$D <- -ifelse(DF$Y > 0, 1, 0) dummy_mod <- lm(Y ~ D, data = DF) # Replace the '???' by the correct values plot(x = ??? , y = ???, pch = 20, cex = 1, col = 'Steelblue', xlab = expression(D[i]), ylab = 'Test Score', main = 'Dummy Regression')

add the regression line

plot(x = DF$D, y = DF$Y, pch = 20, cex = 1, col = 'Steelblue', xlab = expression(D[i]), ylab = 'Test Score', main = 'Dummy Regression')
abline(dummy_mod) test_predefined_objects(c("DF","dummy_mod")) test_function("abline") test_function("plot", args = c("x","y")) success_msg("Nice! Clearly, a line is not a suitable way to think of this model!")

9. Gender Wage Gap I

CPS1985 is a cross-section data set originating from the May 1985 *Current Population Survey* by the *US Census Bureau* and, among others things, containes observations on wage and gender of employees.

CPS1985 is part of the package AER.

Instructions

- Load the package AER and attach the data set CPS1985.
- Estimate the dummy regression model

$$wage_i = \beta_0 + \beta_1 \cdot female_i + u_i$$

where

$$female = \begin{cases} 1, & \text{if employee is female} \\ 0, & \text{if employee is male.} \end{cases}$$

Save the result in wage_mod.

Load the package and attach the data set

Perform the regression

library(AER) data(CPS1985) dummy_mod <- lm(wage ~ gender, data = CPS1985) test_function("library") test_function("data") test_or({ ex() %>% override_solution("attach(CPS1985); dummy_mod <- lm(wage ~ gender)") %>% check_object("dummy_mod") },{ ex() %>% override_solution("dummy_mod <- lm(CPS1985$wage CPS1985$gender)") %>% check_object("dummy_mod") },{ ex() %>% override_solution("dummy_mod <- lm(wage ~ gender, data = CPS1985)") %>% check_object("dummy_mod") }) success_msg("Well done!")

10. Gender Wage Gap II

The wage regression from the previous exercise yields

$$\widehat{wage}_i = 9.995 - 2.116 \cdot female_i.$$

The model object dummy_mod is available in your working environment.

Instructions

- Test the hypothesis that the coefficient on $female_i$ is zero. This would imply that there is no gender wage gap. Use White's (1980) heteroskedasticity-robust estimator.

Hint:

- vcovHC() computes heteroskedasticity-robust estimates of the covariance matrix of the coefficient estimators for the model supplied, see ?vcovHC.
- coeftest() performs significance tests for the coefficients in model objects. A covariance matrix can be supplied using the argument vcov..

library(AER) data(CPS1985) dummy_mod <- lm(wage ~ gender, data = CPS1985) # test whether the gender gap is significantly different from zero # use robust standard errors

coeftest(dummy_mod, vcov. = vcovHC(dummy_mod, type = "HC0")) # or linearHypothesis(dummy_mod, "genderfemale=0", vcov. = vcovHC(dummy_mod, type = "HC0")) test_predefined_objects("dummy_mod") test_or({ test_function("coeftest", args = c("x", "vcov.")) },{ test_function("linearHypothesis", args = c("model", "vcov.", "hypothesis.matrix")) },{ f <- ex() %>% override_solution('linearHypothesis(dummy_mod, "genderfemale", vcov. = vcovHC(dummy_mod, type = "HC0"))') %>% check_function("linearHypothesis") f %>% check_arg("model") f %>% check_arg("vcov.") f %>% check_arg("hypothesis.matrix") }) success_msg("Right! The hypothesis that there is no wage gap is rejected at any common level.")

11. Heteroskedasticity-Robust Standard Errors

In the simple regression model, the covariance matrix of the coefficient estimators is denoted

$$\text{Var}\begin{pmatrix}\hat{\beta}_0 \\ \hat{\beta}_1\end{pmatrix} = \begin{pmatrix}\text{Var}(\hat{\beta}_0) & \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \\ \text{Cov}(\hat{\beta}_0, \hat{\beta}_1) & \text{Var}(\hat{\beta}_1)\end{pmatrix} \tag{6.7}$$

The function vcovHC can be used to obtain estimates of this matrix for a model object of interest.

gender_mod, a model object containing the wage regression dealt with in exercises 9 and 10 is available in your working environment.

Instructions

- Compute robust standard errors of the type HC0 for the coefficients estimators in the model object gender_mod. Store the standard errors in a vector named rob_SEs.

Hint:

- The standard errors we seek can be obtained by taking the square root of the diagonal elements of the estimated covariance matrix.
- diag(A) returns the diagonal elements of the matrix A.

library(AER) data(CPS1985) dummy_mod <- lm(wage ~ gender, data = CPS1985) # compute robust standard errors and save them in rob_SEs

rob_SEs <- sqrt(diag(vcovHC(dummy_mod, typ = "HC0"))) test_predefined_objects("dummy_mod") test_object("rob_SEs") success_msg("Nice!")

12. Robust Confidence Intervals

The function confint() computes confidence intervals for regression models using homoskedasticity-only standard errors such that this function is not an option when there is heteroskedasticity.

The function Rob_CI() is meant to compute and report heteroskedasticity-robust confidence intervals for both model coefficients in a simple regression model.

gender_mod, a model object containing the wage regression dealt with in the previous exercises is available in your working environment.

Instructions

- Complete the code of Rob_CI() given in Script.R such that lower and upper bounds of 95% confidence intervals are returned.
- Use the function to obtain 95% confidence intervals for the model coefficients in dummy_mod.

library(AER) data(CPS1985) dummy_mod <- lm(wage ~ gender, data = CPS1985) # complete the function below

Rob_CI <- function(model) { SEs <- ??? lower <- model$coef - ???$upper < -model$coef + ??? return(cbind("Lower" = lower, "Upper" = upper)) }

Rob_CI <- function(model) { SEs <- sqrt(diag(vcovHC(model, type = "HC0"))) lower <- model$coef - 1.96 * SE$supper < -model$coef + 1.96 * SEs return(cbind("Lower" = lower, "Upper" = upper)) } test_function_definition("Rob_CI", function_test = test_expression_result("Rob_CI(dummy_mod)")) test_output_contains("Rob_CI(dummy_mod)") success_msg("Nice!")

13. A Small Simulation Study I

Consider the data generating process

$$\begin{aligned} X_i &\sim \mathcal{U}[2, 10], \\ e_i &\sim N(0, X_i), \\ Yi &= \beta_1 X_i + e_i \end{aligned}$$

where $\mathcal{U}[0, 10]$ denotes the uniform distribution on the interval $[0, 10]$ and $\beta_1 = 2$.

Notice that the errors e_i are heteroskedastic since the variance is a linear function of X_i.

Instructions

- Write a function DGP_OLS that generates a sample (X_i, Y_i), $i = 1, , 100$ and returns the OLS estimate of β_1.

Hints

runif() can be used to obtain random samples form a uniform distribution, see ?runif

write the function set.seed(1) # write the function DGP_OLS <- function() { X <- runif(100,2,10) Y <- X + rnorm(100, sd = sqrt(X)) return(c("beta_1_hat" = sum(X*Y)/sum(X^2))) } test_function_definition("DGP_OLS", function_test = test_expression_result("DGP_OLS()")) success_msg("Well Done!")

14. A Small Simulation Study II

The function DGP_OLS() from the previous exercise is available in your working environment.

Instructions

- Use replicate() to generate a sample of 1000 OLS estimates $\widehat{\beta}_1$ using the function DGP_OLS. Store the estimates in a vector named estimates.
- Estimate the variance of the OLS estimates. Store the result in est_var_OLS.

DGP_OLS <- function() { X <- runif(100,2,10) Y <- X + rnorm(100, sd = sqrt(X)) return(c("beta_1_hat" = sum(X*Y)/sum(X^2))) } set.seed(1) # Generate 1000 estimates of beta_1 using 'GDP_OLS()', store them in 'estimates'

Estimate the variance of the estimates

set.seed(1) # Generate 1000 estimates of beta_1 using 'GDP_OLS()', store them in 'estimates' estimates <- replicate(1000, DGP_OLS()) # Estimate the variance of the estimates est_var_OLS <- var(estimates) test_predefined_objects("DGP_OLS") test_object("estimates") test_object("est_var_OLS")

15. A Small Simulation Study III

The Gauss-Markov Theorem states that the OLS estimator in linear regression models is no longer the most efficient estimator (in the sense of minimum variance) among the conditionally unbiased linear estimators, that is the OLS estimator loses the BLUE property, under heteroskedasticity.

It turns out that OLS applied to the weighted observations $(w_i X_i, w_i Y_i)$ where $w_i = \frac{1}{\sigma_i}$ yields the BLUE estimator under heteroskedasticity. This estimator is called the *weighted least squares* (WLS) estimator so that its variance is smaller than the variance of the OLS estimator.

The function DGP_OLS() and the estimated variance est_var_OLS from the previous exercises are available in your working environment.

Instructions

- Write a function DGP_WLS() that generates 100 samples using the DGP introduced in exercise 13 and returns the WLS estimate of β_1. Treat σ_i as known, i.e. set $w_i = \frac{1}{X_i}$.
- Repeat exercise 14 using DGP_WLS(). Store the estimated variance in est_var_GLS.
- Compare the estimated variances est_var_OLS and est_var_GLS using logical operators (< or >).

Hints

- DGP_WLS() can be obtained using a modified code of DGP_OLS().
- Remember that functions are objects and you may print the code of a function to the console.

set.seed(1) DGP_OLS <- function() { X <- runif(100,2,10) Y <- X + rnorm(100, sd = sqrt(X)) return(c("beta_1_hat" = sum(X*Y)/sum(X^2))) } estimates <- replicate(1000, DGP_OLS()) est_var_OLS <- var(estimates) set.seed(1) # Define the function 'DGP_GLS()'

estimate the variance, assign the value to 'var_est_GLS'

compare the estimated variances

set.seed(1) # Define the function 'DGP_GLS()' DGP_GLS <- function() { X <- runif(100,2,10) Y <- X + rnorm(100, sd = sqrt(X)) w <- 1/sqrt(X) return(c("beta_1_hat" = sum(w^2XY)/sum((w*X)^2))) } # estimate the variance, assign the value to 'var_est_GLS' est_var_GLS <- var(replicate(1000, DGP_GLS()))

compare the estimated variances est_var_GLS < est_var_OLS test_predefined_objects(c("DGP_OLS","est_var_OLS") test_function_definition("DGP_GLS", function_test = test_expression_result("DGP_OLS()")

) test_object("est_var_GLS") test_or({ test_student_typed("est_var_GLS < est_var_OLS") },{ test_student_typed("est_var_GLS > est_var_OLS") },{ test_student_typed("est_var_OLS > est_var_GLS") },{ test_student_typed("est_var_OLS < est_var_GLS") }) success_msg("Nice! This shows that the GLS estimator indeed has lower variance than OLS.")

Chapter 7

Regression Models with Multiple Regressors

In what follows we introduce linear regression models that use more than just one explanatory variable and discuss important key concepts in multiple regression. As we broaden our scope beyond the relationship of only two variables (the dependent and a single regressor), some potential issues arise, such as *multicollinearity* and *omitted variable bias* (OVB). In particular, this chapter deals with omitted variables and their hazard to causal interpretation of OLS-estimated coefficients.

Naturally, we will introduce estimation of multiple regression models using R. We will also advocate thoughtful usage of multiple regression models by simulation studies in R that demonstrate the consequences of using highly correlated regressors or a misspecified model.

7.1 Omitted Variable Bias

Previous analysis of the relationship between test score and class size discussed in Chapters 4 and 5 has a major flaw: we ignored other potentially important determinants of test scores that vary with our regressor. The influences of those variables were collected in the error term. This might induce an estimation bias, i.e. the mean of the OLS estimator's sampling distribution is no longer equal to the true mean and we measure a wrong effect on test scores of a unit change in the student-teacher ratio. This is called omitted variable bias, see Key Concept 6.1.

Key Concept 6.1

Omitted Variable Bias in Regression with a Single Regressor

Omitted variable bias is the bias in the OLS estimator that arises when the regressor, X, is *correlated* with an omitted variable. For omitted variable bias to occur, two conditions must be fulfilled:

1. X is correlated with the omitted variable.
2. The omitted variable is a determinant of the dependent variable Y.

Together, 1. and 2. result in a violation of the first OLS assumption $E(u_i|X_i) = 0$. Formally, the resulting bias can be expressed as

$$\hat{\beta}_1 \xrightarrow{p} \beta_1 + \rho_{Xu}\frac{\sigma_u}{\sigma_X}. \tag{6.1}$$

See Appendix 6.1 in the book for a detailed derivation. (6.1) states that OVB is a problem that cannot be aleviated by increasing the numner of observations used to estimate β_1 since then $\hat{\beta}_1$ is inconsistent: OVB

prevents the estimator to converge in probability to the true parameter value. Strength and direction of the bias are driven by ρ_{Xu}, the correlation between the error term and the regressor.

In our example of test score and class size, it is fairly easy to come up with variables that may cause such a bias if omitted. As mentioned in the book, a highly relevant variable could be the percentage of english learners in the school district: it is plausible that the ability to speak, read and write english is an important factor for successful learning. Therefore, students that are still learning english are likely to perform worse in the tests than native speakers are. Also, it is conceivable that the share of english learning students is larger in school districts where class sizes are large. Think of less well-off urban districs were a lot of immigrants live.

Let us think about a possible bias induced by omitting the share of english learning students ($PctEL$) in view of (6.1) when the estimated regression model excludes $PctEL$ although the true DGP is

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times PctEL \tag{6.2}$$

where STR and $PctEL$ are correlated,

$$corr(STR, PctEL) \neq 0.$$

After defining our variables in R, we compute the correlation between STR and $PctEL$ as well as the correlation between STR and $TestScore$.

```
# load the AER package
library(AER)

# load the data set
data(CASchools)

# define variables
CASchools$STR <- CASchools$students/CASchools$teachers
CASchools$score <- (CASchools$read + CASchools$math)/2

# compute correlations
cor(CASchools$STR, CASchools$score)
```

```
## [1] -0.2263627
```

```
cor(CASchools$STR, CASchools$english)
```

```
## [1] 0.1876424
```

The fact that $\widehat{corr}(STR, Testscore) = -0.2264$ is cause for concern that omitting $PctEL$ leads to a negatively biased estimate $\hat{\beta}_1$ since then $\rho_{Xu} < 0$ and we have $\sigma_X > 0$, $\sigma_u > 0$ by definition. As consequence we expect $\hat{\beta}_1$, the coefficient on STR, to be too large in absolute value. Put differently, the OLS estimate of $\hat{\beta}_1$ suggests that small classes improve test scores but the estimated effect of small classes is too strong as it captures the effect of having fewer English learners, too.

What happens to the magnitude of $\hat{\beta}_1$ if we add the variable $PctEL$ to the regression, that is if we estimate the model

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times PctEL + u$$

instead? And what do we expect about the sign of, $\hat{\beta}_2$, the estimated coefficient on $PctEL$? Following the reasoning above we should end up with a (still) negative but larger coefficient estimate $\hat{\beta}_1$ and a negative estimate $\hat{\beta}_2$.

Let us estimate both regression models and compare. Performing a multiple regression in R is straightforward. One can simply add additional variables to the right hand side of the `formula` argument of the function `lm()` by using their names and the `+` operator. Notice that `english` is the name of the column in the data set `CASchools` which contains observations on the share of English learning students.

```
# estimate both regression models
mod <- lm(score ~ STR, data = CASchools)
mult.mod <- lm(score ~ STR + english, data = CASchools)

mod
```

```
##
## Call:
## lm(formula = score ~ STR, data = CASchools)
##
## Coefficients:
## (Intercept)          STR
##      698.93        -2.28
```

```
mult.mod
```

```
##
## Call:
## lm(formula = score ~ STR + english, data = CASchools)
##
## Coefficients:
## (Intercept)          STR      english
##    686.0322      -1.1013      -0.6498
```

We find the outcomes to be consistent with our expectations. The following section discusses some thoery on multiple regression models.

7.2 The Multiple Regression Model

In a multiple regression model we extend the basic concept of the simple regression model discussed in Chapter 4 and 5. A multiple regression model enables us to estimate the effect on Y_i of changing a regressor X_{1i} if the remaining regressors $X_{2i}, X_{3i} \dots, X_{ki}$ *do not vary*. In fact we already have performed estimation of the multiple regression model (6.2) using R in the previous section. Recall that the interpretation of the coefficient on student-teacher ratio is the effect on test scores of a one unit change student-teacher ratio if the percentage of English learners is kept constant.

As in the simple regression model, we assume the true relationship between Y and $X_{2i}, X_{3i} \dots$ to be linear. On average, this relation is given by the population regression function

$$E(Y_i|X_{1i} = x_1, X_{2i} = x_2, X_{3i} = x_3, \dots, X_{ki} = x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k. \tag{6.3}$$

As in the simple regression model, this relation does not hold exactly since there are disturbing influences to the dependent variable Y we cannot observe as explanatory variables. Therefore we add an error term u which represents deviations of the observations from the population regression line to (6.3). This yields the population multiple regression model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \cdots + \beta_k X_{ki} + u_i \quad , \quad i = 1, \dots, n. \tag{6.4}$$

Key Concept 6.2

The Multiple Regression Model

The multiple regression model is

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \cdots + \beta_k X_{ki} + u_i \quad , \quad i = 1, \ldots, n.$$

Designations are similar to those in the simple regression model:

- Y_i is the i^{th} observation in the dependent variable. Observations on the k regressors are denoted by $X_{1i}, X_{2i}, \ldots, X_{ki}$ and u_i is the error term.
- The average relationship between Y and the regressors is given by the population regression line

$$E(Y_i|X_{1i} = x_1, X_{2i} = x_2, X_{3i} = x_3, \ldots, X_{ki} = x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k.$$

- β_0 is the intercept; it is the expected value of Y when all Xs equal 0. β_j , $j = 1, \ldots, k$ are the coefficients on X_j , $j = 1, \ldots, k$. β_1 measures the expected change in Y_i that results from a one unit change in X_{1i} while holding all other regressors X_{ji} , $j \neq 1$ constant.

Key Concept 6.2 summarizes the core concepts of the multiple regression model. How can we estimate the coefficients of the multiple regression model (6.4)? We will not go to much into detail on this issue as our focus lies on usage of R instead of the technical refinements. However it should be pointed out that, similarly to the simple regression model, the coefficients of the multiple regression model can be estimated using OLS. As in the simple model, we seek to minimize the sum of squared prediction mistakes by choosing estimates $b_0, b_1, \ldots, b_k$ for the coefficients $\beta_0, \beta_1, \ldots, \beta_k$ such that

$$\sum_{i=1}^{n} (Y_i - b_0 - b_1 X_{1i} - b_2 X_{2i} - \cdots - b_k X_{ki})^2 \tag{6.5}$$

is minimized. Note that (6.5) is simply an extension of SSR in the case with just one regressor and an intercept. The estimators that minimize (6.5) are hence denoted $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$ and, as in the simple regression model, we call them the ordinary least squares estimators of $\beta_0, \beta_1, \ldots, \beta_k$. For the predicted value of Y_i given the regressors and the estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$ we have

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \cdots + \hat{\beta}_k X_{ki}.$$

and, as before, the difference of Y_i and its predicted value $\hat{Y}_i$ is called the OLS residual of observation i: $\hat{u} = Y_i - \hat{Y}_i$.

For further information regarding the theory behind multiple regression, see Chapter 18.1 in the book which inter alia presents a derivation of the OLS estimator in the multiple regression model using matrix notation.

Now let us jump back to the example of test scores and class sizes. The estimated model object is `mult.mod`. As for simple regression models we can use the `summary()` function to obtain information on estimated coefficients and model statistics.

```
summary(mult.mod)$coef
```

```
##                Estimate Std. Error    t value      Pr(>|t|)
## (Intercept) 686.0322445 7.41131160  92.565565 3.871327e-280
## STR          -1.1012956 0.38027827  -2.896026  3.978059e-03
## english      -0.6497768 0.03934254 -16.515882  1.657448e-47
```

So the estimated multiple regression model is

$$\widehat{TestScore} = \underset{(7.41)}{686.03} - \underset{(0.38)}{1.10} \times STR - \underset{(0.04)}{0.65} \times PctEL. \tag{6.6}$$

Unlike in the simple regression model where the data can be represented by points in the two-dimensional cartesian coordinate system, we are now dealing with three dimensions. Hence observations can be represented by points in the three-dimensional real space, denoted $\mathbb{R}^3$. Therefore (6.6) is now longer a regression line but a *regression plane*. This idea extends to higher dimensions when we further expand the number of regressors k. We then say that the regression model can be represented a hyperplane in the $k+1$ dimensional space. It is already hard to imagine such a space if $k = 3$ and we best stick with the general idea that, in the multiple regression model, the dependent variable is explained by a *linear combination of the regressors*. However, in the present case we are able to visualize the situation. The following figure is an interactive 3D visualization of the data and the estimated regression plane (6.6).

We observe that the estimated regression plane fits the data reasonably well — at least with regard to the shape and spatial poistion of the point cloud. The color of the markers is an indicator for the absolute deviation from the predicted regression plane. Observations that are coloured more reddish lie close to the regression plane while the color shifts to blue with growing distance. An anomaly that can be seen from the plot is that there might be heteroskedasticity: we see that the dispersion of regression errors made, i.e. the distance of observations to the regression plane shows a tendency to decrease as the share of English learning students increases.

7.3 Measures of Fit in Multiple Regression

In multiple regression, common summary statistics are SER, $R\hat{}2$ and the adjusted R^2.

Taking the code from Section 6.3 You could simply use `summary(mult.mod)` to print the SER, R^2 and adjusted-R^2. For multiple regression models the SER is computed as

$$SER = s_{\hat{u}} = \sqrt{s^2_{\hat{u}}}$$

where, in contrast to the simple regression model we apply a modification to the denominator of the premultiplied factor in $s^2_{\hat{u}}$ in order to accommodate for additional regressors. Thus,

$$s^2_{\hat{u}} = \frac{1}{n-k-1} SSR$$

with k the number of regressors *excluding* the intercept. While `summary()` computes the R^2 just as in the case of a single regressor, it is not a reliable measure for multiple regression models. This is due to R^2 becoming larger every time an additional regressor is added to the model since adding a regressor decreases the SSR — at least unless the respective estimated coefficient is exactly zero what practically never happens to be the case (see chapter 6.4 in the book). The adjusted R^2 takes this into consideration by "punishing" the addition of regressors using a correction factor. So the adjusted R^2 or simply $\overline{R^2}$ is a modified version of R^2.

$$\overline{R^2} = 1 - \frac{n-1}{n-k-1} \frac{SSR}{TSS}$$

As You may have already suspected, `summary()` adjusts the formula for SER by itself and it computes $\overline{R^2}$ and of course R^2 by default, therby leaving the decision which measure to rely on up to the user.

You can find the measures at the bottom of the output produced by calling `summary(mult.mod)`.

```
summary(mult.mod)
```

```
##
## Call:
## lm(formula = score ~ STR + english, data = CASchools)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.845 -10.240  -0.308   9.815  43.461
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 686.03224    7.41131  92.566  < 2e-16 ***
## STR          -1.10130    0.38028  -2.896  0.00398 **
## english      -0.64978    0.03934 -16.516  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.46 on 417 degrees of freedom
## Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
## F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

We can also compute the measures by hand using the formulas above. Let us check if the results coincide with the values provided by `summary()`.

```
# define the components
n <- nrow(CASchools)                              # number of observations (rows)
k <- 2                                            # number of regressors

y_mean <- mean(CASchools$score)                   # mean of avg. test-scores

SSR <- sum(residuals(mult.mod)^2)                 # sum of squared residuals
TSS <- sum((CASchools$score - y_mean )^2)         # total sum of squares
ESS <- sum((fitted(mult.mod) - y_mean)^2)         # explained sum of squares

# compute the measures

SER <- sqrt(1/(n-k-1) * SSR)                      # standard error of the regression
Rsq <- 1 - (SSR / TSS)                            # R^2
adj_Rsq <- 1 - (n-1)/(n-k-1) * SSR/TSS            # adj. R^2

# Print the measures to the console
c("SER" = SER, "R2" = Rsq, "Adj.R2" = adj_Rsq)
```

```
##        SER         R2     Adj.R2
## 14.4644831  0.4264315  0.4236805
```

We find that the results do match. Now, what can be said about the fit of our multiple regression model for test scores with the percentage of english learners as an additional regressor? Does it improve on the simple model including only an intercept and a measure of the class size? The answer is yes: this is easily seen by comparing these measures of fit with those for the simple regression model `mod`.

```
# SER
summary(mod)$sigma
```

```
## [1] 18.58097
```

```
# R^2
summary(mod)$r.squared
```

```
## [1] 0.05124009
```

```
# Adj. R^2
summary(mod)$adj.r.squared
```

```
## [1] 0.04897033
```

Including $PctEL$ as a regressor boots the R^2 from about 5% to 42.6%. Qualitatively the same is observed for $\overline{R^2}$ which we deem to be more reliable in view of the discussion above. Notice that the difference between R^2 and $\overline{R^2}$ is small since $k = 2$ and n is large. Condensed, the fit of (6.6) improves vastly on the fit of the simple regression model with STR as the only regressor. Comparing prediction errors we find that the prediction precision of the multiple regression model (6.6) improves upon the simple model as adding $PctEL$ lowers the SER from 18.6 to 14.5 units of test score.

As already mentioned R^2 and $\overline{R^2}$ may be used to quantify how good a model fits the data. However it is rarely a good idea to maximize these measures by stuffing the model with regressors in general. You will (hopefully) not find any serious study that does so. Instead it is more fruitful to include regressors that improve estimation of the causal effect of interest which is *not* assessed by means of a low SSR. The issue of variable selection is covered in Chapter 7 of this script and Chapter 7 in the book.

7.4 OLS Assumptions in Multiple Regression

In the multiple regression model we extend the known three least squares assumptions imposed for the simple regression model (see Chapter 4, Key Concept 4.3) and add a fourth assumption. These assumptions are presented in Key Concept 6.4. While we will not go into the details of assumptions 1, 2 and 3 since their ideas have been discussed before and are easily generalized to the case of multiple regressors, we will devote our attention to the fourth assumption. This forth assumption forbids the occurence of perfect correlation between any pair of regressors.

Key Concept 6.4

The Least Squares Assumptions in the Multiple Regression Model

The multiple regression model is given by

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_1 X_{2i} + \cdots + \beta_k X_{ki} + u_i \ , \ i = 1, \ldots, n.$$

The OLS assumptions in the multiple regression model are an extension of the ones made for the simple regression model:

1. Regressors $(X_{1i}, X_{2i}, \ldots, X_{ki}, Y_i) \ , \ i = 1, \ldots, n$ are drawn such that the i.i.d. assumption holds.
2. u_i is an error term with conditional zero given the regressors, i.e.
$$E(u_i | X_{1i}, X_{2i}, \ldots, X_{ki}) = 0.$$
3. Large outliers are unlikely, formally $X_{1i}, \ldots, X_{ki}$ and Y_i have finite fourth moments.
4. No perfect multicollinearity.

Multicollinearity

If two or more regressors in a multiple regression model are *strongly* correlated we say that there is *multicollinearity*. If the correlation between two or more regressors is perfect (correlation coeficient equals 1), we refer to the situation as *perfect multicollinearity*. Perfectly multicollinear regressors can be written as linear combinations of each other. While strong multicollinearity in general is unpleasent as it causes the variance of the OLS estimator to be large (we will discuss is in more detail later), the presence of perfect multicollinearity makes it even impossible to solve for the OLS estimator, that is the model cannot be estimated at all.

The next section presents some examples of perfect multicollinearity and demonstrates how R, specifically the `lm` function deals with them.

Examples of Perfect Multicollinearity

How does R react if we want it to estimate a model with perfectly correlated regressors?

If we use the `lm` function to estimate a model with a set of regressors that suffer from perfect multicollinearity the system will produce a warning in the first line of the coefficient section of the output (1 not defined because of singularities) and ignore the regressor(s) which is (are) assumed to be a linear combination of the others. See the following example where we add another variable `FracEL`, the fraction of Englisch learners to `CASchools` where observations are scaled values of the observations for `english` and use it as a regressor together with `STR` and `english` in a multiple regression model. In this example `english` and `FracEL` are perfectly collinear. The R code is as follows.

```
# define the fraction of english learners
CASchools$FracEL <- CASchools$english/100

# estimate the model
mult.mod <- lm(score ~ STR + english + FracEL, data = CASchools)

# obtain a summary of the model
summary(mult.mod)
```

```
##
## Call:
## lm(formula = score ~ STR + english + FracEL, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.845 -10.240  -0.308   9.815  43.461
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 686.03224    7.41131  92.566  < 2e-16 ***
## STR          -1.10130    0.38028  -2.896  0.00398 **
## english      -0.64978    0.03934 -16.516  < 2e-16 ***
## FracEL             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.46 on 417 degrees of freedom
## Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
## F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

Notice that the row `FracEL` in the coefficients section of the output consists of `NA` entries since `FracEL` was excluded from the model.

If we were to compute OLS by hand we would run into the problem as well but no one would be helping us out! The computation simply fails. Why is this the case? Take the following example:

Assume You want to estimate a simple linear regression model with an intercept and one regressor:

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

As mentioned above, for perfect multicollinearity to be present X has to be a linear combination of the other regressors. Since the only other regressor is a constant (think of the right hand side of the model equation as $\beta_0 \times 1 + \beta_1 X_i + u_i$ so that β_1 is always multiplied by 1 for every observation), X has to be constant as well. For $\hat{\beta}_1$ we have

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} = \frac{\widehat{cov}(X,Y)}{\widehat{Var}(X)}. \tag{6.7}$$

So the variance of the regressor X stands in the denominator. Since the variance of a constant is zero, we are not able to compute this fraction and $\hat{\beta}_1$ remains undefined.

Note: In this special case the nominator in (6.7) equal zero, too. Can You show that?

Let us behold two further examples where our selection of regressors induces perfect multicollinearity. First, assume that we intend to analyse the effect of class size on test score by using a dummy variable that identifies "Not very small" classes (NVS). We define that a school has the NVS attribute when the school's average student-teacher ratio is at least 12.

$$NVS = \begin{cases} 0 & \text{if STR} < 12 \\ 1 & \text{otherwise} \end{cases}$$

We add the corresponding column to `CASchools` and estimate a multiple regression model with covariates `computer` and `english`.

```
# if STR smaller 12, NVS = 0, else NVS = 1
CASchools$NVS <- ifelse(CASchools$STR < 12, 0, 1)

# estimate the model
mult.mod <- lm(score ~ computer + english + NVS, data = CASchools)

# obtain a model summary
summary(mult.mod)
```

```
##
## Call:
## lm(formula = score ~ computer + english + NVS, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.492  -9.976  -0.778   8.761  43.798
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 663.704837   0.984259 674.319  < 2e-16 ***
## computer      0.005374   0.001670   3.218  0.00139 **
## english      -0.708947   0.040303 -17.591  < 2e-16 ***
## NVS                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.43 on 417 degrees of freedom
## Multiple R-squared:  0.4291, Adjusted R-squared:  0.4263
## F-statistic: 156.7 on 2 and 417 DF,  p-value: < 2.2e-16
```

Again, the output of `summary(mult.mod)` tells us that inclusion of `NVS` in the regression would render the estimation infeasible. What happened here? This is an example where we made a logical mistake when defining the regressor `NVS`: if we had taken a closer look at `NVS`, the redefined measure for class size, we would have noticed that there is not a single school with $STR < 12$ hence NVS equals one for all observations. We can check this by printing the contents of `CASchools$NVS` to the console or by using the function `table`, see `?table`.

```
table(CASchools$NVS)
```

```
##
##   1
## 420
```

`CASchools$NVS` is a vector of 420 ones and our data set includes 420 observations. This obviously violates assumption 4 of Key Concept 6.4: remember that observations for the intercept are always 1 so we have

$$intercept = \lambda \cdot NVS \tag{7.1}$$

$$\tag{7.2}$$

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \tag{7.3}$$

$$\Leftrightarrow \lambda = 1. \tag{7.4}$$

Since both regressors can be written as a linear combination of each other, we face perfect multicollinearity and R excludes NVS from the model. Thus the message to take away is: think carefully about how regressors You created Yourself could possibly interact with unrecognized features of the data set!

Another example of perfect multicollinearity is known as the "dummy variable trap". This may occur when multiple dummy variables are used as regressors. A common case for this is when the dummies are used to sort the data set into mutually exclusive categories. For example, suppose we have spatial information that indicates whether a school is located in the North, West, South or East of the US which allows us to create the dummy variables

$$North_i = \begin{cases} 1 & \text{if located in the north} \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

$$\tag{7.6}$$

$$West_i = \begin{cases} 1 & \text{if located in the west} \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

$$\tag{7.8}$$

$$South_i = \begin{cases} 1 & \text{if located in the south} \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

$$\tag{7.10}$$

$$East_i = \begin{cases} 1 & \text{if located in the east} \\ 0 & \text{otherwise.} \end{cases} \tag{7.11}$$

Since the directions are mutually exclusive, for every school $i = 1, \dots, n$ we have

$$North_i + West_i + South_i + East_i = 1.$$

We run into problems when trying to estimate a model that includes a constant and *all four* direction dummies, e.g. in the model

$$TestScore = \beta_0 + \beta_1 \times STR + \beta_2 \times english + \beta_3 \times North_i + \beta_4 \times West_i + \beta_5 \times South_i + \beta_6 \times East_i + u_i \quad (6.8)$$

since then for all observations $i = 1, \dots, n$ the constant term can be written as a linear combination of the dummies:

$$intercept = \lambda_1 \cdot (North + West + South + East) \quad (7.12)$$

$$(7.13)$$

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \quad (7.14)$$

$$\Leftrightarrow \lambda = 1 \quad (7.15)$$

and we have perfect multicollinearity. This is what is meant by the "dummy variable trap": not paying attention and erroneously including all dummies and an intercept term in a regression model.

How does the `lm` function in R handle a regression like (6.8)? To answer this we first generate some artifical categorical data and append a new column named `directions` to `CASchools` and see what `lm` does when asked to estimate the model above.

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

# Generate artificial data on location
CASchools$direction <- sample(c("West","North","South","East"), 420, replace = T)

# estimate the model
mult.mod <- lm(score ~ STR + english + direction, data = CASchools)

# obtain a model summary
summary(mult.mod)
```

```
##
## Call:
## lm(formula = score ~ STR + english + direction, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.018 -10.098  -0.556   9.304  42.596
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    685.67356    7.43308  92.246  < 2e-16 ***
## STR             -1.12246    0.38231  -2.936  0.00351 **
## english         -0.65096    0.03934 -16.549  < 2e-16 ***
## directionNorth   1.60680    1.92476   0.835  0.40431
## directionSouth  -1.17013    2.07665  -0.563  0.57342
## directionWest    2.44340    2.05191   1.191  0.23442
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.45 on 414 degrees of freedom
## Multiple R-squared:  0.4315, Adjusted R-squared:  0.4246
## F-statistic: 62.85 on 5 and 414 DF,  p-value: < 2.2e-16
```

Notice that R solves the problem sketched above on its own by generating and including the dummies `directionNorth`, `directionSouth` and `directionWest` but ommitting `directionEast`. Of course, omission of every other dummy instead would be achieve the same. This is done by convention. Another solution would be to exclude the intercept and include all dummies instead. Does this mean that the information on schools located in the East is discarded? Furtunately, this is not the case: exclusion of `directEast` just alters the interpretation of coefficient estimates on the remaining dummies from absolute to relative. For example, the coefficient estimate on `directionNorth` states that, on average, test scores in the North are about 1.61 points higher than in the East.

A last example considers the case where a perfect linear relationship arises from redundant regressors. Suppose we have a regressor $PctES$, the percentage of English speakers in the school where

$$PctES = 100 - PctEL$$

and both $PctES$ and $PctEL$ are included in a regression model. One regressor is redundant since the other one conveys the same information. Since this obviously is a case where the regressors can be written as linear combination, we end up with perfect multicollinearity, again.

Let us do this in R.

```
# Percentage of english speakers
CASchools$PctES <- 100 - CASchools$english

# estimate the model
mult.mod <- lm(score ~ STR + english + PctES, data = CASchools)

# obtain a model summary
summary(mult.mod)
```

```
##
## Call:
## lm(formula = score ~ STR + english + PctES, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.845 -10.240  -0.308   9.815  43.461
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 686.03224    7.41131  92.566  < 2e-16 ***
## STR          -1.10130    0.38028  -2.896  0.00398 **
## english      -0.64978    0.03934 -16.516  < 2e-16 ***
## PctES             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.46 on 417 degrees of freedom
## Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
## F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

Once more, the `lm` function refuses to estimate the full model using OLS and excludes `PctES`.

See chapter 18.1 of the book for an explanation of perfect multicollinearity and its consequences to the OLS estimator in general multiple regression models using matrix notation.

Imperfect Multicollinearity

As opposed to perfect multicollinearity, imperfect multicollinearity is — to a certain extend — not a problem at all. In fact, imperfect multicollinearity is the reason why we are interested in estimating multiple regression models in the first place: the OLS estimator allows us to *isolate* influences of *correlated* regressors on the dependent variable. So if it was not for these dependencies, there would not be a reason to resort to a multiple regression approach and we could simply work with a single-regressor model. However, reality is complicated and this is rarely the case. Moreover, we already know that ignoring dependencies among regressors which influence the outcome variable has an adverse effect on estimation results (keyword: omitted variable bias!).

So when and why is imperfect multicollinearity a problem? Suppose You got the regression model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + u_i \tag{6.9}$$

and You are interested in estimating β_1, the effect on Y_i of a one unit change in X_{1i} while holding X_{2i} constant. You do not know that the true model indeed includes X_2 but You follow some reasoning and add X_2 as a covariate to the model in order to address a potential omitted variable bias. You are confident that $E(u_i|X_{1i}, X_{2i}) = 0$ for all observations and there is no reason to suspect violation of the assumptions 2 and 3 made in Key Concept 6.4. If now X_1 and X_2 are highly correlated, OLS has its difficulties to estimate β_1 precisely. That means although $\hat{\beta}_1$ is a consistent and unbiased estimator for β_1, it has a large variance due to X_2 being included in the model. If the errors are homoskedastic, this issue can be better understood from inspecting the formula for the variance of $\hat{\beta}_1$ in the model (6.9) (cf. Appendix 6.2 of the book):

$$\sigma^2_{\hat{\beta}_1} = \frac{1}{n}\left(\frac{1}{1-\rho^2_{X_1,X_2}}\right)\frac{\sigma^2_u}{\sigma^2_{X_1}} \tag{6.10}$$

Firstly, notice that if $\rho_{X_1,X_2} = 0$ i.e. if there is no correlation between both regressors, including X_2 in the model has no influence on the variance of $\hat{\beta}_1$. Secondly, if X_1 and X_2 are correlated, $\sigma^2_{\hat{\beta}_1}$ is inversely proportional to $1-\rho^2_{X_1,X_2}$ so the stronger the correlation between X_1 and X_2, the smaller is $1-\rho^2_{X_1,X_2}$ and thus the bigger is the variance of $\hat{\beta}_1$. Thirdly, (6.10) reveals that, in any case, increasing the sample size helps to reduce the variance of $\hat{\beta}_1$. Of course, this is not limited to the case with two regressors: in general multiple regression, imperfect multicollinearity inflates the variance of one or more coefficient estimators and it is an empirical question which estimators are severly affected by this and which are not. So is the decision whether one wants to hazard the consequences of adding a large number of covariates when the sample size is small.

In sum, undesirable consequences of imperfect multicollinearity are not the result of a logical error made by the researcher (as is often the case for perfect multicollinearity) but are rather a problem that is linked to the data used, the model to be estimated and the research question at hand.

Let us conduct a simulation study to illustrate the issues sketched above.

1. We use (6.9) as the data generating process and choose $\beta_0 = 5$, $\beta_1 = 2.5$ and $\beta_2 = 3$ and u_i is an error term distributed as $\mathcal{N}(0, 5)$. In a first step, we sample the regressor data from a bivariate normal distribution:

$$X_i = (X_{1i}, X_{2i}) \overset{i.i.d.}{\sim} \mathcal{N}\left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}10 & 2.5\\2.5 & 10\end{pmatrix}\right]$$

   It is straightforward to see that the correlation between X_1 and X_2 in the population is rather low:

$$\rho_{X_1,X_2} = \frac{cov(X_1, X_2)}{\sqrt{Var(X_1)}\sqrt{Var(X_2)}} = \frac{2.5}{10} = 0.25$$

2. Next, we estimate the model (6.9) and save the estimates for β_1 and β_2. This is repeated 10000 times with a `for` loop so we end up with a large number of estimates that allow us to describe the distributions of $\hat{\beta}_1$ and $\hat{\beta}_2$.

3. We repeat steps 1 and 2 but increase the covariance between X_1 and X_2 from 2.5 to 8.5 such that the correlation between the regressors is high:

$$\rho_{X_1,X_2} = \frac{cov(X_1, X_2)}{\sqrt{Var(X_1)}\sqrt{Var(X_2)}} = \frac{8.5}{10} = 0.85$$

4. In order to assess the effect on the precision of the estimators of increasing the collinearity between X_1 and X_2 we compute estimates of the variances of $\hat{\beta}_1$ and $\hat{\beta}_2$ and compare.

```
# load packages
library(MASS)
library(mvtnorm)

# set number of observations
n <- 50

# initialize vectors of coefficients
coefs1 <- cbind("hat_beta_1" = numeric(10000), "hat_beta_2" = numeric(10000))
coefs2 <- coefs1

# set random seed
set.seed(1)

# loop sampling and estimation
for (i in 1:10000) {

  # for cov(X_1,X_2) = 0.25
  X <- rmvnorm(n, c(50,100), sigma = cbind(c(10,2.5), c(2.5,10)))
  u <- rnorm(n, sd=5)
  Y <- 5 + 2.5*X[,1] + 3*X[,2] + u
  coefs1[i,] <- lm(Y ~ X[,1] + X[,2])$coefficients[-1]

  # for cov(X_1,X_2) = 0.85
  X <- rmvnorm(n, c(50,100), sigma = cbind(c(10,8.5), c(8.5,10)))
  Y <- 5 + 2.5*X[,1] + 3*X[,2] + u
  coefs2[i,] <- lm(Y ~ X[,1] + X[,2])$coefficients[-1]
}

# estimate the variances
var(coefs1)
```

```
##             hat_beta_1  hat_beta_2
## hat_beta_1  0.05674375 -0.01387725
## hat_beta_2 -0.01387725  0.05712459
```

```
var(coefs2)
```

```
##            hat_beta_1 hat_beta_2
## hat_beta_1  0.1904949 -0.1610405
```

```
## hat_beta_2 -0.1610405  0.1909056
```

Since we call `var` on matrices rather than vectors, the oucomes are variance-covariance matrices. We are interested in the variances which are the diagonal elements. We see that due to the high collinearity, the variances of $\hat{\beta}_1$ and $\hat{\beta}_2$ have more than tripled: in Econometrics lingo we say that it has become more difficult to estimate the true coefficients precisely.

7.5 The Distribution of the OLS Estimators in Multiple Regression

As in simple linear regression, different samples will produce different values of the OLS estimators in the multiple regression model. Here again this variation leads to uncertainty of those estimators and we seek to describe it using their sampling distribution(s). In short, if the assumption made in Key Concept 6.4 hold, the large sample distribution of $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ is multivariate normal and the individual estimators themselves are also normally distributed. Key Concept 6.5 summarizes the corresponding statements made in Chapter 6.6 of the book. A more technical derivation of these results can be found in Chapter 18 of the book.

Key Concept 6.5

Large-sample distribution of $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$

If the least squares assumptions in the multiple regression model (see Key Concept 6.4) hold, then in large samples, the OLS estimators $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ are jointly normally distributed. We also say that their joint distribution is *multivariate* normal. Further, each $\hat{\beta}_j$ is distributed as $N(\beta_j, \sigma^2_{\hat{\beta}_j})$.

Essentially, Key Concept 6.5 states that, if the sample size is large, we can approximate the individual sampling distributions of the coefficient estimators by specific normal distributions and their joint sampling distribution by a multivariate normal distribution.

How can we use R two get a notion of what the joint PDF of the coefficient estimators in multiple regression model looks like? When estimating some model on arbitrary data once, we would end up with a set of point estimates that do not reveal any information on the joint density of the estimators. However, with a large number of estimations using repeatedly randomly sampled data from the same population we could generate a large set of point estimates that allows us to plot an *estimate* of the joint density function.

The approach we will use is a follows:

- Generate 10000 random samples of size 75 using the DGP

$$Y_i = 5 + 2.5 \cdot X_{1i} + 3 \cdot X_{2i} + u_i$$

where the regressors X_{1i} and X_{2i} are sampled for each observations as

$$X_i = (X_{1i}, X_{2i}) \sim \mathcal{N}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 10 & 2.5 \\ 2.5 & 10 \end{pmatrix}\right]$$

and

$$u_i \sim \mathcal{N}(0, 5)$$

is an error term.

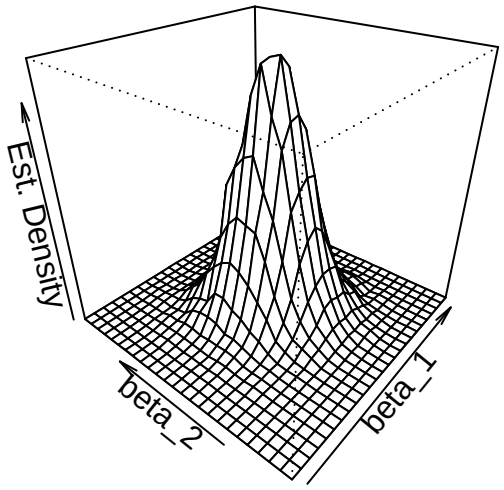
- For each of the 10000 simulated sets of sample data, we estimate the model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + u_i$$

and save the coefficient estimates $\hat{\beta}_1$ and $\hat{\beta}_2$.

- We compute a density estimate of the joint distribution of $\hat{\beta}_1$ and $\hat{\beta}_2$ in the model above using the function `kde2d` from the package `MASS`, see `?MASS`. This estimate is then plotted using the function `persp`.

```
# load packages
library(MASS)
library(mvtnorm)

# set sample size
n <- 50

# initialize vector of coefficients
coefs <- cbind("hat_beta_1" = numeric(10000), "hat_beta_2" = numeric(10000))

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

# loop sampling and estimation
for (i in 1:10000) {
  X <- rmvnorm(n, c(50,100), sigma = cbind(c(10,2.5), c(2.5,10)))
  u <- rnorm(n, sd=5)
  Y <- 5 + 2.5*X[,1] + 3*X[,2] + u
  coefs[i,] <- lm(Y ~ X[,1] + X[,2])$coefficients[-1]
}

# Compute density estimate
kde <- kde2d(coefs[,1], coefs[,2])

# plot density estimate
persp(kde,  theta = 310, phi = 30, xlab = "beta_1", ylab = "beta_2", zlab = "Est. Density")
```



From the plot above we can see that the density estimate has some similarity to a bivariate normal distribution (see chapter 2) though it is not very pretty and probably a little rough. Furthermore, there is correlation between the estimators such that $\rho \neq 0$ in (2.1). Also, the distribution's shape deviates from the symmetric bell shape of the bivariate standard normal distribution and has an eliptical surface area instead.

```
# estimate correlation between estimators
cor(coefs[,1], coefs[,2])
```

```
## [1] -0.2503028
```

Where does this correlation come from? Notice that, due to the way we generated the data, there is correlation between the regressors X_1 and X_2. Correlation between the regressors in a multiple regression model always translates to correlation between the estimators (see Appendix 6.2 of the book). In our case, the positive correlation between X_1 and X_2 translates to negative correlation between $\hat{\beta}_1$ and $\hat{\beta}_2$. To get a better feeling You can vary the point of view in the subsequent smooth interactive 3D plot of the same density estimate used for plotting with `persp`. Here You can see that the shape of the distribution is somewhat stretched due to $\hat{\rho} = -0.20$ and it is also apparent that both estimators are unbiased since their joint density seems to be centered close to the true parameter vector $(\beta_1, \beta_2) = (2.5, 3)$.

7.6 Exercises

For the course of this section, you will work with Boston, the Boston Housing data set which contains 506 observations on housing values in suburbs of Boston. The data set comes with the package MASS which is already installed for the interactive R exercises below.

1. The Boston Housing Data Set

Instructions

- Load both the package and the data set.
- Get yourself an overview over the data set using the summary function. Use ?Boston for detailed information on the variables.
- Estimate a simple linear regression model explaining the median house value (medv) by the percent of households with low socioeconononomic status, lstat, and a constant. Save the model to bh_mod.
- Print a coefficient summary to the console that reports robust standard errors.

Hint

You only need basic functions here: library(), data(), lm() and coeftest().

attach both packages and load the data set

Obtain an overview over the data set

Estimate the simple regression model

print the summary to the console

attach both packages and load the data set library(AER) library(MASS) data("Boston") # Obtain an overview over the data set summary(Boston) # Estimate the simple regression model bh_mod <- lm(medv ~ lstat, data = Boston) # print the summary to the console coeftest(bh_mod, vcov. = vcovHC) test_function("library", index = 1) test_function("library", index = 2) test_function("data")

test_or({ test_function("coeftest", args=c("x","vcov.")) },{ f <- ex() %>% override_solution("coeftest(bh_mod, vcov. = vcovHC)") %>% check_function('coeftest') f %>% check_arg("x") %>% check_equal() f %>% check_arg("vcov.") %>% check_equal() },{ f <- ex() %>% override_solution("coeftest(bh_mod, vcov. = vcovHC(bh_mod, type = 'HC0'))") %>% check_function('coeftest') f %>% check_arg("x") %>% check_equal() f %>% check_arg("vcov.") %>% check_equal() },{ f <- ex() %>% override_solution("coeftest(bh_mod, vcov. = vcovHC(bh_mod, type = 'HC1'))") %>% check_function('coeftest') f %>% check_arg("x") %>% check_equal() f %>% check_arg("vcov.") %>% check_equal() })

test_or({ test_object("bh_mod") },{ f <- ex() %>% override_solution('lm(Boston*medv* *Boston*lstat)') %>% check_function('lm') f %>% check_arg('formula') %>% check_equal() },{ f <- ex() %>% override_solution('attach(Boston);lm(medv ~ lstat)') %>% check_function('lm') f %>% check_arg('formula') %>% check_equal() })

success_msg("Correct. Notice both model coefficients are highly significant.")

2. A Multiple Regression Model of Housing Prices I

Now, let us expand the idea from the previous exercise by adding additional regressors to the model and estimate it again.

As has been discussed in chapter 6.3, adding regressors to the model improves the fit, that is the SER decreases and the R^2 increases.

The packages AER and MASS have been loaded. The model object bh_mod is available in the environment.

Instructions

- Regress the median housing value in a destrict, medv, on the average age of the buildings, age, the per capita crime rate, crim, the percentage of individuals with low socioeconomic status, lstat, and a constant. Put differently, estimate the model

$$medv_i = \beta_0 + \beta_1 lstat_i + \beta_2 age_i + \beta_3 crim_i + u_i.$$

- Print a coefficient summary to the console that reports robust standard errors for the augmented model.
- The simple regression model's R^2 is stored in R2_res. Save the multiple regression model's R^2 to R2_unres and check whether the augmented model yields a higher R^2. Use < or > for the comparison.

library(AER) library(MASS) data("Boston") attach(Boston) bh_mod <- lm(medv ~ lstat, data = Boston) R2_res <- summary(bh_mod)$r.squared # Conduct the regression

Obtain a robust coefficient summary

Compare both coefficients of determination

Conduct the regression mod <- lm(medv ~ lstat + crim + age, data = Boston)

Obtain a robust coefficient summary coeftest(mod, vcov. = vcovHC)

Compare both coefficients of determination R2_unres <- summary(mod)$r.squared R2_unres < R2_res

test_or({ ex() %>% check_function('lm') %>% check_result() },{ ex() %>% override_solution('lm(Boston*medv* *Boston*lstat + Boston*crim* + *Boston*age)') %>% check_function('lm') %>% check_result() },{ ex() %>% override_solution('lm(medv ~ lstat + crim + age)') %>% check_function('lm') %>% check_result() })

test_or({ test_function("coeftest") },{ f <- ex() %>% override_solution('coeftest(lm(Boston*medv* *Boston*lstat + Boston*crim* + *Boston*age), vcov. = vcovHC)') %>% check_function('coeftest') f %>% check_arg('x') %>% check_equal() },{
f <- ex() %>% override_solution('coeftest(lm(medv ~ lstat + crim + age), vcov. = vcovHC)') %>% check_function('coeftest') f %>% check_arg('x') %>% check_equal() })

test_student_typed('vcov. = vcovHC')

test_predefined_objects('R2_res') test_object('R2_unres')

test_or({ test_student_typed("R2_unres > R2_res") },{ test_student_typed("R2_unres < R2_res") },{ test_student_typed('R2_res < R2_unres') },{ test_student_typed('R2_res > R2_unres') })

success_msg("Right. Remember that R2 generally grows when a regressor is added to the model but this does not mean that the model improves upon a model with less regressors in any case.")

3. A Multiple Regression Model of Housing Prices II

Look at the regression equation below describing the previously estimated model with White standard errors in parantheses.

$$\widehat{medv}_i = \underset{(0.74)}{32.828} \underset{(0.08)}{-0.994} \times lstat_i \underset{(0.03)}{-0.083} \times crim_i + \underset{(0.02)}{0.038} \times age_i$$

This model is saved in bh_mult_mod which is available in the working environment.

Instructions

As has been stressed in Chapter 6.3, it is not meaningful to use R^2 when comparing regression models with a different number of regressors. Instead, the $\overline{R}^2$ should be used. $\overline{R}^2$ adjusts for the circumstance that the SSR reduces when a regressor is added to the model.

- Use the model object to compute the correction factor $CF = \frac{n-1}{n-k-1}$ where n is the number of observations and k is the number of regressors, excluding the intercept. Save it to CF.
- Use summary() to obtain R^2 and $\overline{R}^2$ for bh_mult_mod. Is is sufficient if you print both values to the console.
- Check that
$$\overline{R}^2 = 1 - (1 - R^2) \cdot CF$$
. Use the == operator.

library(AER) library(MASS) data("Boston") bh_mult_mod <- lm(medv ~ lstat + crim + age, data = Boston) # Correction Factor n <- k <- CF <-

Obtain R^2 and the adj. R^2

Check that the adj. R^2 can be computed as stated above

Correction Factor n <- nrow(bh_mult_mod$model)$k < -ncol(bh_multmod$model)-1 CF <- (n-1)/(n-k-1)

Obtain R^2 and the adj. R^2 summary(bh_mult_mod)$r.squared$summary(bh_multmod)adj.r.squared

Check that the adj. R^2 can be computed as stated above 1 - (1 - summary(bh_mult_mod)$r.squared) * $CF == summary(bh_multmod)$adj.r.squared

test_predefined_objects("bh_mult_mod") test_object("n") test_object("k") test_object("CF") test_output_contains('summary(bh_mult_mod)$r.squared')$test_output_contains('summary(bh_multmod)adj.r.squared') ex() %>% check_operator("==") %>% check_result() %>% check_equal() success_msg("Well done!")

4. A Fully Fledged Model for Housing Values?

Have a look again at the description of the variables contained in the Boston data set. Which variable would you expect to have the highest p-value in a multiple regression model which uses *all* remaining variables as regressors to explain medv?

Instructions

- Regress medv on all remaining variables that you find in the Boston data set.
- Obtain a heteroskedasticity robust summary of the coefficients.
- The $\overline{R}^2$ for the model in exercise 3 is 0.5533. What can you say about the $\overline{R}^2$ of the large regression model? Does this model improve on the previous one? (no code submission needed)

The packages AER and MASS as well as the data set Boston are loaded to the working environment.

Hints

- For brevity, use the regression formula medv ~. in your call of lm(). This is a shortcut that specifies a regression of medv on all the remaining variables in the data set supplied to the argument data.
- Use summary on both models for a comparison of the $\overline{R}^2$.

library(AER) library(MASS) data(Boston) # Run a regression of 'medv' on all remaining variables in the 'Boston' data set

Obtain a robust summary of the coefficients

What is the model's adj. R^2? # Run a regression of 'medv' on all remaining variables in the 'Boston' data set full_mod <- lm(medv ~., data = Boston)

obtain a robust summary of the coefficients coeftest(full_mod, vcov. = vcovHC(full_mod, type = "HC0"))

What is the model's adj. R^2? summary(full_mod)$adj.r.squared test_object("full_mod") test_function("coeftest", args="x") test_student_typed('vcov. = vcovHC') success_msg("Right. Notice that the smallest p-value is associated with the coefficient on ptratio, the pupil-teacher ratio by town. It is conceivable that quality of the schooling district is an important location factor. According to the adj. R^2 , the full model does better than the model dealt with in exercises 2 and 3 which uses a smaller subset of the variables as regressors.")

5. Model Selection

Maybe we can improve the model by dropping a variable?

In this exercise, you have to estimate several models, each time dropping one of the explanatory variables used in the large regression model of exercise 4 and compare the models' $\overline{R}^2$

The full regression model from the previous exercise, full_mod, is available in your environment.

Instructions

- You are totally free to choose a way to solve this. We recommend the following approach:

1. Start by estimating a model mod_new, say, where e.g. lstat is excluded from the explanatory variables. Next, access this model's $\overline{R}^2$.
2. Compare the model's $\overline{R}^2$ to the $\overline{R}^2$ of the full model (this was about 0.7338).
3. Repeat Steps 1 and 2 for all explanatory variables used in the full regression model. Save the model with the highest improvement in $\overline{R}^2$ to better_mod.

library(AER) library(MASS) data(Boston) full_mod <- lm(medv ~., data = Boston) # Find the model which fits the data better than 'full_mod'

This solution is a bit technical but efficient

Loop estimation of models l <- list() for (i in 1:13) { d <- Boston[,-i] # save each adj. R^2 as a list entry in l l[[i]] <- summary(lm(medv ~., data=d))$adj.r.squared }

assign variable names to the list entries names(l) <- names(Boston[,1:13])

select the variable whose omission leads to the highest improvement in adj. R^2 which.max(l) # 7th column this is "age"

hence a model that fits the data better is better_model <- lm(medv ~., data = Boston[,-7]) test_object("better_model") success_msg("Correct. Notice that this model fits the data better than the full model and does better than all other models that include 12 of the available set of 13 regressors, in

terms of adj. R^2. However, the increase in adj. R^2 is very small and the model is not 'the best' of all conceivable models.")

Chapter 8

Hypothesis Tests and Confidence intervals in Multiple Regression

This chapter discusses methods that allow to quantify the sampling uncertainty inherent to the OLS estimator of coefficients in multiple regression models. The basis for this are standard errors, 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 hypothesis on coefficients of multiple regression models.

8.1 Hypothesis Tests and Confidence Intervals for a Single Coefficient

First, we will 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,
$$t = \frac{\hat{\beta}_j - \beta_{j,0}}{SE(\hat{\beta}_j)}$$
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 statistical software, e.g. by `summary()`.

It is straightforward to verify that principles of testing single hypothesis about the significance of coefficients in the multiple regression model are just 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 6. Let us review this:

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

```
##
## Call:
## lm(formula = score ~ size + english, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.845 -10.240  -0.308   9.815  43.461
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 686.03224    7.41131  92.566  < 2e-16 ***
## size         -1.10130    0.38028  -2.896  0.00398 **
## english      -0.64978    0.03934 -16.516  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.46 on 417 degrees of freedom
## Multiple R-squared:  0.4264, Adjusted R-squared:  0.4237
## F-statistic:   155 on 2 and 417 DF,  p-value: < 2.2e-16
```

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 coeffiecient β_0, that is the coefficient on `(intercept)`, is zero to be approximatley zero.

```
2*(1-pnorm(abs(92.566)))
```

```
## [1] 0
```

Remember that, given a vector of quantiles, `pnorm()` calculates associated probabilities for the standard normal distribution by default which is suitable here since we approximate with the standard normal distribution.

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 all randomly drawn 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

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

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

Let us take a look at the regression from section 6.3 again.

Computing individual confidence intervals for the coefficients in the multiple regression model can be done analogously 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
```

We note that 95% confidence intervals for all three coefficients are computed.

If we want to compute confidence intervals at another level of $\alpha = 0.1$ say, we just have to 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 model coefficients.

Knowing how to use 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 outputs above tell us that zero is not an element of the computed confidence intervals for the coefficient of `size` such that we can reject the null hypothesis at significance levels of 5% and 10%.

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

The same conclusion can be made when beholding the p-value for `size`: $0.00398 < 0.05 = \alpha$.

The 95% confidence interval tells us that we can be 95% confident 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 -1.8488 and an upper bound of -0.3538.

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?

We can pursue this question by augmenting our model equation by an additional regressor that is a measure for spendings per pupil. Using `?CASchools` we find that `CASchools` contains the variable expenditure which provides expenditures per student.

The model we want the estimate now is

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

with *expenditure* the total amount of expenditures 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)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ size + english + expenditure, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.340 -10.111   0.293  10.318  43.181
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 649.57795   15.20572  42.719  < 2e-16 ***
## size         -0.28640    0.48052  -0.596  0.55149
## english      -0.65602    0.03911 -16.776  < 2e-16 ***
## expenditure   3.86790    1.41212   2.739  0.00643 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.35 on 416 degrees of freedom
## Multiple R-squared:  0.4366, Adjusted R-squared:  0.4325
## F-statistic: 107.5 on 3 and 416 DF,  p-value: < 2.2e-16
```

We see that the estimated effect of a one unit change in the student-teacher ratio on test scores with expenditures per pupil and the share of english learning pupils held constant is rather small (-0.29). What is more, the coefficient on *size* is not significantly different from zero anymore even at the level of 10% since p-value $= 0.55$. Can You come up with an interpretation for these findings (see chapter 7.1 of the book)? Mathematically, the insignificance of β_1 could be due to a larger standard error of $\hat{\beta}_1$ resulting from adding *expenditure* to the model so that we are estimating the true coefficent on *size* less precisely. This illustrates the issue of strongly correlated regressors (imperfect multicollinearity). The correlation between *size* and *expenditures* can be computed using `cor()`.

```
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 at the same time.

8.3 Joint Hypothesis Testing Using the F-Statistic

The estimated model is

$$\widehat{TestScore} = 649.58 - 0.29 \times size - 0.66 \times english + 3.87 \times expenditure.$$

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

The homoskedasticity-only F-Statistic is given by

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

with $SSR_{restricted}$ 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 restriction under the null and k is the number of regressors in the unrestricted regression.

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

```
# 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
```

From the output we can infer 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.

Note:

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 pair is

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

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 will now check that 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 linear restriction
# to be tested as strings
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 F-statistic from the model's summary
summary(model)$fstatistic
```

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

The test rejects the null hypothesis that the model has no power in explaining test scores rather clearly.

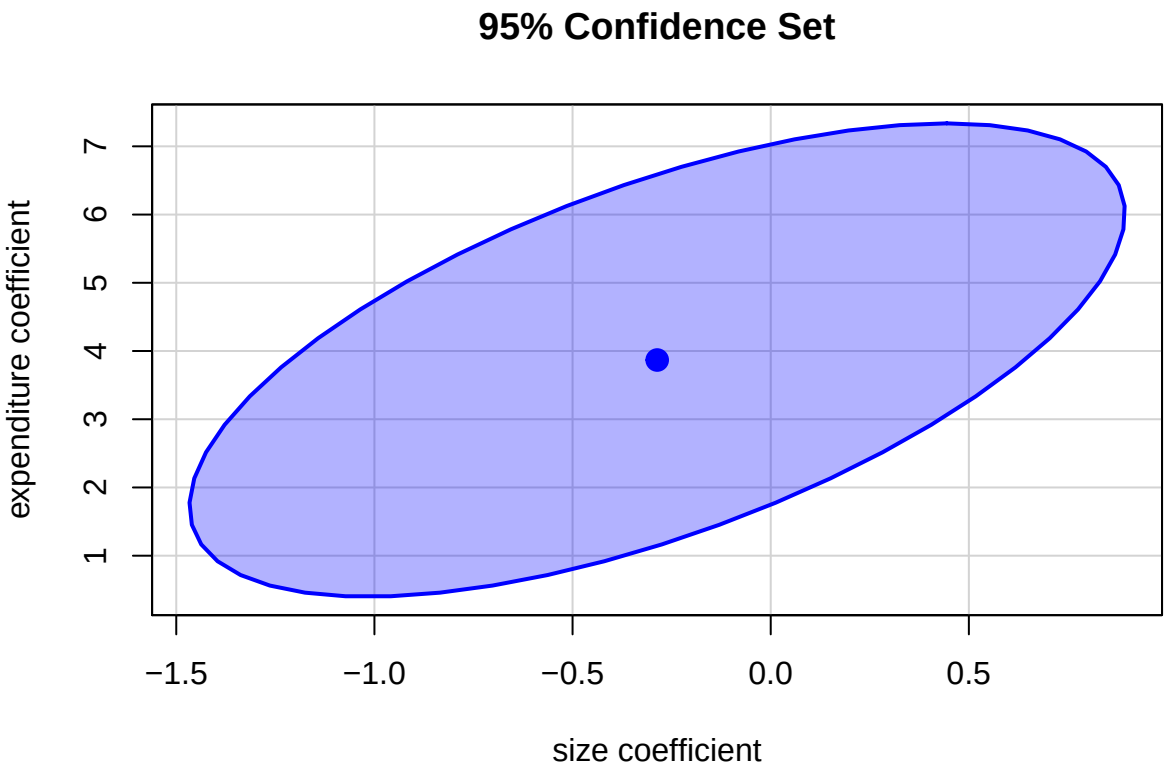
8.4 Confidence Sets for Multiple Coefficients

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, 95% say, of all cases if we could infinitely draw random samples, just as in the univariate case. Put differently, a confidence set is the set of coefficient combinations for which we cannot reject a joint null hypothesis tested using a F-test.

If we consider two coefficients, their confidence set is 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()` which is also coming with the `car` package.

In the following, we 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 which gives us a better impression which set of coefficients is meant.

```
# 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 $(-0.29, 3.87)$ i.e. 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 $H_0 : \beta_1 = 0,\ \beta_3 = 0$.

8.5 Model Specification for Multiple Regression

Choosing a regression specifaction i.e. selecting the variables to be included in a regression model can be quite cumbersome. However, there are some guidelines how to do this. The goal is clear: obtaining an unbiased estimation of the causal effect of interest. As a starting point, one should think about omitted variables, that is to avoid a possible estimation 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 regression model 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 one or more included regressors are correlated 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 will 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 estimated the causal effect of class size on test score. In this estimation, there might be a bias due to omitting "outside learning opportunities" from our regression sice a measure like this 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" is a complicated concept that is difficult to quantify. A surrogate we can consider instead is the students' economic backgroud which should be strongly related to the outside learning opportunities: think of wealthy parents that are able to provide time and/or money for private tuition of there 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 estimate the model again.

```
# estimate the model and print summary to console
model <- lm(score ~ size + english + lunch, data = CASchools)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ size + english + lunch, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.849  -5.151  -0.308   5.243  31.501
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 700.14996    4.68569 149.423  < 2e-16 ***
## size         -0.99831    0.23875  -4.181 3.54e-05 ***
## english      -0.12157    0.03232  -3.762 0.000193 ***
## lunch        -0.54735    0.02160 -25.341  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.08 on 416 degrees of freedom
## Multiple R-squared:  0.7745, Adjusted R-squared:  0.7729
## F-statistic: 476.3 on 3 and 416 DF,  p-value: < 2.2e-16
```

Thus, the estimated regression line is

$$\widehat{TestScore} = \underset{(4.7)}{700.15} - \underset{(0.24)}{1.00} \times size - \underset{(0.032)}{0.12} \times english + \underset{(0.022)}{0.55} \times lunch.$$

We observe no substantial changes in the conclusion about the effect of `size`: the coefficient changes by only 0.1 and keeps its significance.

Although the difference in estimated coefficients is not big in this case, it might be a good idea to keep `lunch` in the model 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

R^2 and $\overline{R^2}$: What They Tell You — and What They Don't

The R^2 and $\overline{R^2}$ tell you whether the regressors are good at predicting, or "explaining" the values of the independent variable in the sample of data at hand. if the R^2 (or $\overline{R^2}$) is nearly 1, then the regressors produce 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.

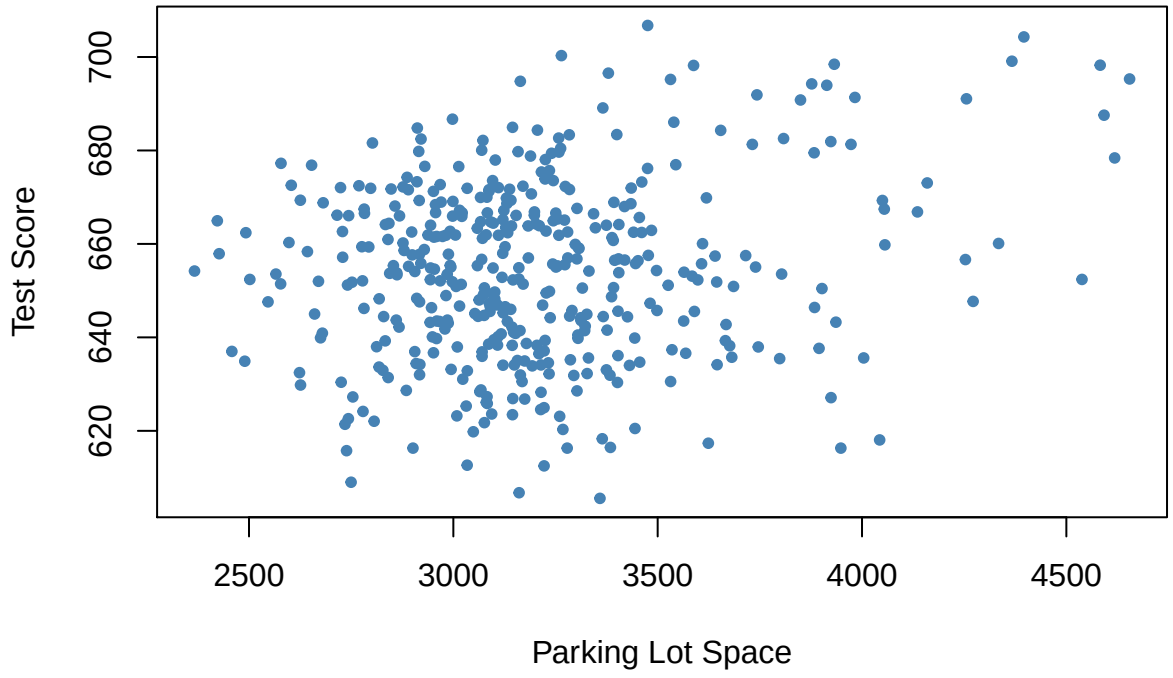
The R^2 and $\overline{R^2}$ do not tell you whether:

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

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

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.

```
# Generate observations for parking lot space
CASchools$PLS <- 5 + 0.6*CASchools$expenditure + 0.6*CASchools$income + CASchools$size/100 + rnorm(nrow

# 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
## -50.16 -14.25   0.65  13.56  49.88
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.224e+02  7.715e+00  80.679  < 2e-16 ***
## PLS         9.915e-03  2.393e-03   4.144 4.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.7 on 418 degrees of freedom
## Multiple R-squared:  0.03946,    Adjusted R-squared:  0.03717
## F-statistic: 17.17 on 1 and 418 DF,  p-value: 4.131e-05
```

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

$$\widehat{TestScore} = \beta_0 + \beta_1 \times PLS + u \tag{8.1}$$

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 0.47 which is a lot more than the roughly 0.05 observed when regressing test score on class size only.

This suggests that increasing the parking lot space boosts a school's test scores and that model (8.1) 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 of 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 determinantes and/or control variables. Increasing parking lot space is not an appropriate measure to generate more learning success!

8.6 Analysis of the Test Score Data Set

Chapter 6 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 that cannot be measured directly. Recall that in our example of test scores, we are interested in estimating the causal effect of a change in the student-teacher ratio on test scores. In what follows, we will provide an example how to use multiple regression models in order to alleviate omitted variable bias and we will 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 programm. Students eligible for CalWorks live in families with a total income below the threshold for the subsidized lunch programm so both variables are indicators for the share of economically disadvantaged children. Using R we can confirm that both indicators are highly correlated:

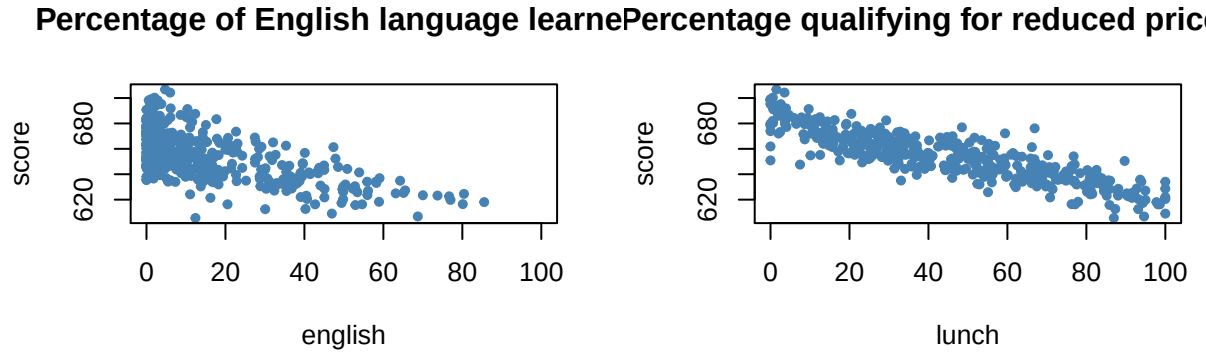
```
# estimate correlation between calworks and lunch
cor(CASchools$calworks, CASchools$lunch)
```

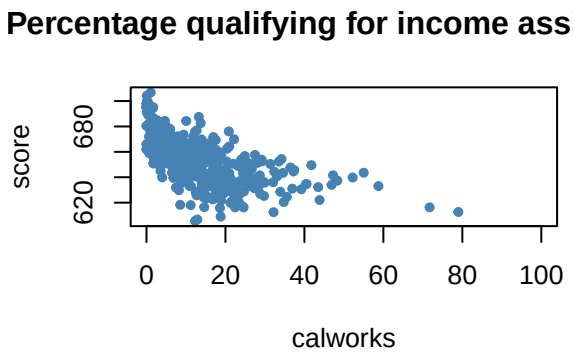
```
## [1] 0.7394218
```

There is no unabigious way to proceed when deciding which variable to use. In any case it is not a good idea to use both variables as regressors having in mind consequences of colinearity. Therefore, we will also consider alternative model specifications.

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

```
par(mfrow = c(1,3))

m <- rbind(c(1, 2), c(3, 0))
layout(m)

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

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

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





We see that all relationships are negative. We can use R to estimate 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 will consider 5 different model equations:

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

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 scienctific standards. One simply has to provide one or multiple object(s) of class `lm` and 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)

# generate a Latex table using stargazer
stargazer(spec1, spec2, spec3, spec4, spec5,
          column.labels = c("(I)", "(II)", "(III)", "(IV)", "(V)")
          )
```

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Thu, Apr 26, 2018 - 10:53:51 % Requires LaTeX packages: rotating

The table states that *scores* is the dependent variable and that we consider 5 models. We see that the coloumns of Table 7.1 contain all the information provided by `summary()` for the regression models `spec1` to `spec5`: the coefficient section of the table presents coefficients estimates equipped with signifiance codes (asterisks) and standard errors in parantheses below. Although there a no t-statistics, it is straightforward for the reader to compute them simply by dividing a coefficient estimate by the correspondung standard error. In the botton of the table we find 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?

1. We see that adding control variables roughly halves the coefficient on `size`. Also the coefficient 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. Morever, the estimated coefficients on all control variables are consistent with the impressions gained from figure 7.2.
3. We see that the control variables are not always individually statistically significant: for example in `spec5` we see that the coefficient on *calworks* is not significantly different from zero at the level of 5% since $|-0.048/0.061| = 0.79 < 1.64$. 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 setting.

8.7 Exercises

Inference in the Multiple Regression Model – t-statistics

Instructions

Hint

library(MASS) data("Boston") mod <- lm(medv ~ lstat + age + crim, data = Boston)

Table 8.1:

	Dependent variable:				
	score				
	(I)	(II)	(III)	(IV)	(V)
	spec1	spec2	spec3	spec4	spec5
size	−2.280***	−1.101***	−0.998***	−1.308***	−1.014***
	(0.480)	(0.380)	(0.239)	(0.307)	(0.240)
english		−0.650***	−0.122***	−0.488***	−0.130***
		(0.039)	(0.032)	(0.033)	(0.034)
lunch			−0.547***		−0.529***
			(0.022)		(0.032)
calworks				−0.790***	−0.048
				(0.053)	(0.061)
Constant	698.933***	686.032***	700.150***	697.999***	700.392***
	(9.467)	(7.411)	(4.686)	(6.024)	(4.698)
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

Chapter 9

Nonlinear Regression Functions

Until now we assumed the regression function to be linear, i.e. we have treated the slope of the regression function as a constant. This implies that the effect on Y of a one unit change in X does not depend on the level of X. If however the effect of a change in X on Y does depend on the value of X, we have to use a nonlinear regression function.

9.1 A General Strategy for Modeling Nonlinear Regression Functions

Let us have a look at an example where in fact using a nonlinear regression function is better suited for estimation of the population relationship between the regressor, X, and the regressand, Y: the relationship between the income of schooling districts and their test scores.

```
#Prepare the data
library(AER)
data(CASchools)
CASchools$size <- CASchools$students/CASchools$teachers
CASchools$score <- (CASchools$read + CASchools$math)/2
```

We start our analysis by computing the correlation between the two variables.

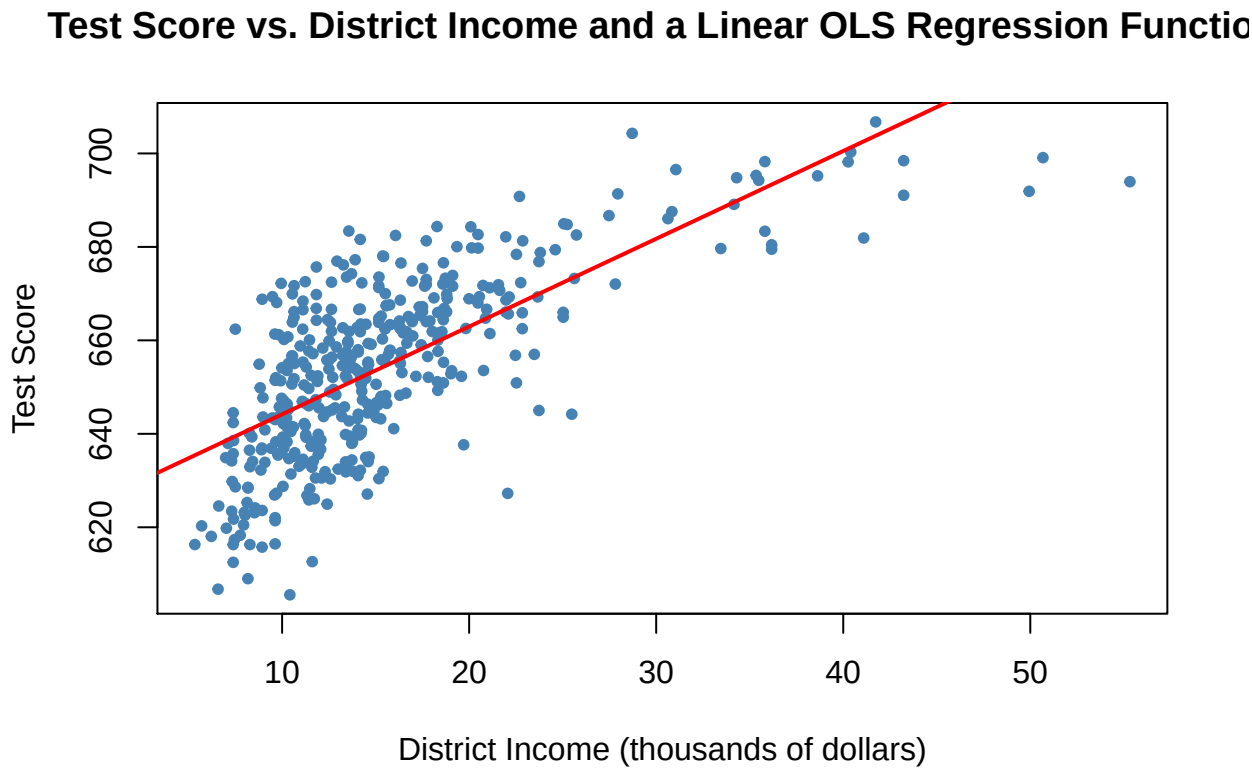
```
cor(CASchools$income, CASchools$score)
```

```
## [1] 0.7124308
```

The correlation coefficient is about 0.71. This means that income and test scores are positively correlated. In other words, children whose parents have an above average income tend to achieve above average test scores. Can we use the correlation coefficient to assess whether a linear regression model does fit the data adequately? To answer this question we visualize the data and add a linear regression line to the plot.

```
#Fit linear model
linear_model<- lm(score ~ income, data = CASchools)

# Plot observations
plot(CASchools$income, CASchools$score,
     col = "steelblue",
     pch = 20,
     xlab = "District Income (thousands of dollars)",
     ylab = "Test Score",
```

```
     main = "Test Score vs. District Income and a Linear OLS Regression Function")

# Add the regression line to the plot
abline(linear_model, col="red", lwd=2)
```



As Stock and Watson point out, the linear regression line seems to overestimate the true relationship when income is very high or very low and underestimates it in the midrange.

Fortunately, usage of the OLS is not restricted to linear functions of the regressors. Thus we can for example model test scores as a function of income and the square of income. The corresponding regression model is

$$TestScore_i = \beta_0 + \beta_1 Income_i + \beta_2 Income_i^2 + u_i.$$

This equation is called the *quadratic regression model*. Note, that $Income^2$ is treated as an additional explanatory variable. Hence, the quadratic model is a special case of a multivariate regression model. When fitting the model with `lm()` we have to use the `^` operator in conjunction with the function `I()` to add the quadratic term as an additional regressor to the `formula` argument.

```
# Fit the quadratic Model
quadratic_model <- lm(score ~ income + I(income^2), data = CASchools)

# Generate model summary
summary(quadratic_model)
```

```
##
## Call:
## lm(formula = score ~ income + I(income^2), data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -44.416  -9.048   0.440   8.347  31.639
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 607.30174    3.04622 199.362  < 2e-16 ***
## income        3.85099    0.30426  12.657  < 2e-16 ***
## I(income^2)  -0.04231    0.00626  -6.758 4.71e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.72 on 417 degrees of freedom
## Multiple R-squared:  0.5562, Adjusted R-squared:  0.554
## F-statistic: 261.3 on 2 and 417 DF,  p-value: < 2.2e-16
```

The output tells us that the estimated model equation is

$$\widehat{TestScore}_i = \underset{(3.05)}{607.3} + \underset{(0.30)}{3.85} \times Income_i - \underset{(0.01)}{0.0423} \times Income_i^2.$$

Notice that this estimated model equation allows us to test the hypothesis that the relationship between test scores and district income is linear against the alternative hypothesis that it is nonlinear. This corresponds to testing

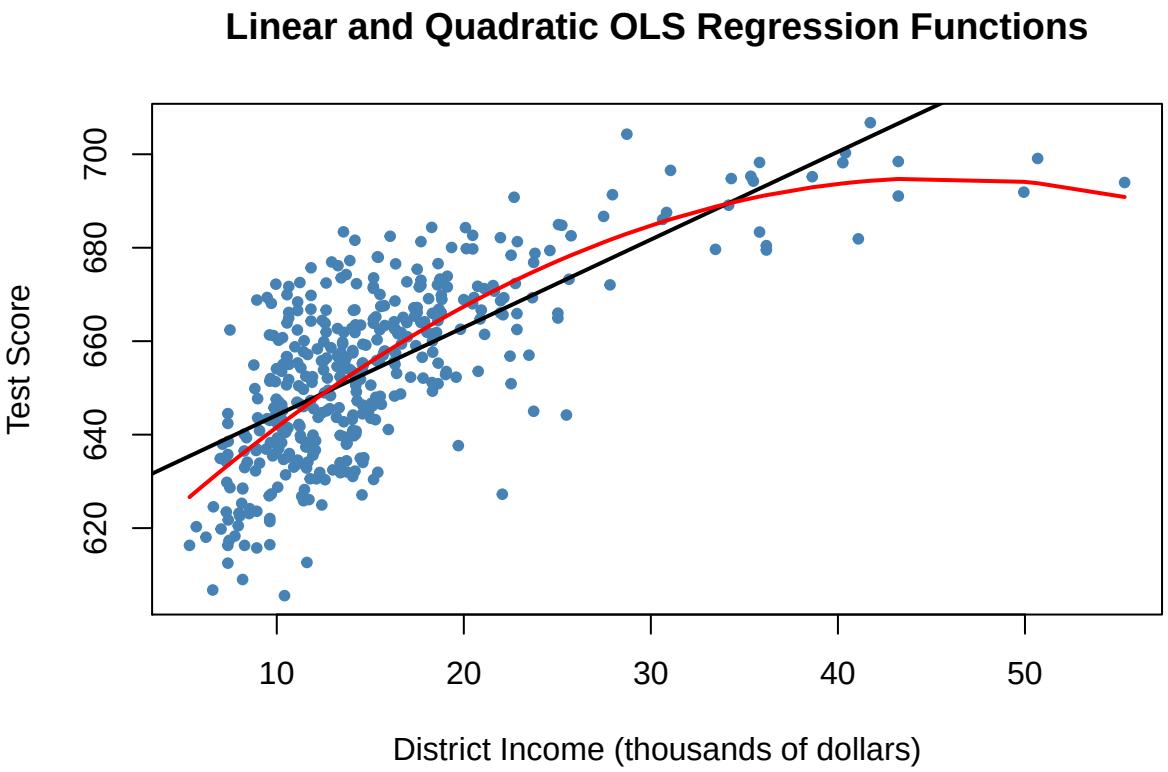
$$H_0 : \beta_2 = 0 \quad \text{vs.} \quad H_1 : \beta_2 \neq 0,$$

since $\beta_2 = 0$ corresponds to a simple linear equation and $\beta_2 \neq 0$ implies a quadratic relationship. We find that $t = (\hat{\beta}_2 - 0)/SE(\hat{\beta}_2) = -0.0423/0.01 = -4.23$ so the null is rejected at any common level of significance and we conclude that the relationship is nonlinear. This is consistent with our informal inspection of the plotted data.

We can now draw the same scatterplot as for the linear model and add the regression line for the quadratic model. Because `abline()` can only draw straight lines, it cannot be used for this task. A function which can be used to draw lines without being restricted to straight lines is `lines()`, see `?lines`. The most basic call of `lines()` is `lines(x_values, y_values)` where `x_values` and `y_values` are vectors of the same length that provide coordinates of the points to be sequentially connected by a line. This makes it necessary to sort the coordinate pairs according to the X-values. Otherwise You will not get the desired result! In this example we use the function `order()` to sort the fitted values of *TestScore* according to the observations for *income*.

```
# Scatterplot of observatuib for income and TestScore
plot(CASchools$income, CASchools$score,
     col  = "steelblue",
     pch = 20,
     xlab = "District Income (thousands of dollars)",
     ylab = "Test Score",
     main = "Linear and Quadratic OLS Regression Functions")

#Add linear function to the plot
abline(linear_model , col="black", lwd=2)

#Add quatratic function to the plot
order_id <- order(CASchools$income)

lines(x = CASchools$income[order_id],
      y = fitted(quadratic_model)[order_id],
```

```
      col="red",
      lwd=2)
```



We see that the quadratic model does much better in fitting the data than the linear model.

9.2 Nonlinear Functions of a Single Independent Variable

Polynomials

The approach used to obtain a quadratic model can be generalized to polynomial models of arbitrary order r.

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \ldots + \beta_r X_i^r + u_i$$

A cubic model ($r = 3$) for instance can be estimated in the same way as the quadratic model — we just have to add `I(income^3)` to the `formula` argument in our call of `lm()`.

```
cubic_model <- lm(score ~ income + I(income^2) + I(income^3), data = CASchools)
```

In practice the question will arise which polynomial order should be chosen. First note that, similarly as for $r = 2$, we can test the null hypothesis that the true relation is linear against the alternative hypothesis that the relationship is a polynomial of degree r:

$$H_0 : \beta_2 = 0,\ \beta_3 = 0, \ldots, \beta_r = 0 \quad \text{vs.} \quad H_1 : \text{at least one } \beta_j \neq 0,\ j = 2, \ldots, r$$

This is a joint null hypothesis with $r - 1$ restrictions so it can be tested using the F-test presented in chapter 7. Remember that the function `linearHypothesis()` can compute such test statistics. If, for example, we would like to test the null hypothesis of a linear model against the alternative of a polynomial of a maximal degree $r = 3$ we could simply do the following:

```
library(car)

# test the hypothesis
linearHypothesis(cubic_model,
                 c("I(income^2)=0", "I(income^3)=0")
)
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ income + I(income^2) + I(income^3)
##
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    418 74905
## 2    416 67170  2    7735.5 23.954 1.424e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value for this test is very small so that we reject the null hypothesis. However, this does not tell us which r to choose. In practice, one approach to determine degree of the polynomial is to use sequential testing:

1. Estimate a polynomial model for some maximum value r.
2. Use a t-test to test whether $\beta_r = 0$. Rejection of the null means that X^r belongs in the regression equation.
3. Acceptance of the null in step 2 means that X^r can be eliminated from the model. Continue by repeating step 1 with order $r-1$ and test whether $\beta_{r-1} = 0$. If the test rejects, use a polynomial model of order $r-1$.
4. If the tests from step 3 rejects, continue with the procedure until the coefficient on the highest power is statistically significant.

There is no unambigous guideline how to choose r in step one. However as Stock and Watson point out, economic data is often smooth such that it is appropriate to choose small orders like 2, 3, or 4.

We will now demonstrate how to apply sequential testing in the example of a cubic model.

```
summary(cubic_model)
```

```
##
## Call:
## lm(formula = score ~ income + I(income^2) + I(income^3), data = CASchools)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -44.28  -9.21   0.20   8.32  31.16
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.001e+02  5.830e+00 102.937  < 2e-16 ***
## income       5.019e+00  8.595e-01   5.839 1.06e-08 ***
## I(income^2) -9.581e-02  3.736e-02  -2.564   0.0107 *
## I(income^3)  6.855e-04  4.720e-04   1.452   0.1471
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.71 on 416 degrees of freedom
## Multiple R-squared:  0.5584, Adjusted R-squared:  0.5552
## F-statistic: 175.4 on 3 and 416 DF,  p-value: < 2.2e-16
```

The estimated cubic model stored in `cubic_model` is

$$\widehat{TestScore}_i = \underset{(5.83)}{600.1} + \underset{(0.86)}{5.02} \times Income + \underset{(0.03)}{-0.96} \times Income^2 + \underset{(-0.00047)}{0.00069} \times Income^3.$$

Summary tells us that the t-statistic on $Income^3$ is 1.42 so the null that the relationship is quadratic cannot be rejected, even at the 10% level. This is contrary to the result of the book which reports robust standard errors throughout so we will also use robust variance-covariance estimation to reproduce these results.

```
# load the lmtest package for coeftest()
library(lmtest)

# test the hypothesis using robust standard errors
coeftest(cubic_model, vcov. = vcovHC(cubic_model, type = "HC1"))
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)  6.0008e+02  5.1021e+00 117.6150 < 2.2e-16 ***
## income       5.0187e+00  7.0735e-01   7.0950 5.606e-12 ***
## I(income^2) -9.5805e-02  2.8954e-02  -3.3089  0.001018 **
## I(income^3)  6.8549e-04  3.4706e-04   1.9751  0.048918 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the reported standard errors have changed. Furthermore, the coefficient for `income^3` is now significant at the 5% level. This means we reject the hypothesis that the regression function is quadratic against the alternative that it is cubic. Furthermore, we can also test if the coefficients for `income^2` and `income^3` are jointly significant using a robust version of the F-test.

```
# robust F-test for
linearHypothesis(cubic_model,
                 vcov. = vcovHC(cubic_model, type = "HC1"),
                 c("I(income^2)=0", "I(income^3)=0")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ income + I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    418
```

```
## 2    416  2 37.691 9.043e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With a p-value of `9.043e-16`, i.e. much less than 0.05, the null hypothesis of linearity is rejected in favour of the alternative that the relationship is quadratic or cubic.

Interpretation of Coefficients in Nonlinear Regression Models

The coefficients in polynomial regression do not have a simple interpretation. Think of a quadratic model. It is not helpful to think of the coefficient on X as the expected change in Y associated with a change in X holding the other regressors constant because one other is X^2 which changes as X is varied. This is also the case for other deviations from linearity, for example in models where regressors and/or the dependent variable are log-transformed. The best way to approach this is to calculate the estimated effect on Y associated with a change in X for one or more values of X. This idea is summarized in Key Concept 8.1.

Key Concept 8.1

The Expected Effect on Y of a Change in X_1 in a Nonlinear Regression Model

Consider the nonlinear population regression model

$$Y_i = f(X_{1i}, X_{2i}, \dots, X_{ki}) + u_i \ , \ i = 1, \dots, n,$$

where $f(X_{1i}, X_{2i}, \dots, X_{ki})$ is the population regression function and u_i is the error term.

The expected change in Y, ΔY, associated with the change in X_1, ΔX_1, holding $X_2, \cdots, X_k$ constant. That is, the expected change in Y is the difference:

$$\Delta Y = f(X_1 + \Delta X_1, X_2, \cdots, X_k) - f(X_1, X_2, \cdots, X_k).$$

The estimator of this unknown population difference is the difference between the predicted values for these two cases. Let $\hat{f}(X_1, X_2, \cdots, X_k)$ be the predicted value of of Y based on the estimator $\hat{f}$ of the population regression function. Then the predicted change in Y is

$$\Delta \hat{Y} = \hat{f}(X_1 + \Delta X_1, X_2, \cdots, X_k) - \hat{f}(X_1, X_2, \cdots, X_k).$$

For example, we may ask the following: what is the predicted change in test scores associated with a one unit change (i.e. \$1000), based on the estimated quadratic regression function

$$\widehat{TestScore} = 607.3 + 3.85 \times Income - 0.0423 \times Income^2 \ ?$$

Because the regression function is quadratic, this effect depends on the initial district income. We therefore consider two cases: an increase in district income form 10 to 11 (i.e. from \$10000 per capita to \$11000) and an increase in district income from 40 to 41 (that is from \$40000 to \$41000).

In order to obtain the $\Delta \hat{Y}$ associated with a change in income form 10 to 11, we use the following formula:

$$\Delta \hat{Y} = \left(\hat{\beta}_0 + \hat{\beta}_1 \times 11 + \hat{\beta}_2 \times 11^2 \right) - \left(\hat{\beta}_0 + \hat{\beta}_1 \times 10 + \hat{\beta}_2 \times 10^2 \right)$$

To compute $\hat{Y}$ using R we may use `predict()`.

```
# compute and assign the quadratic model
quadriatic_model <- lm(score ~ income + I(income^2), data = CASchools)

# set up data to predict
new_data <- data.frame(income = c(10, 11))

# do the prediction
Y_hat <- predict(quadriatic_model, newdata = new_data)

# compute the difference
diff(Y_hat)
```

```
##        2
## 2.962517
```

Analogously we can compute the effect of a change in *income* from 40 to 11:

```
# set up data to predict
new_data <- data.frame(income = c(40, 41))

# do the prediction
Y_hat <- predict(quadriatic_model, newdata = new_data)

# compute the difference
diff(Y_hat)
```

```
##         2
## 0.4240097
```

So for the quadratic model, the expected change in $TestScore$ induced by an increase in *income* from 10 to 11 is about 2.96 points but an increase in *income* from 40 to 41 increases the predicted score by only 0.42. Hence the slope of the estimated quadratic regression function is steeper at low levels of income than at the higher levels.

Logarithms

Another way to specify a nonlinear regression function is to use the natural logarithm of Y and/or X. Logarithms convert changes in variables into percentage changes which is convenient as many relationships are naturally expressed in terms of percentages.

There are three different cases in which logarithms might be used.

1. X could be transformed by taking its logarithm but Y is not.
2. We could transform Y to its logarithm but leave X at level.
3. A third case is that both Y and X are transformed to their logarithms. The interpretation of the regression coefficients is different in each case.

Case I: X is in logarithm, Y is not.

The regression model then is

$$Y_i = \beta_0 + \beta_1 \times \ln(X_i) + u_i,\ i = 1, ..., n.$$

Similar as for polynomial regression we do not have to create a new variable by computing $\ln(X)$. We can simply adjust the `formula` argument of `lm()` to tell R that the log-transformation of a variable should be used.

```
# estimate a level-log model
LinearLog_model <- lm(score ~ log(income), data = CASchools)

# compute robust summary
coeftest(LinearLog_model,
         vcov = vcovHC(LinearLog_model, type = "HC1")
         )
```

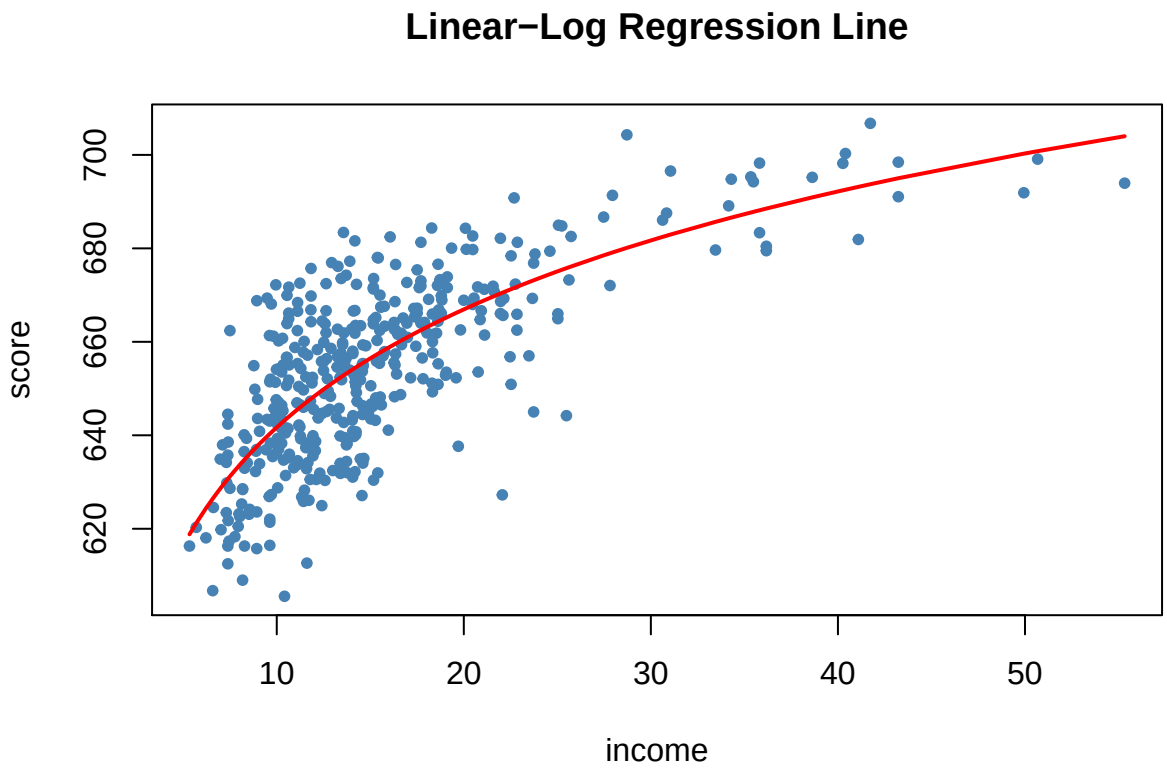
```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 557.8323     3.8399 145.271 < 2.2e-16 ***
## log(income)  36.4197     1.3969  26.071 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

According to the output the estimated regression function is:

$$\widehat{TestScore} = 557.8 + 36.42 \times \ln(Income).$$

Let us draw a plot of this function.

```
# draw scatterplot
plot(score ~ income,
     col = "steelblue",
     pch = 20,
     data = CASchools,
     main = "Linear-Log Regression Line")

# add Linear-Log regression line
order_id  <- order(CASchools$income)

lines(CASchools$income[order_id],
      fitted(LinearLog_model)[order_id],
      col = "red",
      lwd = 2)
```



We can interpret $\hat{\beta}_1$ as follows: a 1% increase in income is associated with an increase in test scores of $0.01 \times 36.42 = 0.36$ points. In order to get the estimated effect of a one unit change in income (that is a change in the original units,thousands of dollars, not in logarithms) on test scores, the method presented in Key Concept 8.1 can be used.

```
# set up new data
new_data <- data.frame(income = c(10, 11, 40, 41))

# predict outcomes
Y_hat <- predict(LinearLog_model, newdata = new_data)

# compute expected difference
changes <- matrix(Y_hat, nrow = 2, byrow = TRUE)
changes[ ,2] - changes[ ,1]
```

```
## [1] 3.471166 0.899297
```

The estimated model states that for an income increase from $\$10,000$ to $\$11,000$, test scores increase by an expected amount of 3.47 points. When income increases from $\$40,000$ to $\$41,000$, the expected increase in test scores is only about 0.90 points.

Case II: Y is in logarithm, X is not

If You want to learn about the absolute impact of an explanatory variable on Your dependent variable, it is not recommended to log-transform the latter. There are, however, cases where we want to learn about $\ln(Y)$ instead of Y.

The corresponding regression then model is

$$\ln(Y_i) = \beta_0 + \beta_1 \times X_i + u_i \quad , \quad i = 1, ..., n.$$

```
# estimate a log-linear model
LogLinear_model <- lm(log(score) ~ income, data = CASchools)

# compute a robust summary
coeftest(LogLinear_model,
         vcov = vcovHC(LogLinear_model, type = "HC1")
         )
```

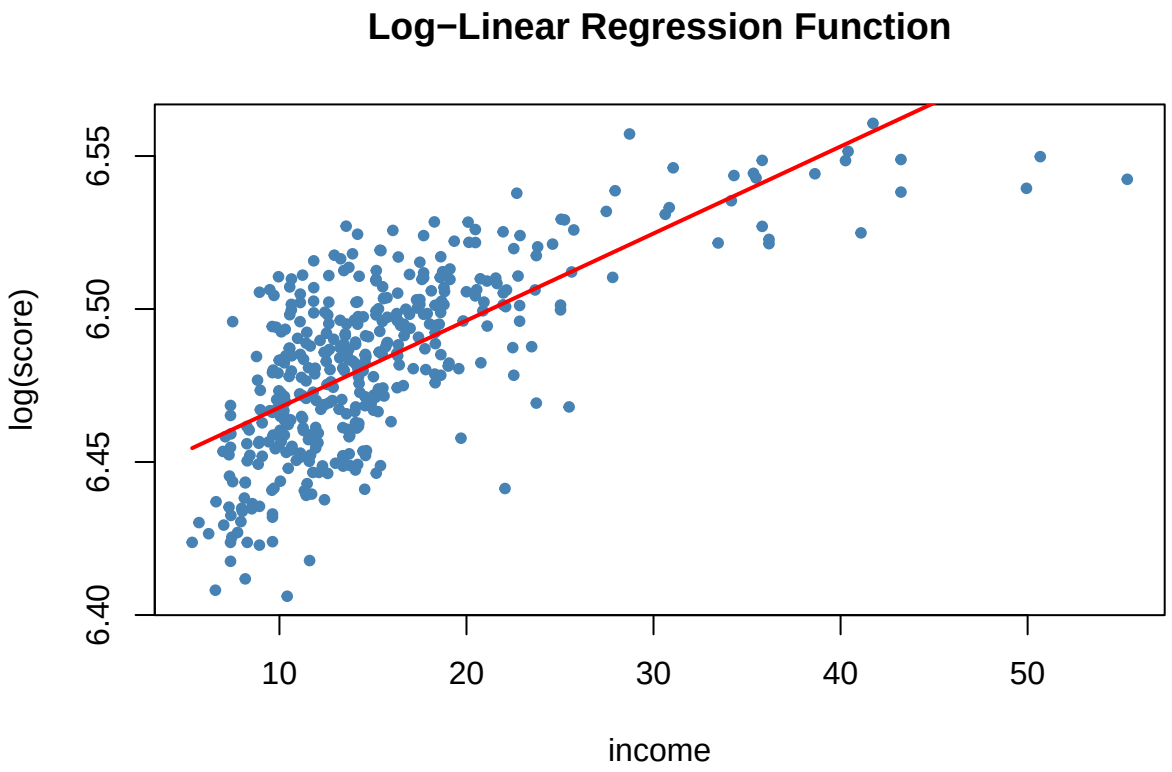
```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 6.43936234 0.00289382 2225.210 < 2.2e-16 ***
## income      0.00284407 0.00017509   16.244 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression function is

$$\widehat{\ln(TestScore)} = 6.439 + 0.00284 \times income.$$

Since we are interested in $\ln(Y)$ rather than Y, we do not retransform the dependent variable.

```
# scatterplot
plot(log(score) ~ income,
     col="steelblue",
     pch=20,
     data = CASchools,
     main = "Log-Linear Regression Function"
     )

# add the Log-Linear regression line
order_id  <- order(CASchools$income)

lines(CASchools$income[order_id],
      fitted(LogLinear_model)[order_id],
      col = "red",
      lwd = 2)
```



Note that the Y-Axis is now log-transformed.

In a log-linear model, a one-unit change in X is associated with an estimated $100 \times \hat{\beta}_1\%$ change in Y. This time we left the X values unchanged.

```
# do predictions
Y_hat <- predict(LogLinear_model, newdata = new_data)

# calculate changes
changes <- matrix(Y_hat, nrow = 2, byrow = TRUE)
changes[ ,2] - changes[ ,1]
```

```
## [1] 0.00284407 0.00284407
```

Case III: X and Y are in logarithms

The log-log regression model is

$$\ln(Y_i) = \beta_0 + \beta_1 \times \ln(X_i) + u_i \quad , \quad i = 1, ..., n.$$

```
# Estimate the log-log model
LogLog_model <- lm(log(score) ~ log(income), data = CASchools)

# print robust summary to the console
coeftest(LogLog_model,
         vcov = vcovHC(LogLog_model, type = "HC1")
         )
```

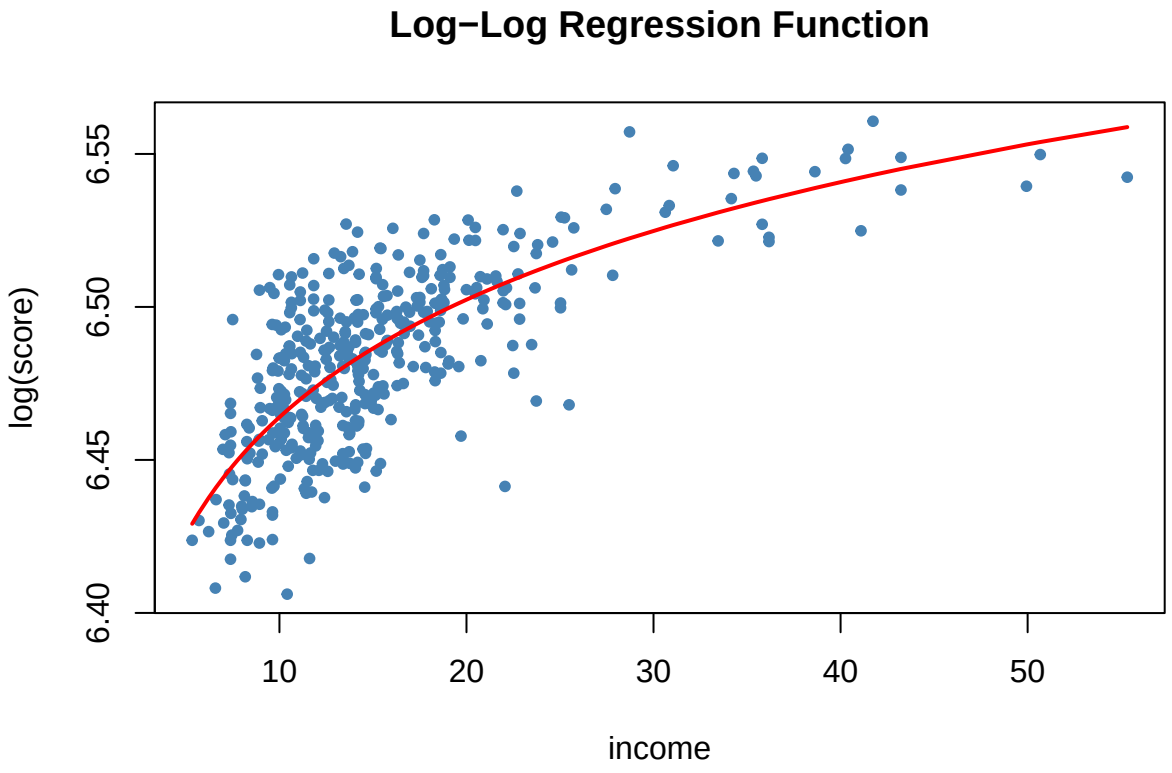
```
##
## t test of coefficients:
```

```
##
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 6.3363494  0.0059246 1069.501 < 2.2e-16 ***
## log(income) 0.0554190  0.0021446   25.841 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression function hence is

$$\widehat{\ln(TestScore)} = 6.336 + 0.0554 \times income.$$

```
# scatterplot
plot(log(score) ~ income,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     main = "Log-Log Regression Function")

# plot the estimate log-log regression line
lines(sort(CASchools$income),
      fitted(LogLog_model)[order(CASchools$income)],
      col = "red",
      lwd = 2)
```



In a log-log model, a 1% change in X is associated with an estimated $\hat{\beta}_1$% change in Y.

```
# predict Y
Y_hat <- predict(LogLog_model, newdata = new_data)

# compute changes
changes <- matrix(Y_hat, nrow = 2, byrow = TRUE)
```

```
changes[ ,2] - changes[ ,1]
```

```
## [1] 0.005281992 0.001368439
```

Key Concept 8.2 summarizes the three logarithmic regression models.

Key Concept 8.2

Logarithms in Regression: Three Cases

Logarithms can be used to transform the dependent variable Y or the independent variable X, or both (the variable being transformed must be positive). The following table summarizes these three cases and the interpretation of the regression coefficient β_1. In each case, β_1, can be estimated by applying OLS after taking the logarithm(s) of the dependent and/or the independent variable.

Case

Model Specification

Interpretation of β_1

(I)

$Y_i = \beta_0 + \beta_1 \ln(X_i) + u_i$

A 1% change in X is associated with a change in Y of $0.01 \times \beta_1$.

(II)

$\ln(Y_i) = \beta_0 + \beta_1 X_i + u_i$

A change in X by one unit ($\Delta X = 1$) is associated with a $100 \times \beta_1\%$ change in Y.

(III)

$\ln(Y_i) = \beta_0 + \beta_1 \ln(X_i) + u_i$

A 1% change in X is associated with a $\beta_1\%$ change in Y, so β_1 is the elasticity of Y with respect to X.

Of course we can also estimate a polylog model like

$$TestScore_i = \beta_0 + \beta_1 \times \ln(income_i) + \beta_2 \times \ln(income_i)^2 + \beta_3 \times \ln(income_i)^3 + u_i$$

which models the dependent variable $TestScore$ by a third-degree polynomial of the log-transformed regressor $income$

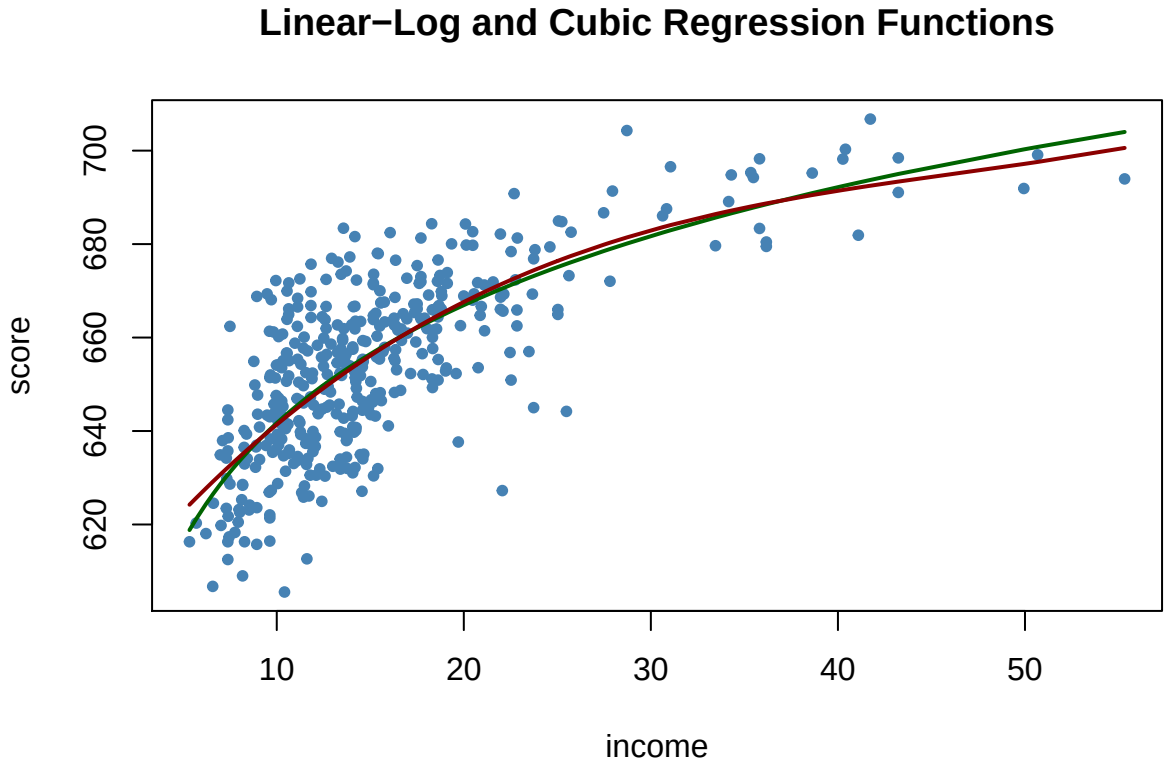
```
# estimate the polylog model
polyLog_model <- lm(score ~ log(income) + I(log(income)^2) + I(log(income)^3),
                    data = CASchools)

# print robust summary to the console
coeftest(polyLog_model,
         vcov = vcovHC(polyLog_model, type = "HC1"))
```

```
##
## t test of coefficients:
##
##                  Estimate Std. Error t value  Pr(>|t|)
## (Intercept)      486.1341    79.3825  6.1239 2.115e-09 ***
## log(income)      113.3820    87.8837  1.2901    0.1977
## I(log(income)^2) -26.9111    31.7457 -0.8477    0.3971
## I(log(income)^3)   3.0632     3.7369  0.8197    0.4128
```

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

Which of the models presented here is the most suitable? Comparing by $\overline{R^2}$ we find that, leaving out the log-linear model, all models have a similar fit. In the class of polynomial models, the cubic specification has the highest $\overline{R^2}$ whereas the linear-log specification is the best of the log-models (check this!). Let us first compare both models graphically by plotting the corresponding estimated regression functions

```
# scatter plot
plot(score ~ income,
     data = CASchools,
     col = "steelblue",
     pch = 20,
     main = "Linear-Log and Cubic Regression Functions")

# add Linear-Log regression line
order_id  <- order(CASchools$income)

lines(CASchools$income[order_id],
      fitted(LinearLog_model)[order_id],
      col = "darkgreen",
      lwd = 2)

# add cubic regression line
lines(x = CASchools$income[order_id],
      y = fitted(cubic_model)[order_id],
      col="darkred",
      lwd=2)
```



We see that both regression lines are nearly identical. Here the linear-log model is preferable since it has a slightly higher $\overline{R^2}$ and is more parsimonious in terms of regressors: it does not need higher-degree

polynomials.

9.3 Interactions Between Independent Variables

There are research questions where it is interesting to know how the effect on Y of a change in one independent variables dependends on the value of another independent variable. For example, we could ask if districts with many english learners benefit differentially from a decrease in class sizes than those with few english learning students. To answer this, we need to include an interaction term into a multiple regression model. We will consider three cases:

1. Interactions between two binary variables
2. Interactions between a binary and a continuous variable
3. Interactions between two continuous variables

The following subsections discuss those cases briefly and demonstrate how to perform such regressions using R.

Interactions Between Two Binary Variables

Take two binary variables D_1 and D_2 and the population regression

$$Y_i = \beta_0 + \beta_1 \times D_{1i} + \beta_2 \times D_{2i} + u_i.$$

Now assume

$$Y_i = \ln(Earnings_i), \tag{9.1}$$

$$\tag{9.2}$$

$$D_{1i} = \begin{cases} 1 & \text{if } i^{th} \text{ person has a college degree} \\ 0 & \text{else,} \end{cases} \tag{9.3}$$

$$\tag{9.4}$$

$$D_{2i} = \begin{cases} 1 & \text{if } i^{th} \text{ person is female} \\ 0 & \text{if } i^{th} \text{ person is male.} \end{cases} \tag{9.5}$$

$$\tag{9.6}$$

By now You should know that β_1 measures the average difference in $\ln(Earnings)$ between individuals with and without a college degree and β_2 is the gender differential in $\ln(Earnings)$, ceteris paribus. This model does not allow us to determine if there is a geneder specific effect of heaving a college degree and, if so, how strong this is. Luckily it is easy to come up with a model specification that allows to investigate this:

$$Y_i = \beta_0 + \beta_1 \times D_{1i} + \beta_2 \times D_{2i} + \beta_3 \times (D_{1i} \times D_{2i}) + u_i$$

$(D_{1i} \times D_{2i})$ is called an interaction term and β_3 measures the difference in the effect of having a college degree for women versus men.

Key Concept 8.3

A Method for Interpreting Coefficients in Regression with Binary Variables

Compute expected values of Y for each possible set described by the set of binary variables. Compare the expected values. The coefficients can be expressed either as expected values or as the difference between at least two expected values.

Following Key Concepts 8.1 we have

$$\begin{aligned} E(Y_i|D_{1i}=0,D_{2i}=d_2) &= \beta_0+\beta_1\times 0+\beta_2\times d_2+\beta_3\times(0\times d_2) \\ &= \beta_0+\beta_2\times d_2. \end{aligned}$$

Changing D_{1i} from 0 to 1 we obtain

$$\begin{aligned} E(Y_i|D_{1i}=1,D_{2i}=d_2) &= \beta_0+\beta_1\times 1+\beta_2\times d_2+\beta_3\times(1\times d_2) \\ &= \beta_0+\beta_1+\beta_2\times d_2+\beta_3 d_2 \end{aligned}$$

and hence the overall effect is

$$E(Y_i|D_{1i}=1,D_{2i}=d_2)-E(Y_i|D_{1i}=0,D_{2i}=d_2)=\beta_1+\beta_3 d_2$$

so the effect is a difference of expected values.

9.3.0.0.1 Application to the student-teacher ratio and the percentage of English learners

Now let

$$HiSTR=\begin{cases}1, & \text{if } STR\geq 20\\ 0, & \text{else,}\end{cases}$$

$$HiEL=\begin{cases}1, & \text{if } PctEL\geq 10\%\\ 0, & \text{else.}\end{cases}$$

We may use R construct the variables above.

```
# Add HiSTR to CASchools
CASchools$HiSTR <- as.numeric(CASchools$size >= 20)

# Add HiEL to CASchools
CASchools$HiEL <- as.numeric(CASchools$english >= 10)
```

We proceed by estimating the model

$$TestScore_i=\beta_0+\beta_1\times HiSTR+\beta_2\times HiEL+\beta_3\times(HiSTR\times HiEL)+u_i$$

using `lm()`. There are several ways to add an interaction term to the model `formula` argument of `lm()` but the most intuitive is to use the product of both variables `HiEL * HiSTR`.

```
# estimate the model with binary interaction term
bi_model <- lm(score ~ HiSTR + HiEL + HiSTR * HiEL, data = CASchools)

# print summary
summary(bi_model)
```

```
##
## Call:
## lm(formula = score ~ HiSTR + HiEL + HiSTR * HiEL, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.078 -10.679  -1.282   9.665  45.522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  664.143      1.316 504.852  < 2e-16 ***
## HiSTR         -1.908      2.235  -0.854    0.394
## HiEL         -18.316      2.144  -8.544 2.49e-16 ***
## HiSTR:HiEL    -3.260      3.223  -1.012    0.312
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.06 on 416 degrees of freedom
## Multiple R-squared:  0.2948, Adjusted R-squared:  0.2897
## F-statistic: 57.97 on 3 and 416 DF,  p-value: < 2.2e-16
```

The estimated regression model is

$$\widehat{TestScore} = 664.1 - 1.9 \times HiSTR - 18.3 \times HiEL - 3.3 \times (HiSTR \times HiEL)$$

and it predicts that the effect of moving from a school district with a low student-teacher ratio to a district with a high student-teacher ratio, depending on high or low percentage of english learners is $-1.9 - 3.3 \times HiEL$. So for districts with a low share of english learners ($HiEL = 0$), the estimated effect is a decrease of 1.9 points in test scores while for districts with a big fraction of english learner ($HiEL = 1$), the predicted decrease in test scores amounts to $1.9 + 3.3 = 5.2$ points.

We can also use the model to estimate the mean test score for each combination of binary variables.

```
# estimate means for all combinations of HiSTR and HiEL
predict(bi_model, newdata = data.frame("HiSTR"=0, "HiEL"=0))
```

```
##        1
## 664.1433
```

```
predict(bi_model, newdata = data.frame("HiSTR"=0, "HiEL"=1))
```

```
##        1
## 645.8278
```

```
predict(bi_model, newdata = data.frame("HiSTR"=1, "HiEL"=0))
```

```
##        1
## 662.2354
```

```
predict(bi_model, newdata = data.frame("HiSTR"=1, "HiEL"=1))
```

```
##        1
## 640.6598
```

Interactions Between a Continuous and a Binary Variable

Now consider a continuous variable X_i, the years of working experience of person i instead of the gender. We than have

$$Y_i = \ln(Earnings_i)$$

$$X_i = \text{working experience of person } i$$

$$D_i = \begin{cases} 1, & \text{if } i^{th} \text{ person has a college degree} \\ 0, & \text{else.} \end{cases}$$

The base model thus is

$$Y_i = \beta_0 + \beta_1 \times X_i + \beta_2 \times D_i + u_i,$$

a simple multiple regression model that allows us to estimate the average benefit of having a college degree holding working experience constant and the average effect on earnings of a change in working experience holding college degree constant.

By adding the interaction term $X_i \times D_i$ we allow the effect of an additional year of work experience to differ for person with and without college degree.

$$Y_i = \beta_0 + \beta_1 \times X_i + \beta_2 \times D_i + \beta_3 \times (X_i \times D_i) + u_i$$

Here, β_3 measures the difference in the effect of an additional year of work experience for college graduates versus nongraduates. A further possible specifications is

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i \times D_i) + u_i.$$

This model states that the expected impact of an additional year of work experience on earnings is the same for individuals without college degree but differs for college graduates. All three population regression functions can be visualized by straight lines. Key Concept 8.4 summarizes the differences.

Key Concept 8.4

Interactions Between Binary and Continuous Variables

An interaction term like $X_i \times D_i$ (where X is continuous and D is binary) allows for the slope to depend on the binary variable D. There are three possibilities:

1. Different intercept and same slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + u_i$$

2. Different intercept and different slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3 \times (X_i \times D_i) + u_i$$

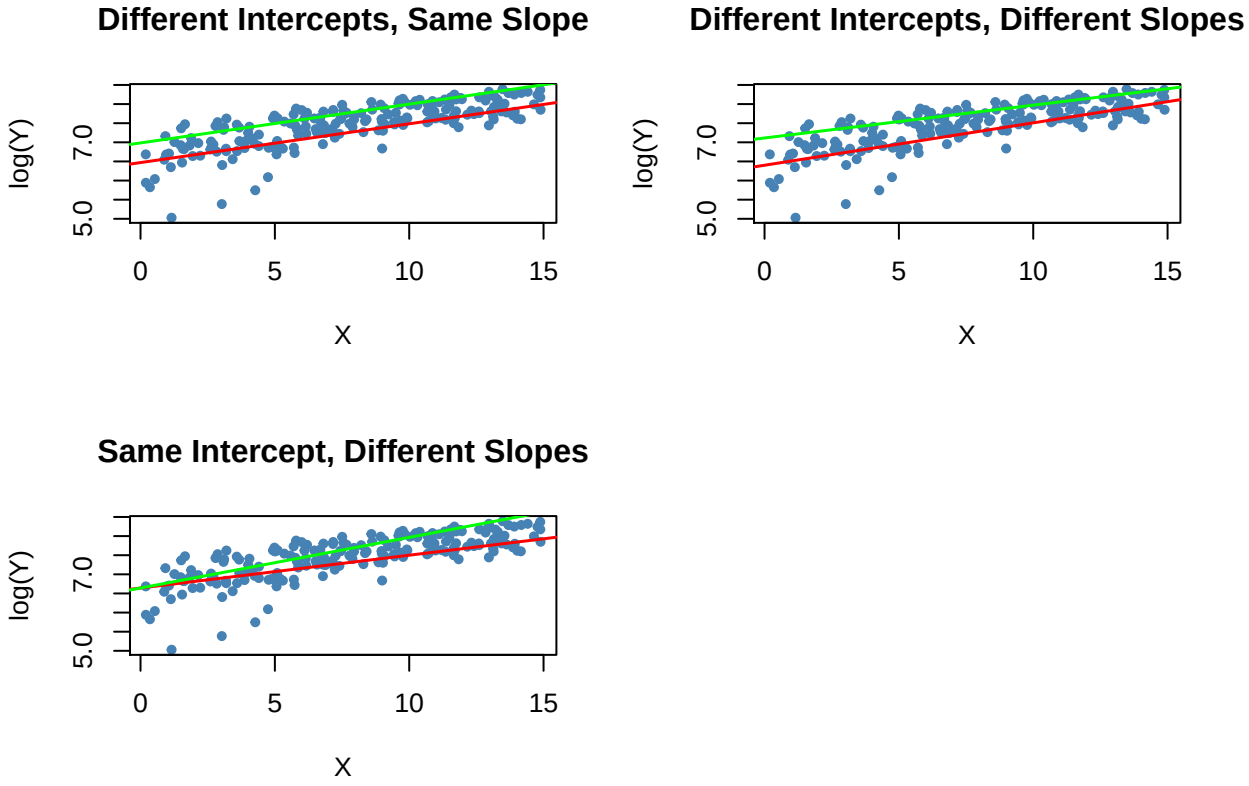
3. Same intercept and different slope:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 (X_i \times D_i) + u_i$$

The following code chunk shows how replicate the results shown in Figure 8.8 using fictional data.

```
# generate fictional data
set.seed(1)

X <- runif(200,0, 15)
D <- sample(0:1, 200, replace = T)
Y <- 450 +  150 * X + 500 * D + 50 * (X * D) + rnorm(200, sd=300)

# divide plotting area
m <- rbind(c(1, 2), c(3, 0))
layout(m)

# Estimate models and plot regression lines

# 1.(base model)
plot(X,log(Y),
     pch = 20,
     col = "steelblue",
     main = "Different Intercepts, Same Slope"
)
mod1_coef <- lm(log(Y) ~ X + D)$coefficients

abline(coef = c(mod1_coef[1], mod1_coef[2]),
       col = "red",
       lwd = 1.5
       )

abline(coef = c(mod1_coef[1] + mod1_coef[3], mod1_coef[2]),
       col = "green",
       lwd = 1.5
       )


# 2. (base model + interaction term)
plot(X,log(Y),
     pch = 20,
     col = "steelblue",
     main = "Different Intercepts, Different Slopes"
)

mod2_coef <- lm(log(Y) ~ X + D + X:D)$coefficients

abline(coef = c(mod2_coef[1], mod2_coef[2]),
       col = "red",
       lwd = 1.5
)

abline(coef = c(mod2_coef[1] + mod2_coef[3], mod2_coef[2] + mod2_coef[4]),
       col = "green",
       lwd = 1.5
)

# 3. (omission of D as regressor + interaction term)
```

```
plot(X,log(Y),
     pch = 20,
     col = "steelblue",
     main = "Same Intercept, Different Slopes"
)

mod3_coef <- lm(log(Y) ~ X + X:D)$coefficients

abline(coef = c(mod3_coef[1], mod3_coef[2]),
       col = "red",
       lwd = 1.5
)

abline(coef = c(mod3_coef[1], mod3_coef[2] + mod3_coef[3]),
       col = "green",
       lwd = 1.5
)
```



9.3.0.0.2 Application to the student-teacher ratio and the percentage of English learners

Using a model specification like 2. in Key Concept 8.3 we may answer the question whether the effect on test scores of decreasing the student-teacher ratio depends on the whether there are many or few English learners. We estimate the regression model

$$\widehat{TestScore}_i = \beta_0 + \beta_1 \times size_i + \beta_2 \times HiEL_i + \beta_2(size_i \times HiEL_i) + u_i.$$

```
# estimate the model
bci_model <- lm(score ~ size + HiEL + size * HiEL, data = CASchools)
```

```
# print summary to console
summary(bci_model)
```

```
##
## Call:
## lm(formula = score ~ size + HiEL + size * HiEL, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.356 -10.790  -0.841   9.911  46.457
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 682.2458    10.5109  64.908   <2e-16 ***
## size         -0.9685     0.5398  -1.794   0.0735 .
## HiEL          5.6391    16.7177   0.337   0.7360
## size:HiEL    -1.2766     0.8441  -1.512   0.1312
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.88 on 416 degrees of freedom
## Multiple R-squared:  0.3103, Adjusted R-squared:  0.3054
## F-statistic:  62.4 on 3 and 416 DF,  p-value: < 2.2e-16
```

The estimated regression model is

$$\widehat{TestScore} = 682.2 - 0.97 \times size + 5.6 \times HiEL - 1.28 \times (size \times HiEL).$$

We find that the estimated regression line for districts with a low fraction of English learners ($HiEL_i = 0$) is

$$\widehat{TestScore} = 682.2 - 0.97 \times size_i.$$

For districts with a high fraction of English learners we have

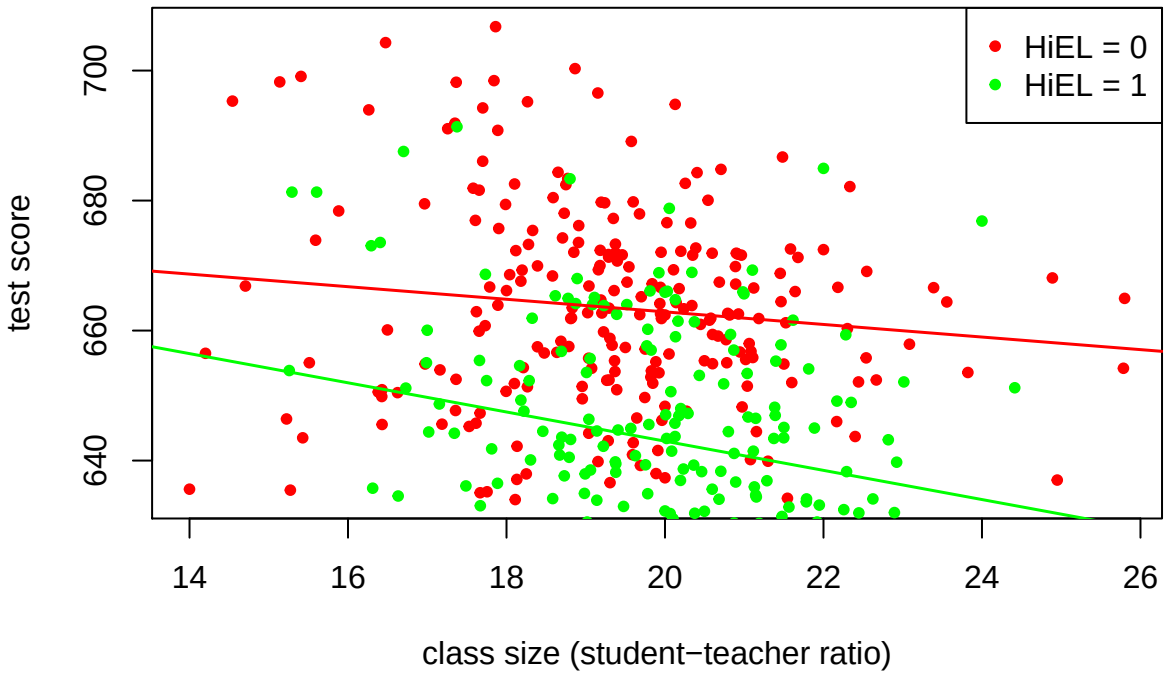
$$\begin{aligned} \widehat{TestScore} &= 682.2 + 5.6 - 0.97 \times size_i - 1.28 \times size_i && (9.7) \\ &= 687.8 - 2.25 \times size_i && (9.8) \end{aligned}$$

so the predicted increase in test scores following a reduction of the student-teacher ratio by 1 is about 0.97 points in districts where the fraction of English learners is low but 2.25 in districts with a high share of English learners. The difference between these effects is 1.28, the magnitude of the coefficient on the interaction term $size \times HiEL$.

The next code chunk draws both lines belonging to the model. In order to make observations with $HiEL = 0$ distinguishable from those with $HiEL = 0$, we assign different colors.

```
# identify observations PctEL >= 10
id <- CASchools$english >= 10

# plot observations with HiEL = 0 as red dots
plot(CASchools$size[!id], CASchools$score[!id],
     pch = 20,
```

```
     col = "red",
     main = "",
     xlab = "class size (student-teacher ratio)",
     ylab = "test score"
     )

# plot observations with HiEL = 1 as green dots
points(CASchools$size[id], CASchools$score[id],
     pch = 20,
     col = "green"
     )

# read out estimated coefficients of bci_model
coefs <- bci_model$coefficients

# Draw the estimated regression line for HiEL = 0
abline(coef = c(coefs[1], coefs[2]),
       col = "red",
       lwd = 1.5
       )

# Draw the estimated regression line for HiEL = 1
abline(coef = c(coefs[1]+coefs[3], coefs[2]+coefs[4]),
       col = "green",
       lwd = 1.5
       )

# Add a legend to the plot
legend("topright",
       pch=c(20,20),
       col = c("red","green"),
       legend = c("HiEL = 0", "HiEL = 1")
       )
```



Interactions Between Two Continuous Variables

If we have a regression model with Y the log earnings and two continuous regressors X_1, the years of work experience, and X_2, the years of schooling, a simple multiple regression model cannot be used to estimate the effect on wages of an additional year of work experience depending on a given level of schooling. This effect can be assessed by including the interaction term $(X_{1i} \times X_{2i})$ in the model:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 \times (X_{1i} \times X_{2i}) + u_i$$

Following Key Concept 8.1 we find that the effect on Y of a change on X_1 given X_2 is

$$\frac{\Delta Y}{\Delta X_1} = \beta_1 + \beta_3 X_2.$$

In the earnings example, a positive β_3 implies that the effect on log earnings of an additional year of work experience grows linearly with years of schooling.

Vice versa we have

$$\frac{\Delta Y}{\Delta X_2} = \beta_2 + \beta_3 X_1.$$

as the effect on log earnings of an additional year of schooling holding work experience constant.

Altogether we find that β_3 measures the effect of a unit increase in X_1 and X_2 beyond the effects of increasing X_1 alone and X_2 alone by one unit. The overall change in Y is thus

$$Y_i = (\beta_1 + \beta_3 X_2)\Delta X_1 + (\beta_2 + \beta_3 X_1)\Delta X_2 + \beta_3 \Delta X_1 \Delta X_2. \tag{9.9}$$

Key Concept 8.5 summarizes interactions between two regressors in multiple regression.

Key Concept 8.5

Interactions in Multiple Regression

The interaction term between the two regressors X_1 and X_2 is given by their product $X_1 \times X_2$. Adding this interaction term as a regressor to the model

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + u_i$$

allows the effect of change on X_2 to depend on the value of X_1 and vice versa. Thus the coefficient β_3 in the model

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \times X_2) + u_i$$

measures the effect of a one-unit increase in both X_1 and X_2 above and beyond the sum of both individual effects. This holds for continuous and binary regressors.

9.3.0.0.3 Application to the student-teacher ratio and the percentage of English learners

We will now examine the interaction between student-teacher ratio and the percentage of english learners which both are continuous variables.

```
# estimate regression model including the interaction between 'PctEL' and 'size'
cci_model <- lm(score ~ size + english + english:size, data = CASchools)

# print a summary to the console
summary(cci_model)
```

```
##
## Call:
## lm(formula = score ~ size + english + english:size, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.836 -10.226  -0.343   9.796  43.447
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  686.338527   9.402603  72.995   <2e-16 ***
## size          -1.117018   0.482537  -2.315   0.0211 *
## english       -0.672912   0.437985  -1.536   0.1252
## size:english   0.001162   0.021905   0.053   0.9577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.48 on 416 degrees of freedom
## Multiple R-squared:  0.4264, Adjusted R-squared:  0.4223
## F-statistic: 103.1 on 3 and 416 DF,  p-value: < 2.2e-16
```

The estimated model equation is

$$\widehat{TestScore} = 686.3 - 1.12 \times STR - 0.67 \times PctEL + 0.0012(STR \times PctEL).$$

For the interpretation, let us consider some quartiles of $PctEL$.

```
summary(CASchools$english)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.941   8.778  15.768  22.970  85.540
```

According to (9.9), if $PctEL$ is at its median 8.78 the slope of the regression function relating test scores and the student teacher ratio is predicted to be $-1.12 + 0.0012 \times 8.78 = -1.11$. This means that increasing STR by one unit deteriorates test scores by estimated 1.11 points. For the 75% quantile the estimated change on $TestScore$ of a one-unit increase in STR is estimated by $-1.12 + 0.0012 \times 23.0 = -1.09$ so the slope is somewhat lower. The interpretation is that for a school district with 23% english learners, a reduction of the studtent-teacher ratio by one unit is expected to increase the test scores by only 1.09 points.

Note, however, that the output produced by summary indicates that the difference of the effect for the median and the 75% quantile is not statistically significant. The p-value for the test $H_0 : \beta_3 = 0$ cannot be rejected at the 5% level of significance. Using robust standard errors, one can show that β_3 is not significantly different from zero even at the level of 10% (verify this using R!).

Example: The Demand for Economic Journals

In this section we replicate the empirical example *The Demand for Economic Journals* presented at pages 336 - 337 of the book. The central question is: how elastic is the demand by libraries for economic journals? The idea here is to analyze the relationship between the number of subscription to a journal at U.S. libraries and the journal's subscription price. The study uses the dataset `Journals` wich is provided with the `AER` package and contains observations for 180 economic journals for the year 2000. You can use the help function (`?Journals`) to get more information on the data after loading the package.

```
# load package and the dataset
library(AER)
data("Journals")
```

We measure the price as "price per citation" and have to compute journal age and the number of character manually. For consistency with the book we also rename the variables.

```
# define and rename variables
Journals$PricePerCitation <- Journals$price/Journals$citations
Journals$Age <- 2000 - Journals$foundingyear
Journals$Characters <- Journals$charpp * Journals$pages/10^6
Journals$Subscriptions <- Journals$subs
```

Note that the range of "price per citation" is quite large:

```
# compute summary statistics for price per citation
summary(Journals$PricePerCitation)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.005223  0.464495  1.320513  2.548455  3.440171 24.459459
```

The lowest price observed is a mere 0.5¢ per citation while the highest price is more than 20¢ per citation.

After loading and preparing the data, we estimate the four different model specifications. All models are log-log models. This is useful because it allows us to directly interpret the ceofficients as elasticities, see Key Concept 8.2. (I) is a simple linear model. To aleviate a possible omitted variable bias, (II) augments (I) by the covariates $\ln(Age)$ and $\ln(Characters)$. The largest model (III) attempts to capture nonlinearities in the relationship of $\ln(Subscriptions)$ and $\ln(PricePerCitation)$ using a cubic regression function of $\ln(PricePerCitation)$ and also adds the interaction term $(PricePerCitation \times Age)$ while specfication (IV) does not include the cubic term.

$$(I) \quad \ln(Subscriptions_i) = \beta_0 + \beta_1 \ln(PricePerCitation_i) + u_i$$

$$(II) \quad \ln(Subscriptions_i) = \beta_0 + \beta_1 \ln(PricePerCitation_i) + \beta_4 \ln(Age_i) + \beta_6 \ln(Characters_i) + u_i$$

$$\begin{aligned}(III) \quad \ln(Subscriptions_i) = {} & \beta_0 + \beta_1 \ln(PricePerCitation_i) + \beta_2 \ln(PricePerCitation_i)^2 \\ & + \beta_3 \ln(PricePerCitation_i)^3 + \beta_4 \ln(Age_i) + \beta_5 \left[\ln(Age_i) \times \ln(PricePerCitation_i)\right] \\ & + \beta_6 \ln(Characters_i) + u_i\end{aligned}$$

$$(IV) \quad \ln(Subscriptions_i) = \beta_0 + \beta_1 \ln(PricePerCitation_i) + \beta_4 \ln(Age_i) + \beta_5 + \beta_6 \ln(Characters_i) + u_i$$

```
# Estimate models (I) - (IV)
Journals_mod1 <- lm(log(Subscriptions) ~ log(PricePerCitation), data = Journals)

Journals_mod2 <- lm(log(Subscriptions) ~ log(PricePerCitation) + log(Age) +
    log(Characters), data = Journals)

Journals_mod3 <- lm(log(Subscriptions) ~ log(PricePerCitation) + I(log(PricePerCitation)^2) +
    I(log(PricePerCitation)^3) + log(Age) + log(Age):log(PricePerCitation) +
    log(Characters), data = Journals)

Journals_mod4 <- lm(log(Subscriptions) ~ log(PricePerCitation) + log(Age) +
    log(Age):log(PricePerCitation) + log(Characters), data = Journals)
```

Using `summary()`, we obtain the following estimated model equations:

$$(I) \quad \widehat{\ln(Subscriptions_i)} = 4.77 - 0.53 \ln(PricePerCitation_i)$$

$$(II) \quad \widehat{\ln(Subscriptions_i)} = 3.21 - 0.41 \ln(PricePerCitation_i) + 0.42 \ln(Age_i) + 0.21 \ln(Characters_i)$$

$$\begin{aligned}(III) \quad \widehat{\ln(Subscriptions_i)} = {} & 3.41 - 0.96 \ln(PricePerCitation_i) + 0.02 \ln(PricePerCitation_i)^2 \\ & + 0.004 \ln(PricePerCitation_i)^3 + 0.37 \ln(Age_i) \\ & + 0.16 \left[\ln(Age_i) \times \ln(PricePerCitation_i)\right] \\ & + 0.23 \ln(Characters_i)\end{aligned}$$

$$\begin{aligned}(IV) \quad \widehat{\ln(Subscriptions_i)} = {} & 3.43 - 0.90 \ln(PricePerCitation_i) + 0.37 \ln(Age_i) \\ & + 0.14 \left[\ln(Age_i) \times \ln(PricePerCitation_i)\right] + 0.23 \ln(Characters_i)\end{aligned}$$

It is of interest whether the coefficients the nonlinear transformations of $\ln(PricePerCitation)$ in Model (III) are statistically significant. To answer this, we use a F-Test:

```
# F-Test for significance of cubic terms
linearHypothesis(Journals_mod3,
                 c("I(log(PricePerCitation)^2)=0","I(log(PricePerCitation)^3)=0")
                 )
```

```
## Linear hypothesis test
```

```
##
## Hypothesis:
## I(log(PricePerCitation)^2) = 0
## I(log(PricePerCitation)^3) = 0
##
## Model 1: restricted model
## Model 2: log(Subscriptions) ~ log(PricePerCitation) + I(log(PricePerCitation)^2) +
##     I(log(PricePerCitation)^3) + log(Age) + log(Age):log(PricePerCitation) +
##     log(Characters)
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    175 82.738
## 2    173 82.500  2    0.2379 0.2494 0.7795
```

We cannot reject the null hypothesis $H_0 : \beta_3 = \beta_4 = 0$ in model (III).

We will now use the `stargazer()` function to generate a verbose tabular representation of all four estimated models.

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

# generate a Latex table using stargazer
stargazer(Journals_mod1, Journals_mod2, Journals_mod3, Journals_mod4,
          column.labels = c("(I)", "(II)", "(III)", "(IV)")
          )
```

Dependent variable:

log(Subscriptions)

(I)

(II)

(III)

(IV)

log(PricePerCitation)

-0.533***

-0.408***

-0.961***

-0.899***

(0.036)

(0.042)

(0.189)

(0.162)

I(log(PricePerCitation)2)

0.017

(0.024)

I(log(PricePerCitation)3)

0.004

(0.007)

log(Age)

0.424***

0.373***

0.374***

(0.090)

(0.089)

(0.089)

log(Characters)

0.206*

0.235**

0.229**

(0.107)

(0.106)

(0.105)

log(PricePerCitation):log(Age)

0.156***

0.141***

(0.055)

(0.045)

Constant

4.766***

3.207***

3.408***

3.434***

(0.056)

(0.314)

(0.318)

(0.315)

Observations

180

180

180

180

R2

0.557

0.613

0.635

0.634

Adjusted R2

0.555

0.607

0.622

0.626

Residual Std. Error

0.750 (df = 178)

0.705 (df = 176)

0.691 (df = 173)

0.688 (df = 175)

F Statistic

224.037*** (df = 1; 178)

93.009*** (df = 3; 176)

50.149*** (df = 6; 173)

75.749*** (df = 4; 175)

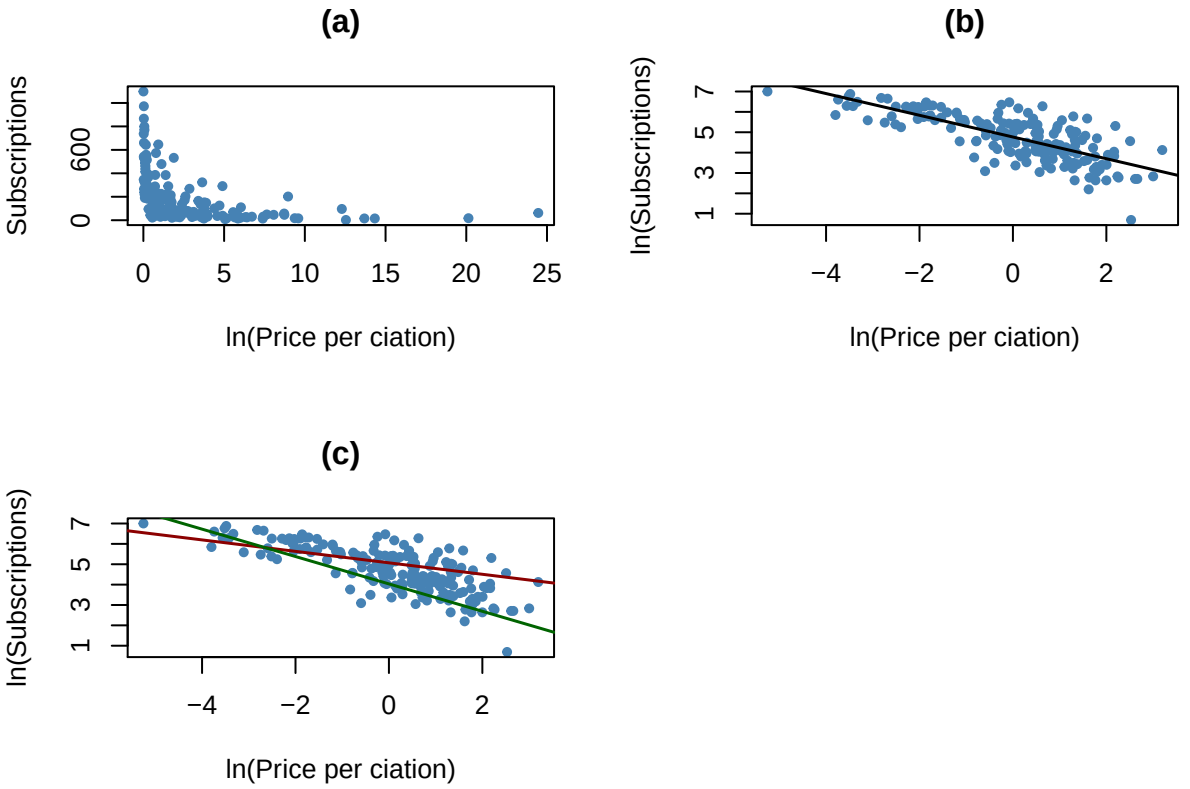
Note:

p<0.1; ***p<0.05;*** p<0.01

The subsequent code chunk reproduces figure 8.9 of the book:

```
# divide plotting area
m <- rbind(c(1, 2), c(3, 0))
layout(m)

# scatterplot
plot(Journals$PricePerCitation,
     Journals$Subscriptions,
     pch = 20,
     col = "steelblue",
     ylab = "Subscriptions",
     xlab = "ln(Price per ciation)",
     main = "(a)"
     )

# log-log scatterplot and estimated regression line (I)
plot(log(Journals$PricePerCitation),
     log(Journals$Subscriptions),
     pch = 20,
     col = "steelblue",
     ylab = "ln(Subscriptions)",
     xlab = "ln(Price per ciation)",
```

```
     main = "(b)"
     )

abline(Journals_mod1,
       lwd = 1.5
       )

# log-log scatterplot and regression lines (IV) for Age = 5 and Age = 80
plot(log(Journals$PricePerCitation),
     log(Journals$Subscriptions),
     pch = 20,
     col = "steelblue",
     ylab = "ln(Subscriptions)",
     xlab = "ln(Price per ciation)",
     main = "(c)"
     )

JM4C <-Journals_mod4$coefficients

# Age = 80
abline(coef = c(JM4C[1] + JM4C[3] * log(80),
                JM4C[2] + JM4C[5] * log(80)
                ),
       col = "darkred",
       lwd = 1.5
)

# Age = 5
abline(coef = c(JM4C[1] + JM4C[3] * log(5),
                JM4C[2] + JM4C[5] * log(5)
                ),
       col = "darkgreen",
       lwd = 1.5
)
```



As can be seen from plots (a) and (b), the relation between subscriptions and the citation price is inverse and nonlinear and log-transforming both variables makes it approximately linear. Plot (c) shows that the price elasticity of journal subscriptions depends on the journal's age: the red line shows the estimated relationship for $Age = 80$ while the green line represents the prediction from model (IV) for $Age = 5$.

Which conclusion can be made?

1. We conclude that the demand for journals is more elastic for young journals than for old journals.
2. For model (III) we cannot reject the null hypothesis that the coefficients on $\ln(PricePerCitation)^2$ and $\ln(PricePerCitation)^3$ are both zero using an F-test. This is evidence supporting a linear relation between log-subscriptions and log-price.
3. Demand is greater for Journals with more characters, holding price and age constant.

Hence we have found evidence, that the price elasticity of demand for economic journals depends on the age of the journal. Altogether our estimates suggest that the demand is very inelastic, i.e. the librararies' demand for economic journals is quite insensitve to the price: using model (IV), even for a young journal $(Age = 5)$ we estimate the price elasticity to be $-0,899 + 0,374 \times \ln(5) + 0,141 \times [\ln(1) \times \ln(5)] \approx -0.3$ so that a one percent increase in price is predicted to reduce the demand by only 0.3 percent.

This finding comes at no surprise since providing the most recent research is a necessity for libraries.

9.4 Nonlinear Effects on Test Scores of the Student-Teacher Ratio

In this section we will discuss three specific questions about the relatin between test scores and the student-teacher ratio:

1. Does the effect on test scores of decreasing the student-teacher ratio depend on the fraction of english learners when we control for economic idiosyncracies of different districts?

2. Does this effect depend on the the student-teacher ratio?
3. How strong is the effect of decreasing the student-teacher ratio (by two students per teacher) if we take into account economic characteristics and nonlinearities?

Two answer these questions we will consider a total of seven models, some of which are nonlinear regression specifications of the types that have been discussed before. As measures for the students' economic backgrounds, we will additionally consider the regressors *lunch* and $\ln(income)$. We use the logarithm of *income* because the analysis discussed in chapter 8.2 showed that the nonlinear relationship between *income* and *TestScores* is approximately logarithmic. We do not include expenditures per pupil (*expenditure*) because doing so allows the expenditures to vary with the student-teacher ratio, see chapter 7.2.

9.4.0.0.1 Nonlinear Regression Models of Test Scores

The considered model specifications are:

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_4 english_i + \beta_9 lunch_i + u_i \tag{9.10}$$

$$\tag{9.11}$$

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_4 english_i + \beta_9 lunch_i + \beta_{10} \ln(income_i) + u_i \tag{9.12}$$

$$\tag{9.13}$$

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_5 HiEL_i + \beta_6 (HiEL_i \times size_i) + u_i \tag{9.14}$$

$$\tag{9.15}$$

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_5 HiEL_i + \beta_6 (HiEL_i \times size_i) + \beta_9 lunch_i + \beta_{10} \ln(income_i) + u_i \tag{9.16}$$

$$\tag{9.17}$$

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_2 size_i^2 + \beta_5 HiEL_i + \beta_9 lunch_i + \beta_{10} \ln(income_i) + u_i \tag{9.18}$$

$$\tag{9.19}$$

$$\begin{aligned} \widehat{TestScore}_i = {} & \beta_0 + \beta_1 size_i + \beta_2 size_i^2 + \beta_3 size_i^3 + \beta_5 HiEL_i + \beta_6 (HiEL \times size) \qquad (9.20) \\ & + \beta_7 (HiEL_i \times size_i^2) + \beta_8 (HiEL_i \times size_i^3) + \beta_9 lunch_i + \beta_{10} \ln(income_i) + u_i \qquad (9.21) \end{aligned}$$

$$\tag{9.22}$$

$$\widehat{TestScore}_i = \beta_0 + \beta_1 size_i + \beta_2 size_i^2 + \beta_3 size_i^3 + \beta_4 english + \beta_9 lunch_i + \beta_{10} \ln(income_i) + u_i \tag{9.23}$$

```
# Estimate all models
TestScore_mod1 <- lm(score ~ size + english + lunch, data = CASchools)

TestScore_mod2 <- lm(score ~ size + english + lunch + log(income), data = CASchools)

TestScore_mod3 <- lm(score ~ size + HiEL + HiEL:size, data = CASchools)

TestScore_mod4 <- lm(score ~ size + HiEL + HiEL:size + lunch + log(income),
    data = CASchools)

TestScore_mod5 <- lm(score ~ size + I(size^2) + I(size^3) + HiEL + lunch + log(income),
    data = CASchools)

TestScore_mod6 <- lm(score ~ size + I(size^2) + I(size^3) + HiEL + HiEL:size +
    HiEL:I(size^2) + HiEL:I(size^3) + lunch + log(income), data = CASchools)

TestScore_mod7 <- lm(score ~ size + I(size^2) + I(size^3) + english + lunch +
    log(income), data = CASchools)
```

Next, we may use `summary()` to assess the models. Using `stargazer()` we may also obtain a LATEX-based tabular representation of all regression outputs and which is more convenient for comparison of the models.

```
# generate a latex table of regression outputes
stargazer(TestScore_mod1, TestScore_mod2, TestScore_mod3, TestScore_mod4, TestScore_mod5,
    TestScore_mod6, TestScore_mod7, column.labels = c("(1)", "(2)", "(3)", "(4)",
        "(5)", "(6)", "(7)"))
```

Dependent variable:

score

(1)

(2)

(3)

(4)

(5)

(6)

(7)

size

-1.00***

-0.73***

-0.97*

-0.53*

64.34**

83.70***

65.29**

(0.24)

(0.23)

(0.54)

(0.30)

(25.46)

(29.69)

(25.48)

english

-0.12***

-0.18***

-0.17***

(0.03)

(0.03)

(0.03)

I(size2)

-3.42***

-4.38***

-3.47***

(1.29)

(1.51)

(1.29)

I(size3)

0.06***

0.07***

0.06***

(0.02)

(0.03)

(0.02)

lunch

-0.55***

-0.40***

-0.41***

-0.42***

-0.42***

-0.40***

(0.02)

(0.03)

(0.03)

(0.03)

(0.03)

(0.03)

log(income)

11.57***

12.12***

11.75***

11.80***

11.51***

(1.74)

(1.77)

(1.73)

(1.75)

(1.73)

HiEL

5.64

5.50

-5.47***

816.08*

(16.72)

(9.14)

(1.03)

(434.61)

size:HiEL

-1.28

-0.58

-123.28*

(0.84)

(0.46)

(66.35)

I(size2):HiEL

6.12*

(3.35)

I(size3):HiEL

-0.10*

(0.06)

Constant

700.15***

658.55***

682.25***

653.67***

252.05

122.35

244.81

(4.69)

(7.68)

(10.51)

(8.89)

(165.82)

(192.18)

(165.93)

Observations

420

420

420

420

420

420

420

R2

0.77

0.80

0.31

0.80

0.80

0.80

0.80

Adjusted R2

0.77

0.79

0.31

0.79

0.80

0.80

0.80

Residual Std. Error

9.08 (df = 416)

8.64 (df = 415)

15.88 (df = 416)

8.63 (df = 414)

8.56 (df = 413)

8.55 (df = 410)

8.57 (df = 413)

F Statistic

476.31*** (df = 3; 416)

405.36*** (df = 4; 415)

62.40*** (df = 3; 416)

325.80*** (df = 5; 414)

277.21*** (df = 6; 413)

185.78*** (df = 9; 410)

276.52*** (df = 6; 413)

Note:

p<0.1; ***p<0.05;*** p<0.01

Let us summarize what can be concluded from the table above:

First of all, it is apparent that the student-teacher ratio is statistically significant in all seven models. Adding ln($income$) to model (1) we find that the corresponding coefficient is statistically significant at the level of 1% while all other coefficients remain at their significance level. Furthermore, the estimate for the coefficient on $size$ is roughly 0.27 points larger which could be a sign of attenuated omitted variable bias. We consider this a reason to include ln($income$) as a regressor in other models, too.

Regressions (3) and (4) are regression that aim to assees the effect of allowing for an interaction between $size$ and $HiEL$, without and with economic control variables. In both modesl, the interaction term and the dummy are not statistically significant. Thus, even with economic control variables we cannot the reject the null hypothesis, that the effect of the student-teacher ratios on test scores is the same for districts with high and districts with low share of english learning students.

Regression (5) includes a cubic term for the student-teacher ratio and omittes the interaction between $size$ and $HiEl$. The estimation results indicate that there is a nonlinear effect of the student-teacher ratio on test scores. (Can You verify this using an F-test of $H_0 : \beta_2 = \beta_3 = 0$?)

Consequently, regression (6) further explores whether the fraction of english learners is also accountable for the effect of the student-teacher ration by using $HiEL \times size$ and the interactions $HiEL \times size^2$ and $HiEL \times size^3$. All individual t-tests indicate that that there are significant effects. We check this using a robust F-test of $H_0 : \beta_6 = \beta_7 = \beta_8 = 0$.

```
# check joint significance of interactions
linearHypothesis(TestScore_mod6,
                 c("size:HiEL=0", "I(size^2):HiEL=0", "I(size^3):HiEL=0"),
                 vcov. = vcovHC(TestScore_mod6, type = "HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## size:HiEL = 0
## I(size^2):HiEL = 0
## I(size^3):HiEL = 0
##
## Model 1: restricted model
## Model 2: score ~ size + I(size^2) + I(size^3) + HiEL + HiEL:size + HiEL:I(size^2) +
##     HiEL:I(size^3) + lunch + log(income)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F  Pr(>F)
```

```
## 1    413
## 2    410  3 2.6903 0.04597 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

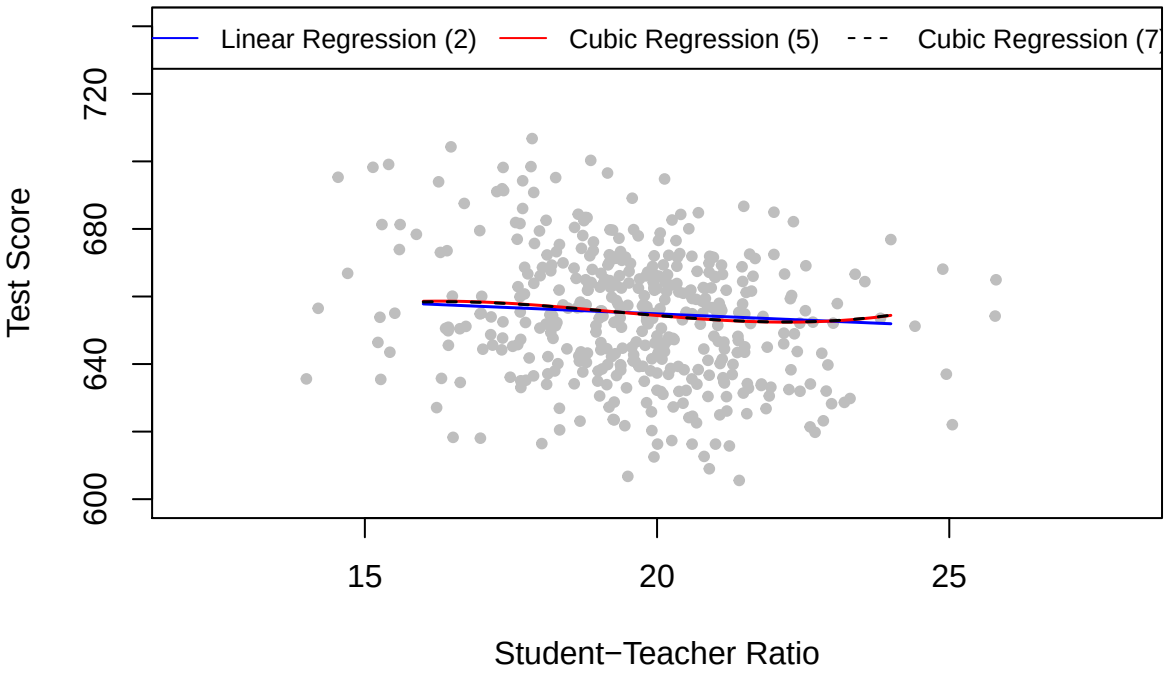
We find that the null can be rejected at the level of 5% and conclude that the regression function differs for districts with high and low percentage of english learners.

Specification (7) uses a continuous measure for the share of english learners instead of a dummy variable (and thus omitts the interaction terms). We observe only small changes to the coefficient estimates on the other regressors and thus conclude that the results observed for specification (5) are not sensitive to the way the percentage of english learners is measured.

We continue by reproducing figure 8.10 of the book using R for interpretation of the nonlinear specifications (2), (5) and (7).

```
# scatterplot
plot(CASchools$size,
     CASchools$score,
     xlim = c(12,28),
     ylim = c(600,740),
     pch=20,
     col="gray",
     xlab="Student-Teacher Ratio",
     ylab = "Test Score")

# add a legend
legend("top",
       legend = c("Linear Regression (2)","Cubic Regression (5)", "Cubic Regression (7)"),
       cex = 0.8,
       ncol = 3,
       lty = c(1,1,2),
       col = c("blue","red","black")
       )

# data for use with predict()
new_data <- data.frame("size"=seq(16,24,0.05),
                       "english"=mean(CASchools$english),
                       "lunch"=mean(CASchools$lunch),
                       "income"=mean(CASchools$income),
                       "HiEL"=mean(CASchools$HiEL)
                       )

# add estimated regression function for model (2)
fitted <- predict(TestScore_mod2, newdata = new_data)

lines(new_data$size,
      fitted,
      lwd=1.5,
      col="blue")

# add estimated regression function for model (5)
fitted <- predict(TestScore_mod5, newdata = new_data)

lines(new_data$size,
      fitted,
```

```
      lwd=1.5,
      col="red")

# add estimated regression function for model (7)
fitted <- predict(TestScore_mod7, newdata = new_data)

lines(new_data$size,
      fitted,
      col="black",
      lwd=1.5,
      lty=2)
```

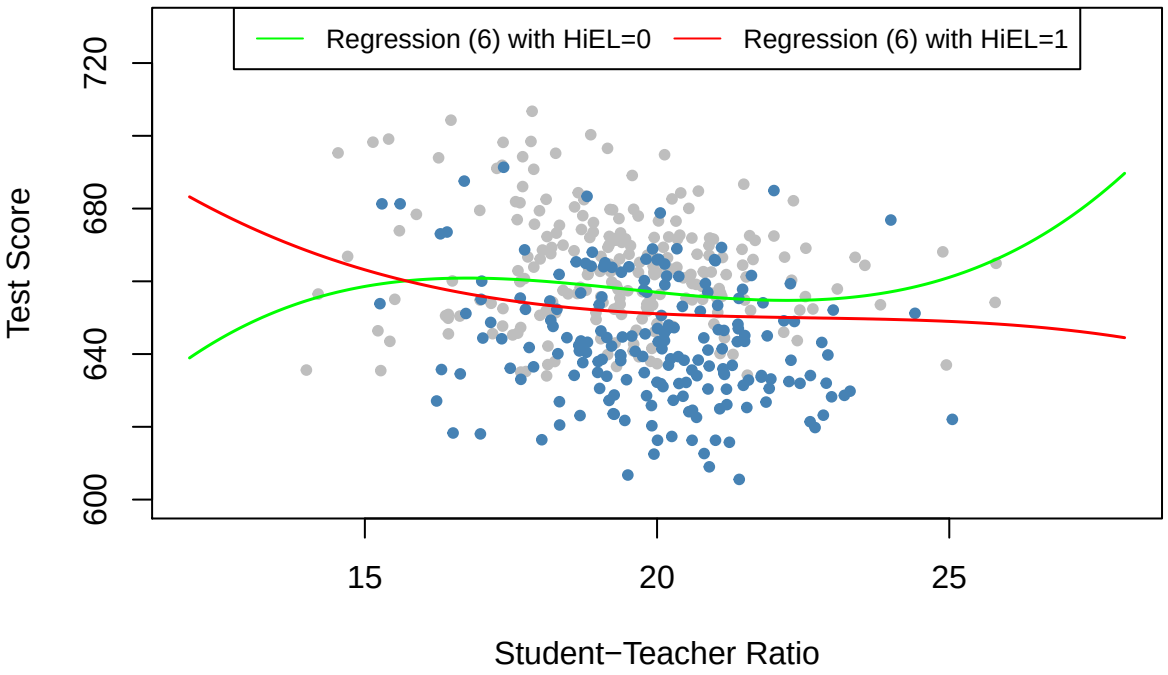


Note that for the figure above all regressors except for *size* are set to their sample averages. We see that the cubic regressions (5) and (7) are almost identical. They indicate that the relation between test scores and the student-teacher ratio has only a small amount of nonlinearity since they do not deviate much from the regression function of (2).

The next code chunk reproduces figure 8.11 of the book. We use `plot()` and `points()` to color observations depending on $HiEL$. Again, the regression lines are drawn based on predictions using average sample averages of all regressors except for *size*.

```
# scatterplot

# observations with HiEL = 0
plot(CASchools$size[CASchools$HiEL==0],
     CASchools$score[CASchools$HiEL==0],
     xlim = c(12,28),
     ylim = c(600,730),
     pch=20,
     col="gray",
     xlab="Student-Teacher Ratio",
     ylab = "Test Score")
```

```
# observations with HiEL = 1
points(CASchools$size[CASchools$HiEL==1],
       CASchools$score[CASchools$HiEL==1],
       col="steelblue",
       pch=20)

# add a legend
legend("top",
       legend = c("Regression (6) with HiEL=0", "Regression (6) with HiEL=1"),
       cex = 0.8,
       ncol = 2,
       lty = c(1,1),
       col = c("green","red")
       )

# data for use with predict()
new_data <- data.frame("size"=seq(12,28,0.05),
                       "english"=mean(CASchools$english),
                       "lunch"=mean(CASchools$lunch),
                       "income"=mean(CASchools$income),
                       "HiEL"=0
                       )

# add estimated regression function for model (6) with HiEL=0
fitted <- predict(TestScore_mod6, newdata = new_data)

lines(new_data$size,
      fitted,
      lwd=1.5,
      col="green")

# add estimated regression function for model (6) with HiEL=1
new_data$HiEL=1

fitted <- predict(TestScore_mod6, newdata = new_data)

lines(new_data$size,
      fitted,
      lwd=1.5,
      col="red")
```



From the regression output we see that (6) finds statistically significant coefficients on the interaction terms $HiEL : size$, $HiEL : size^2$ and $HiEL : size^3$ i.e. there is evidence that the nonlinear relationship connecting test scores and student-teacher ration depends on the amount of English learning students in the district. However, the figure shows that this difference is not/ of practical importance and is a good example of why one should be careful when inpreting nonlinear models: though the two regression functions look different, we see that the slope of both functions, that is for both districts with high and districs with low share of English learning students, is almost identical for student-teacher ratios between 17 and 23. Since this range includes almost 90% of all observations, we can be confident about this.

One might be tempted to object since both functions show opposing slopes for student-teacher ratios below 15 and beyond 24. Two things can be set aginst that:

1. The data has only few observations with low and high values of the student-teacher ratio so there is only little information to be exploited when estimating the model. This means the estimated function is less precise in the extremes of the dataset

2. Such a behaviour is a typical caveat when using cubic functions since they generally show extreme behaviour for extreme observations. Think of the graph of $f(x) = x^3$.

We thus conclude that there is no dependence of the relation between class size and test scores on the percentage of English learners in the district.

Summary

We are now able to answer the three question posed at the beginning of this section.

1. Considering the linear models, the percentage of English learners has only little influence on the effect of test scores of changing the student-teacher ratio. This result keeps valid if we control for economic background of the students. While the cubic specification (6) provides evidence, the strength of the effect is negligible.

2. When controlling for the students' economic backgroumd we find evidence of nonlinearities in the relationship between student-teacher ratio and test scores.

3. The linear specification (2) predicts that a reduction of the student-teacher ratio by two students per teacher leads to an improvement in test scores of about $-0.73 \times (-2) = 1.46$ points. Since the model is linear, this effect is independent of the class size. Asssume that the student-teacher ratio is 20. For example, model (5) predicts that the reduction increases test scores by

$$64.33 * 18 + 18^2 * (-3.42) + 18^3 * (0.059) - (64.33 * 20 + 20^2 * (-3.42) + 20^3 * (0.059)) \approx 3.3$$

points. if the ratio is 22, a reduction to 20 leads to a predicted improvement in test scores of

$$64.33 * 20 + 20^2 * (-3.42) + 20^3 * (0.059) - (64.33 * 22 + 22^2 * (-3.42) + 22^3 * (0.059)) \approx 2.4$$

points. This suggests that the effect is stronger in smaller classes.

Chapter 10

Assessing Studies Based on Multiple Regression

The majority of this chapter of the book is of a theoretical nature. Therefore this section briefly reviews the concepts of internal and external validity in general and discusses examples of threats to internal and external validity of multiple regression models. We will disuss consequences of

- Misspecification of the functional form of the regression function
- Measurement errors
- Missing data and sample selection
- Simultaneous causality

as well as sources of inconsitency of OLS standard errors. We also review concerns of internal validity and external validity in the context of forecasting using regression models.

The chapter closes with an application with R where we assess whether results found by multiple regression using the `CASchools` data can be generalized to school districs of another federal state of the United States.

For a more detailed treatment of these topics we encourage you to work through chapter 9 of the book.

10.1 Internal and External Validity

Key Concept 9.1

Internal and External Validity

We say a statistical analysis has **internal** validity if the statistical inferences made about causal effects are valid for the considered population.

An analysis is said to have **external** validity if inferences and conclusion are valid for the studies population and can be generalized to other populations and settings.

Threats to Internal Validity

For internal validity to exist, there are two conditions:

1. The estimator of the causal effect, that is the coefficient(s) of interest, needs to be unbiased and consistent.

2. Statistical Inference is valid, that is hypotheses tests should have the desired significance level and confidence intervals should have the desired confidence level.

In multiple regression, we estimate the model coefficients using OLS. Thus for condition 1. to be fulfilled we need the OLS estimator to be unbiased and consistent. For the second condition to be valid, the standard errors must be valid such that hypothesis testing and computation of confidence intervals yield results that are trustworthy. Remember that a necessary condition for conditions 1. and 2. to be fulfilled is that the assumptions of Key Concept 6.4 hold.

Threats to External Validity

External validity might be invalid

- if there are **differences between the considered populations**.
- if there are **differences in the settings** of the considered populations, e.g. the legal framework, the time of the investigation etc.

10.2 Threats to Internal Validity of Multiple Regression Analysis

This section lists five sources that cause the OLS estimator in (multiple) regression models to be biased and inconsistent for the causal effect of interest and discusses possible remedies. Note that all five sources arise from violation of the first least squares assumption in Key Concept 6.4.

This sections treats:

- Omitted variable Bias
- Misspecification of the functional form
- Measurement errors
- Missing data and sample selection
- Simultaneous causality bias

Beside these threats for consistency of the coefficient estimation, we will also briefly discuss sources of inconsistent estimation of OLS standard errors.

Omitted Variable Bias

Key Concept 9.2

Omitted Variable Bias: Should I include More Variables in My Regression?

Inclusion of additional variables reduces the risk of omitted variable bias but may increase the variance of the estimator of the coefficient of interest.

We present some guidelines that help deciding whether to include an additional variable:

1. Specify the coefficient(s) of interest.
2. Identify the most important potential sources of omitted variable bias by using knowlegde available *before* estimating the model. You should end up with a base specification and a set of regressors that are questionable.
3. Use different model specifactions to test whether questionable regressors have coefficients different from zero.

4. Use tables to provide "full disclosure" of your results i.e. present different model specifications that do both support your argument and enable the reader to see the effect of including questionable regressors.

By now you should be aware of omitted variable bias and its consequences. Key Concept 9.2 gives some guidelines on how to proceed if there are control variables that possibly allow to reduce an omitted variable bias. If including additional variables to mitigate the bias is not an option because there are no adequate controls, there are different approaches to solve the problem:

1. Usage of Panel data (discussed in Chapter 10)
2. Usage of Instrumental variables regression (discussed in Chapter 12)
3. Usage of a randomized control experiment (discussed in Chapter 13)

Misspecification of the Functional Form of the Regression Function

If the population regression function is nonlinear but the regression model is linear, we say that the functional form of the regression function is misspecified. This leads to a bias of the OLS estimator.

Key Concept 9.3

Functional Form Misspecification

We say a regression suffers from misspecification of the functional form when the functional form of the estimated regression model differs from the functional form of the population regression function. Functional form misspecification leads to biased and inconsistent coefficient estimators. A way to detect functional form misspecification is to plot the estimated regression function and the data. This may also be helpful to choose the correct functional form.

It is easy to come up with an example case of misspecification of the functional form:

Consider the case where the population regression function is

$$Y_i = -X_i^2$$

but the estimated model is

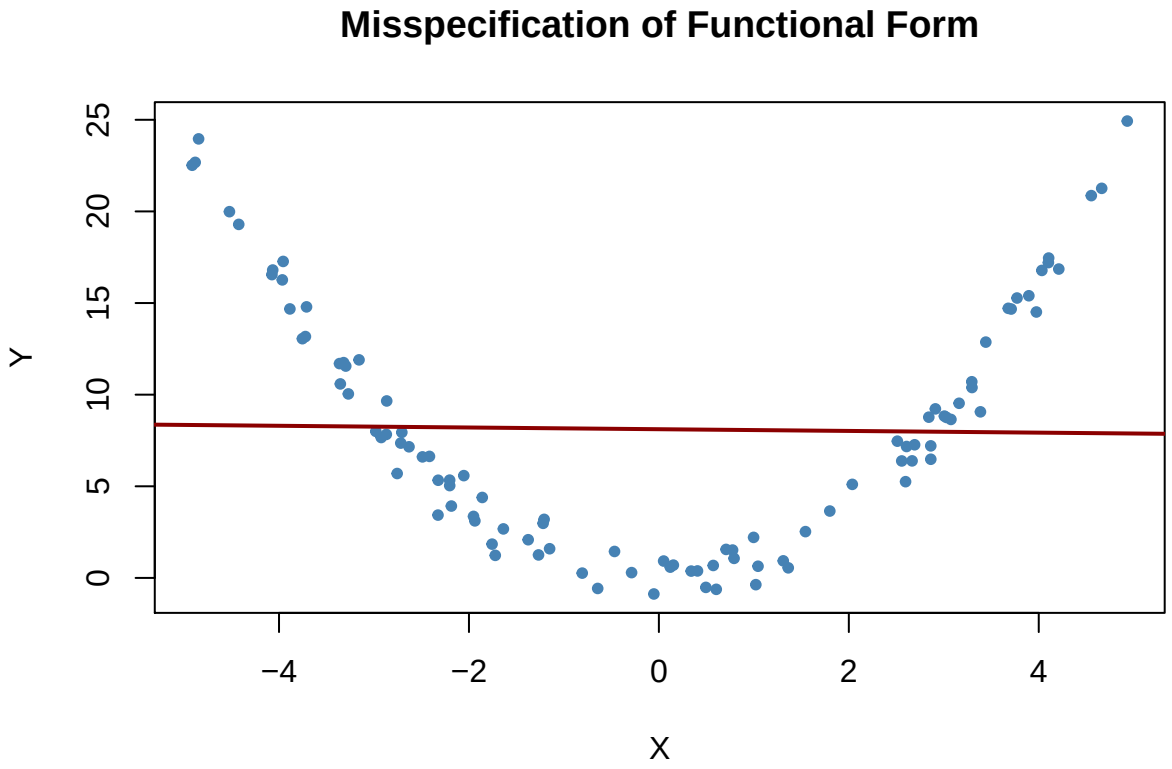
$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

so that the regression function is misspecified.

```
# set random seed for reproducibility
set.seed(3)

# generate data set
X <- runif(100, -5, 5)
Y <- X^2 + rnorm(100)

# plot the data
plot(X, Y,
     main = "Misspecification of Functional Form",
     pch = 20,
     col = "steelblue"
     )

# estimate and plot the regression function
ms_mod <- lm(Y ~ X)
ms_mod
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Coefficients:
## (Intercept)            X
##     8.11363     -0.04684
```

```
abline(ms_mod,
       col = "darkred",
       lwd = 2
       )
```



It is evident that regression errors are relatively small for observations close to $X = -3$ and $X = 3$ but increase for X values closer to zero and even more for regressor values beyond -4 and 4. Consequences are drastic: the intercept is estimated to be 8.1 and for the slope parameter we obtain an estimate close to zero which is obviously wrong. This issue does not dissapear as the number of observations is inreased because OLS is biased *and* inconsistent due to the misspecification of the regression function.

Measurement Error and Errors-in-Variables Bias

Key Concept 9.4

Errors-in-Variable Bias

When independent variables are measure imprecisely, we speak of Errors-in-variables bias. This bias does not disappear if the sample size is large. If the measurement error has mean zero and is independent of the affected variable, the OLS estimator of the respective coefficient is biased towards zero.

Suppose you are measuring data on the single regressor X_i incorrectly so that there is a measurement error

and you observe $\widetilde{X}_i$ instead of X_i. Then, instead of estimating the population the regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

you end up estimating

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 \widetilde{X}_i + \underbrace{\beta_1 (X_i - \widetilde{X}_i) + u_i}_{=v_i} \\ Y_i &= \beta_0 + \beta_1 \widetilde{X}_i + v_i \end{aligned}$$

where $\widetilde{X}_i$ and the error term v_i are correlated. Thus OLS would be biased and inconsitent for the true β_1 in this example.

One can show that direction and strength of the bias depend on the correlation between the observed regressor, $\widetilde{X}_i$, and the measurement error, $(X_i - \widetilde{X}_i)$. The correlation depends on the type of the measurement error.

The classical measurement error model assumes that the error, w_i, has zero mean and that the error is uncorrelated with the variable, X_i, and the error term of the poulation regression model, u_i:

$$\widetilde{X}_i = X_i + w_i \quad , \quad corr(w_i, u_i) = 0 \quad , \quad corr(w_i, X_i) = 0 \tag{10.1}$$

Then it holds that

$$\widehat{\beta}_1 \xrightarrow{p} \frac{\sigma_X^2}{\sigma_X^2 + \sigma_w^2} \tag{10.2}$$

Which implies inconsistency as $\sigma_X^2, \sigma_w^2 > 0$ such that the fraction in (10.2) is smaller than 1. Note that, there are two extreme cases: first, if there is no measurement error, $\sigma_w^2 = 0$ such that $\widehat{\beta}_1 \xrightarrow{p} \beta_1$. Second, if $\sigma_w^2 \gg \sigma_X^2$ we have $\widehat{\beta}_1 \xrightarrow{p} 0$. This is the case if the measurement error is so large that there essentially is no information on X.

The most obvious way to deal with errors-in-variables bias is to use an accurately measured X. If this not possible, instrumental variables regression can be an option. One might also deal with the issue by using a mathematical model of the measurement error and correct the estimates. For example, if it is plausible that the classical measurement error model applies and if there is information that can be used to estimate the ratio in equation (10.2), one could compute an estimate that corrects for the downwoard bias.

For example, consider two bivariate normal distributed random variables X, Y. It is a well known result, that the conditional expectation function of Y given X has the form

$$E(Y|X) = E(Y) + \rho_{X,Y} \frac{\sigma_Y}{\sigma_X} \left[X - E(X) \right]. \tag{10.3}$$

Thus for

$$(X, Y) \sim \mathcal{N} \left[\begin{pmatrix} 50 \\ 100 \end{pmatrix}, \begin{pmatrix} 10 & 5 \\ 5 & 10 \end{pmatrix} \right] \tag{10.4}$$

according to (10.3), the population regression function is

$$\begin{aligned} Y_i &= 100 + 0.5(X - 50) \\ &= 75 + 0.5X. \end{aligned}$$

Suppose you gather data on X and Y but that can you only measure $\widetilde{X}_i = X_i + w_i$ with $w_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sqrt{10})$. Since the w_i are purely random, there is no correlation between the X_i and the w_i so that we have a case of the classical measurement error model. We can illustrate this using R.

```
# random seed
set.seed(1)

# load the mvtnorm package and simulate data
library(mvtnorm)
dat <- data.frame(
  rmvnorm(1000, c(50,100),
          sigma = cbind(c(10,5), c(5,10))
          )
  )

# set columns names
colnames(dat) <- c("X","Y")
```

We now estimate a simple linear regression of Y on X using this sample data and run the same regression again but this time add i.i.d. $\mathcal{N}(0, \sqrt{10})$ errors to X.

```
# estimate the model (without measurement error)
nme_mod <- lm(Y ~ X, data = dat)

# estimate the model (measurement error in X)
dat$X <- dat$X + rnorm(n = 1000, sd = sqrt(10))
me_mod <- lm(Y ~ X, data = dat)

# print estimated coefficients to console
nme_mod$coefficients
```

```
## (Intercept)           X
##  76.3002047   0.4755264
```

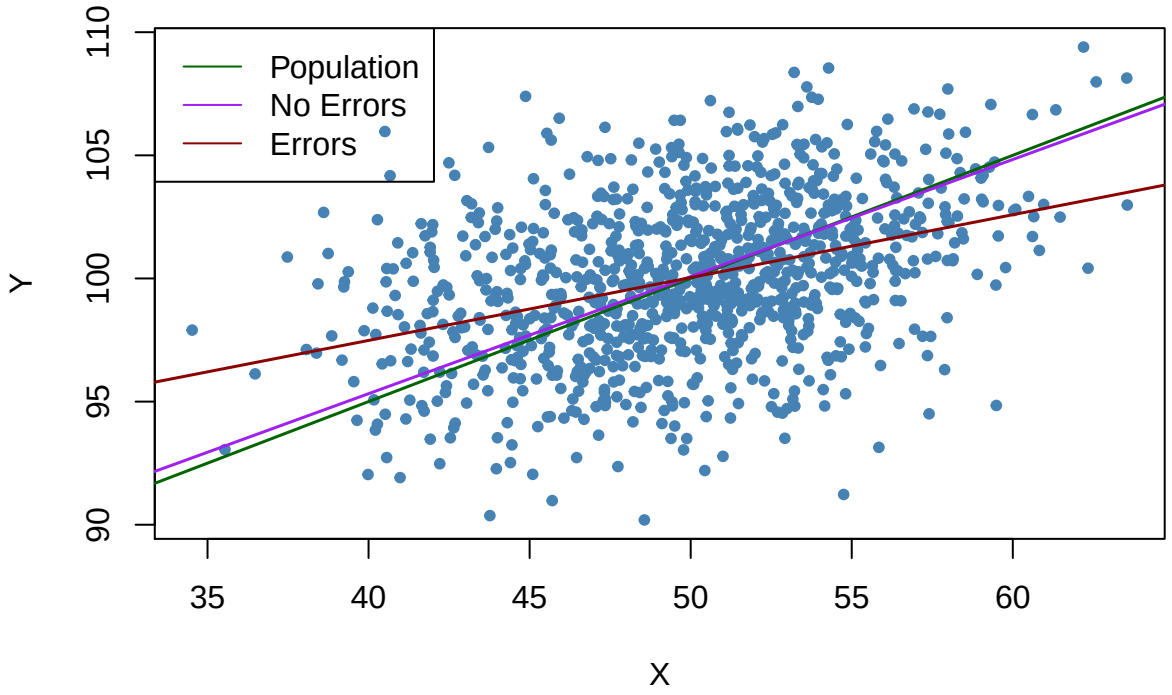
```
me_mod$coefficients
```

```
## (Intercept)           X
##   87.276004    0.255212
```

We visualize the estimation results and compare with the population regression function.

```
# plot sample data
plot(dat$X, dat$Y,
     pch=20,
     col="steelblue",
     xlab = "X",
     ylab = "Y"
     )

# add population regression function
abline(coef = c(75,0.5),
       col = "darkgreen",
```

```
       lwd  = 1.5
      )

# add estimated regression functions
abline(nme_mod,
       col = "purple",
       lwd  = 1.5
       )
abline(me_mod,
       col = "darkred",
       lwd  = 1.5
       )

# add legend
legend("topleft",
       lty = 1,
       col = c("darkgreen", "purple", "darkred"),
       legend = c("Population", "No Errors", "Errors")
       )
```



Notice that in the situation without measurement error, the estimated regression function is close to the population regression function. Things are different when we use the error afflicted data on X: both the estimate for the intercept and the estimate for the coefficient on X differ considerably from results obtained using the "clean" data on X. In particular $\widehat{\beta}_1 = 0.255$. This is evidence for the downward bias. We are in the comfortable situation to know σ_X^2 and σ_w^2. This allows us to correct for bias using (10.2). Using this information we obtain a biased-corrected estimate

$$\frac{\sigma_X^2 + \sigma_w^2}{\sigma_X^2}\widehat{\beta}_1 = \frac{10+10}{10}0.255 = 0.51$$

which is fairly close to the true coefficient in the population regression function, $\beta_1 = 0.5$.

Missing Data and Sample Selection

Key Concept 9.5

Sample Selection Bias

When the sampling process influences the availability of data and when there is a relation of this sampling process to the dependend variable that goes beyond the dependence on the regressors, we say that there is a sample selection bias. This bias is due to correlation between one ore more regressors and the error term. This implies both bias and inconsistency of the OLS estimator.

There are three cases of Sample selection but only one of which poses a threat to internal validity of a regression study. The three cases are:

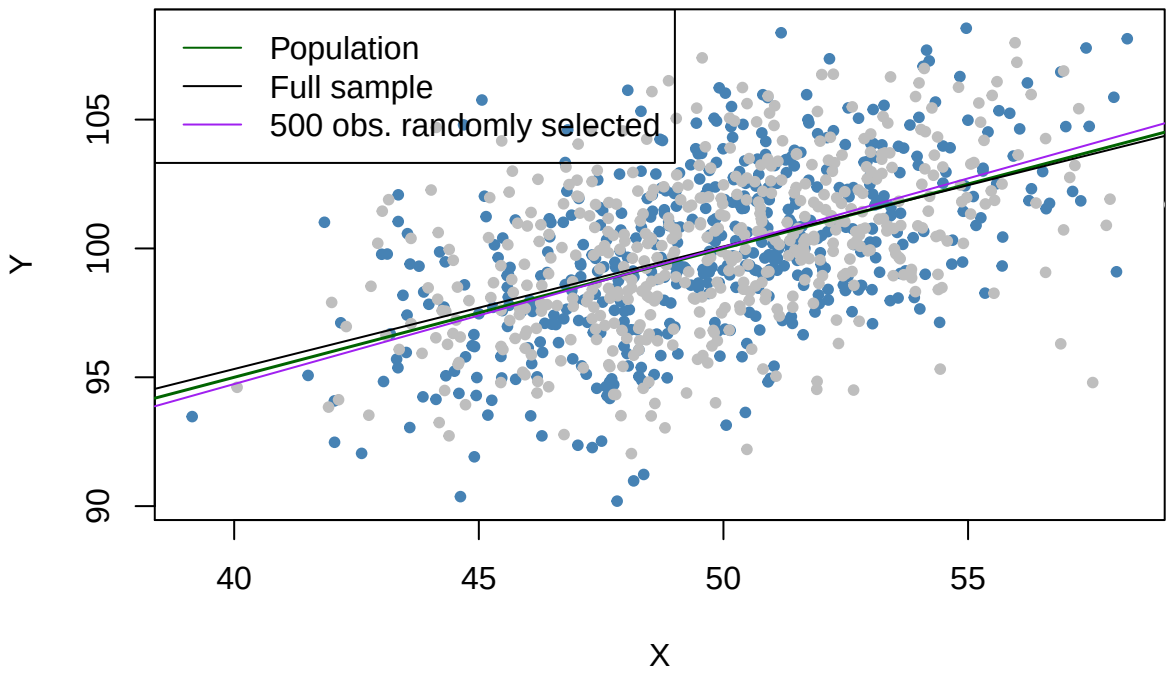
1. Data are missing at random.
2. Data are missing based on the value of a regressor.
3. Data are missing due to a selection process that is releated to the dependent variable.

Let us jump back to the example of variables X and Y distributed as stated in equation (10.4) and illustrate all three cases using R.

If data are missing at random, this is nothing but loosing observations. For example, loosing 50% of sample would be the same as never having seen the (randomly chosen) half of the sample observed. This does not introduce an estimation bias.

```
# random seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50,100),
          sigma = cbind(c(10,5), c(5,10))
          )
  )

colnames(dat) <- c("X","Y")

# mark 500 randomly selected observations
id <- sample(1:1000, size = 500)

plot(dat$X[-id],
     dat$Y[-id],
     col = "steelblue",
     pch = 20,
     xlab = "X",
     ylab = "Y")

points(dat$X[id],
       dat$Y[id],
       col = "gray",
       pch = 20)

# add population regression function
abline(coef = c(75,0.5),
       col = "darkgreen",
       lwd  = 1.5
```

```
      )

# add estimated regression function for full sample
abline(nme_mod)

# estimate model case 1, add regression line
dat <- dat[-id,]

c1_mod <- lm(dat$Y ~ dat$X, data = dat)
abline(c1_mod, col = "purple")

# add legend
legend("topleft",
       lty = 1,
       col = c("darkgreen", "black", "purple"),
       legend = c("Population", "Full sample", "500 obs. randomly selected")
       )
```

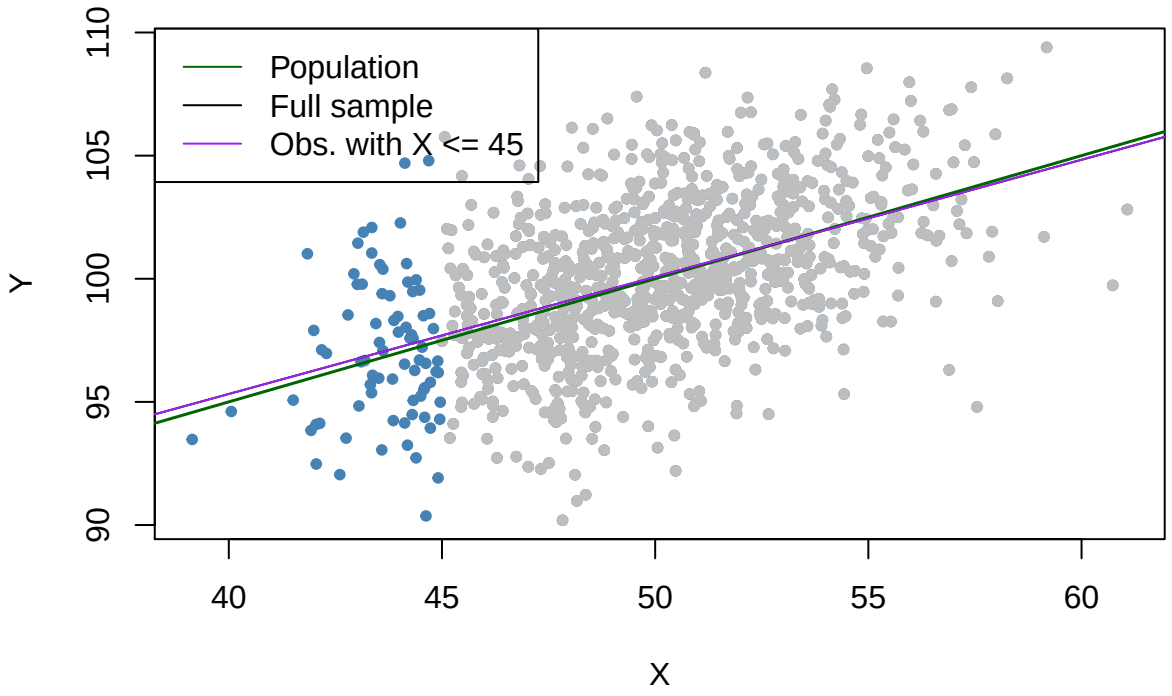


The gray dots are the 500 discarded observations. The estimation results when using the remaining observations deviate only marginally from the results obtained using the full sample.

Selecting data randomly based on the value of a regressor has also the effect of reducing the sample size and does not introduce estimation bias. We will now drop observations with $X > 45$, reestimate the model and compare.

```
# random seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50,100),
          sigma = cbind(c(10,5), c(5,10))
          )
```

```
  )

colnames(dat) <- c("X","Y")

# mark observations
id <- dat$X >= 45

plot(dat$X[-id],
     dat$Y[-id],
     col = "steelblue",
     pch = 20,
     xlab = "X",
     ylab = "Y")

points(dat$X[id],
       dat$Y[id],
       col = "gray",
       pch = 20)

# add population regression function
abline(coef = c(75,0.5),
       col = "darkgreen",
       lwd  = 1.5
      )

# add estimated regression function for full sample
abline(nme_mod)

# estimate model case 1, add regression line
dat <- dat[-id,]

c2_mod <- lm(dat$Y ~ dat$X, data = dat)
abline(c2_mod, col = "purple")

# add legend
legend("topleft",
       lty = 1,
       col = c("darkgreen", "black", "purple"),
       legend = c("Population", "Full sample", "Obs. with X <= 45")
       )
```

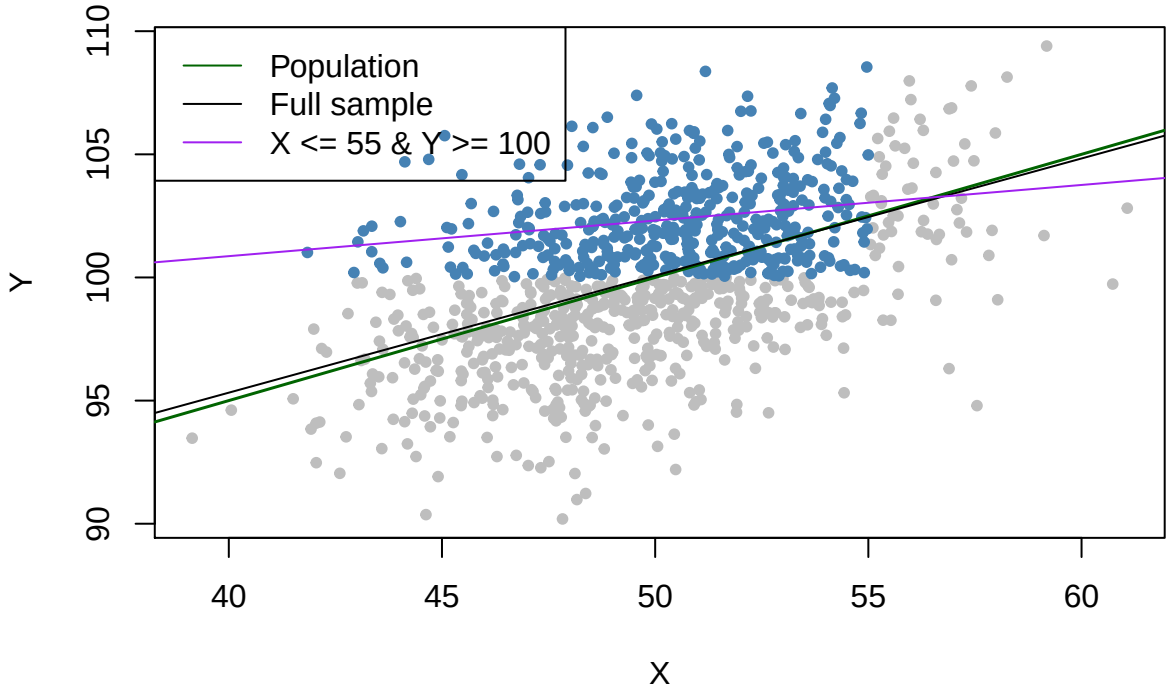


Note that although we dropped more than 90% of all observations, the estimated regression function is very close to the line estimated based on the full sample.

In the third case we face sample selection bias. We can illustrate this by using the selection procedure `dat[which(dat$X <= 55 & dat$Y >= 100)]`

```
# random seed
set.seed(1)

# simulate data
dat <- data.frame(
  rmvnorm(1000, c(50,100),
          sigma = cbind(c(10,5), c(5,10))
          )
  )

colnames(dat) <- c("X","Y")

# mark observations
id <- which(dat$X <= 55 & dat$Y >= 100)

plot(dat$X[-id],
       dat$Y[-id],
       col = "gray",
       pch = 20,
       xlab = "X",
       ylab = "Y")

points(dat$X[id],
     dat$Y[id],
     col = "steelblue",
     pch = 20
     )
```

```
# add population regression function
abline(coef = c(75,0.5),
       col = "darkgreen",
       lwd  = 1.5
      )

# add estimated regression function for full sample
abline(nme_mod)

# estimate model case 1, add regression line
dat <- dat[id,]

c3_mod <- lm(dat$Y ~ dat$X, data = dat)
abline(c3_mod, col = "purple")

# add legend
legend("topleft",
       lty = 1,
       col = c("darkgreen", "black", "purple"),
       legend = c("Population", "Full sample", "X <= 55 & Y >= 100")
       )
```



We see that the selection process leads to biased estimation results. There are methods that allow to correct for sample selection bias. However, these methods are beyond the scope of the book and therefore not considered here. The concept of sample selection bias is summarized in Key Concept 9.5.

Simultaneous Causality

Key Concept 9.6

Simultaneous Causality Bias

So far we have assumed that the changes in the independent variable X are responsible for changes in the dependent variable Y. When the reverse is also true, say that there is simultaneous causality between X and Y. This reverse causality leads to correlaltion between X and the error in the population regression of interest such that the coefficient on X is estimated with a bias.

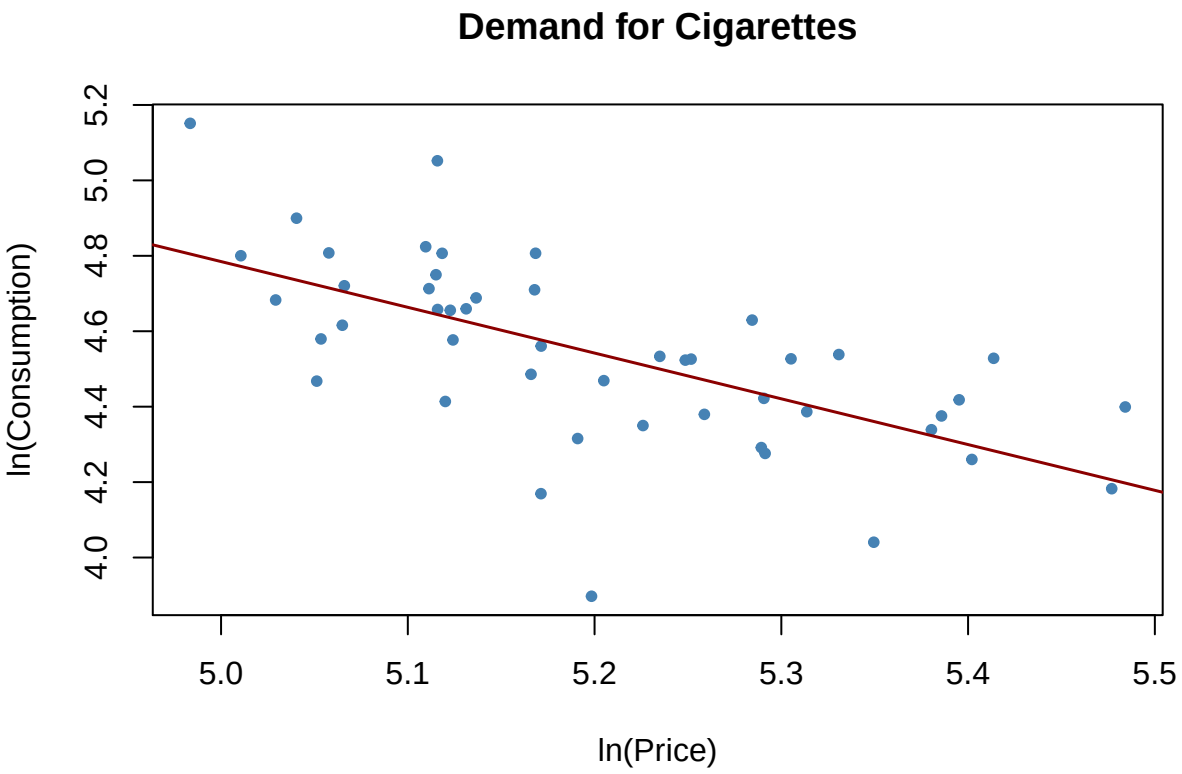
Suppose we are interested in estimating the effect of a 20% increase of cigarettes prices on ciggarettes consumption in the United States using a multiple regression model. This may be investigated using the dataset `CigarettesSW` which is part of the `AER` package. `CigarettesSW` is a panel data set on cigarette consumption for all 48 continental US States from 1985-1995 and provides data on economic indicators and average local prices, taxes and per capita pack consumption.

After loading the dataset, we pick observations for the year 1995 and plot logarithmns of average federal and local excise taxes, `tax`, against pack consumtion, `packs`, and estimate a simple linear regression model.

```
# load data
library(AER)
data("CigarettesSW")
c1995 <- subset(CigarettesSW, year == "1995")

# estimate model
cigcon_mod <- lm(log(packs) ~ log(price), data = c1995)
cigcon_mod
```

```
##
## Call:
## lm(formula = log(packs) ~ log(price), data = c1995)
##
## Coefficients:
## (Intercept)   log(price)
##      10.850       -1.213
```

```
# plot estimated regression line and data
plot(log(c1995$price), log(c1995$packs),
     xlab = "ln(Price)",
     ylab = "ln(Consumption)",
     main = "Demand for Cigarettes",
     pch = 20,
     col = "steelblue"
     )

abline(cigcon_mod,
       col="darkred",
       lwd=1.5)
```



Remember from chapter 8 that, due to the log-log specification, in the population regression the coefficient on the logarithm of price is interpreted as the price elasticity of consumption. The estimated coefficient suggest that a 1% increase in cigarettes prices reduces cigarette consumption by about 1.2%, on average. Have we estimated a demand curve? The answer is no: this is a classic example of simultaneous causality (see Key Concept 9.6) since the observations are market equilibria which are determined by both changes in supply and changes in demand such that the price is correlated with the error term and the OLS estimator is biased. We can neither estimate a demand nor a supply curve consistently using this approach.

We will return to this issue (and the dataset) in chapter 12 which presents instrumental variables regression, an approach that allows consistent estimation when there is simultaneous causality.

10.2.0.1 Sources of Inconsistency of OLS Standard Errors

There are two central threats to computation of consistent OLS standard errors:

1. Heteroskedasticity: implications of heteroskedasticiy have been discussed in chapter 5. Heteroskedasticity robust standard errors as computed by the function `vcovHC()` from the package `sandwich` produce valid standard errors unter heteroskedasticity.
2. Serial correlation: if the population regression error is correlated across observations, we speak of serial correlation. This happens often in applications were repeated observations are used like in panel data studies. As for heteroskedasticity, the `vcovHC()` package can be used to obtain valid standard errors when there is serial correlation.

Inconsistently computed standard errors will produce invalid hypothesis tests and wrong confidence intervals. E.g. when testing the null hypothesis that some model coefficient is zero, we cannot trust the outcome anymore because the test may fail to have the significance level of 5% due to a wrongly computed standard error.

Key Concept 9.7 summarizes all discussed threats to internal validity.

Key Concept 9.7

Threats to Internal Validity of a Regression Study

The five primary threats to internal validity of a multiple regression study are:

1. Omitted variables
2. Misspecefication of functional form
3. Errors in variables (due to measurement error in regressors)
4. Sample selection
5. Simultaneous causality

All these threats lead to failure of the first least squares assumption

$$E(u_i|X_{1i},\dots,X_{ki}) \neq 0$$

so that the OLS estimator is biased *and* inconsistent.

Furthermore, if one does not adjust for heteroskedasticity *and/or* serial correlation, if present, incorrect standard errors may be a threat to internal validity of the study.

10.3 Internal and External Validity When the Regression is Used for Forecasting

Recall the regression of test scores on the student-teacher ratio (STR) performed in chapter 4:

```
linear_model <- lm(score ~ STR, data = CASchools)
linear_model
```

```
##
## Call:
## lm(formula = score ~ STR, data = CASchools)
##
## Coefficients:
## (Intercept)          STR
##      698.93        -2.28
```

The estimated regression function was

$$\widehat{TestScore} = 698.9 - 2.28 \times STR.$$

The book gives the example of a parent moving to a metropolitan area who plans to choose where to live based on quality of local schools. A school disctrics average test score is an adequate mesaure for quality. However, the parent has information on the student-teacher ratio only such that test scores need to be predicted. Although we have established that there is omitted variable bias in this model due to omisssion of variables like student learning opportunities outside school, the share of english learners and so on, `linear_model` may in fact be useful for the parent:

The parent does not care if the coefficien on STR has causal interpretation, she wants STR to explain as much variation in test scores as possible. Therefore, despite the fact that `linear_model` cannot be used to estimate the causal effect of a change in STR on test scores, it can be considered a *reliable predictor* of test scores in general.

Thus, the threats to internal validity as summarized in Key Concept 9.7 are negligible for the parent. This is, as instanced in the book, different for a superindent who has been tasked to take measures that increase test scores: she requires a more reliable model that does not suffer from the threats listed in Key Concept 9.7.

10.4 Example: Test Scores and Class Size

This section discusses internal and external validity of results gained from analyzing the California test score data using regression models.

External Validity of the Study

External validity of the California analysis means that its results can be generalized. If this is possible depends on the population and the setting. Following the book we conduct the same analysis using data for fourth graders in 220 public school districts in Massechusetts in 1998. Just as `CASchools`, the data set `MASchools` is part of the `AER` package. Use the help function (`?MASchools`) to get info on the definitions of all variables contained.

We start by loading the data set and proceed by computing some summary statistics.

```
data("MASchools")
summary(MASchools)
```

```
##    district         municipality          expreg       expspecial
##  Length:220         Length:220         Min.   :2905   Min.   : 3832
##  Class :character   Class :character   1st Qu.:4065   1st Qu.: 7442
##  Mode  :character   Mode  :character   Median :4488   Median : 8354
##                                        Mean   :4605   Mean   : 8901
##                                        3rd Qu.:4972   3rd Qu.: 9722
##                                        Max.   :8759   Max.   :53569
##
##      expbil            expocc          exptot        scratio
##  Min.   :     0   Min.   :    0   Min.   :3465   Min.   : 2.300
##  1st Qu.:     0   1st Qu.:    0   1st Qu.:4730   1st Qu.: 6.100
##  Median :     0   Median :    0   Median :5155   Median : 7.800
##  Mean   :  3037   Mean   : 1104   Mean   :5370   Mean   : 8.107
##  3rd Qu.:     0   3rd Qu.:    0   3rd Qu.:5789   3rd Qu.: 9.800
##  Max.   :295140   Max.   :15088   Max.   :9868   Max.   :18.400
##                                                  NA's   :9
##     special          lunch          stratio          income
##  Min.   : 8.10   Min.   : 0.40   Min.   :11.40   Min.   : 9.686
##  1st Qu.:13.38   1st Qu.: 5.30   1st Qu.:15.80   1st Qu.:15.223
##  Median :15.45   Median :10.55   Median :17.10   Median :17.128
##  Mean   :15.97   Mean   :15.32   Mean   :17.34   Mean   :18.747
##  3rd Qu.:17.93   3rd Qu.:20.02   3rd Qu.:19.02   3rd Qu.:20.376
##  Max.   :34.30   Max.   :76.20   Max.   :27.00   Max.   :46.855
##
##      score4          score8          salary         english
##  Min.   :658.0   Min.   :641.0   Min.   :24.96   Min.   : 0.0000
##  1st Qu.:701.0   1st Qu.:685.0   1st Qu.:33.80   1st Qu.: 0.0000
##  Median :711.0   Median :698.0   Median :35.88   Median : 0.0000
##  Mean   :709.8   Mean   :698.4   Mean   :35.99   Mean   : 1.1177
##  3rd Qu.:720.0   3rd Qu.:712.0   3rd Qu.:37.96   3rd Qu.: 0.8859
##  Max.   :740.0   Max.   :747.0   Max.   :44.49   Max.   :24.4939
##                  NA's   :40      NA's   :25
```

It is fairly easy to replicate key components of table 9.1 of the book using `R`. To be consistent with variable names used in the `CASchools` data set, we do some formatting beforehand.

```
# Customized variables in MASchools
MASchools$score <- MASchools$score4
MASchools$STR <- MASchools$stratio

# Reproduce Table 9.1 of the book
vars <- c("score", "STR", "english", "lunch", "income")

cbind(
  CA_mean = sapply(CASchools[ ,vars], mean),
  CA_sd   = sapply(CASchools[ ,vars], sd),
  MA_mean = sapply(MASchools[ ,vars], mean),
  MA_sd   = sapply(MASchools[ , vars], sd)
  )
```

```
##           CA_mean     CA_sd    MA_mean     MA_sd
## score   654.15655 19.053347 709.827273 15.126474
## STR      19.64043  1.891812  17.344091  2.276666
## english  15.76816 18.285927   1.117676  2.900940
## lunch    44.70524 27.123381  15.315909 15.060068
## income   15.31659  7.225890  18.746764  5.807637
```

The summary statistics reveal that the average test score is higher for school districts in Massachusetts. Notice that the test is somewhat different from that used in California (the Massachusetts test score also includes results for the school subject 'science') so a direct comparison of test scores is not appropriate. We also see that, on average, classes are smaller in Massachusetts than in California and that the average district income, average percentage of english learners as well as the average share of students receiving subsidized lunch differ considerably from the averages computed for California. There are also notable differences in the observed dispersion of variables.

Following the book we examine the relationship between district income and test scores in Massachusetts as we have done before in chapter 8 for the California data and reproduce figure 9.2 of the book.

```
# estimate linear model
Linear_model_MA <- lm(score ~ income, data = MASchools)
Linear_model_MA
```

```
##
## Call:
## lm(formula = score ~ income, data = MASchools)
##
## Coefficients:
## (Intercept)       income
##     679.387        1.624
```

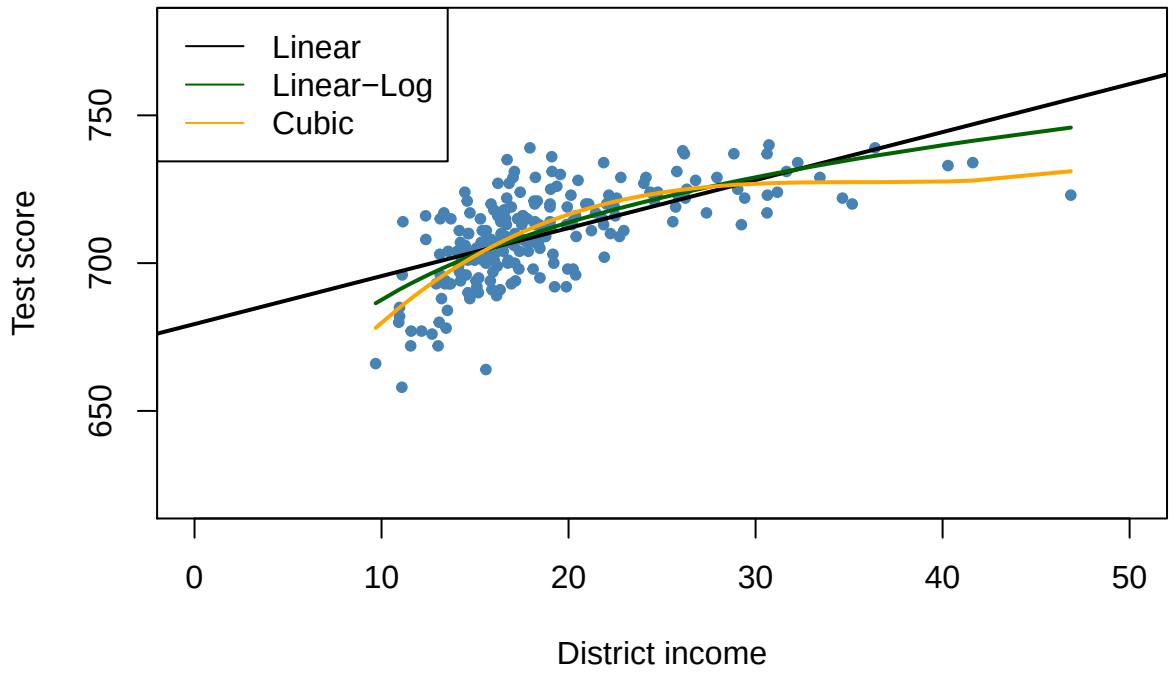
```
# estimate linear-log model
Linearlog_model_MA <- lm(score ~ log(income), data = MASchools)
Linearlog_model_MA
```

```
##
## Call:
## lm(formula = score ~ log(income), data = MASchools)
##
## Coefficients:
## (Intercept)  log(income)
##      600.80        37.71
```

```
# estimate Cubic model
cubic_model_MA <- lm(score ~ I(income) + I(income^2) + I(income^3), data = MASchools)
cubic_model_MA
```

```
##
## Call:
## lm(formula = score ~ I(income) + I(income^2) + I(income^3), data = MASchools)
##
## Coefficients:
## (Intercept)    I(income)  I(income^2)  I(income^3)
##  600.398531    10.635382    -0.296887     0.002762
```

```
# plot data
plot(MASchools$income, MASchools$score,
     pch = 20,
     col = "steelblue",
     xlab = "District income",
     ylab = "Test score",
     xlim = c(0,50),
     ylim = c(620,780))

# add estimated regression line for the linear model
abline(Linear_model_MA, lwd = 2)

# add estimated regression function for Linear-log model
order_id  <- order(MASchools$income)

lines(MASchools$income[order_id],
      fitted(Linearlog_model_MA)[order_id],
      col = "darkgreen",
      lwd = 2)

# add estimated cubic regression function
lines(x = MASchools$income[order_id],
      y = fitted(cubic_model_MA)[order_id],
      col = "orange",
      lwd = 2)

# add a legend
legend("topleft",
       legend = c("Linear","Linear-Log","Cubic"),
       lty = 1,
       col = c("Black","darkgreen","orange")
       )
```



The plot indicates that the cubic specification fits the data best. Interstingly, this is different from the `CASchools` data where the pattern of nonlinearity is better described by the linear-log specification.

We continue by estimating the most of the model specifications used for analysis of the `CASchools` data set in chapter 8 and use `stargazer()` to generate a tabular representation of the regression results.

```
# Add HiEL to 'MASchools'
MASchools$HiEL <- as.numeric(MASchools$english > median(MASchools$english))

# estimate model specifications from table 9.2 of the book
TestScore_MA_mod1 <- lm(score ~ STR, data = MASchools)

TestScore_MA_mod2 <- lm(score ~ STR + english + lunch + log(income), data = MASchools)

TestScore_MA_mod3 <- lm(score ~ STR + english + lunch + income + I(income^2) +
    I(income^3), data = MASchools)

TestScore_MA_mod4 <- lm(score ~ STR + I(STR^2) + I(STR^3) + english + lunch +
    income + I(income^2) + I(income^3), data = MASchools)

TestScore_MA_mod5 <- lm(score ~ STR + I(income^2) + I(income^3) + HiEL:STR +
    lunch + income, data = MASchools)

TestScore_MA_mod6 <- lm(score ~ STR + I(income^2) + I(income^3) + HiEL + HiEL:STR +
    lunch + income, data = MASchools)

# robust estimation of variance-covariance matrices
library(sandwich)

rob_se <- list(sqrt(diag(vcovHC(TestScore_MA_mod1, type = "HC1"))), sqrt(diag(vcovHC(TestScore_MA_mod2,
    type = "HC1"))), sqrt(diag(vcovHC(TestScore_MA_mod3, type = "HC1"))), sqrt(diag(vcovHC(TestScore_MA
    type = "HC1"))), sqrt(diag(vcovHC(TestScore_MA_mod5, type = "HC1"))), sqrt(diag(vcovHC(TestScore_MA
    type = "HC1"))))
```

```
# generate table with stargazer()

library(stargazer)

stargazer(Linear_model_MA, TestScore_MA_mod2, TestScore_MA_mod3, TestScore_MA_mod4,
    TestScore_MA_mod5, TestScore_MA_mod6, type = "latex", se = rob_se, object.names = TRUE,
    model.numbers = FALSE, column.labels = c("(I)", "(II)", "(III)", "(IV)",
        "(V)", "(VI)"))
```

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Thu, Apr 26, 2018 - 10:54:09 % Requires LaTeX packages: rotating

We continue by reproducing the F-statistics and p-values testing exclusion of groups of variables.

```
# F-test model (3)
linearHypothesis(TestScore_MA_mod3,
                 c("I(income^2)=0","I(income^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod3, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + english + lunch + income + I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    215
## 2    213  2 7.7448 0.0005664 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# F-tests model (4)
linearHypothesis(TestScore_MA_mod4,
                 c("STR=0","I(STR^2)=0", "I(STR^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod4, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## STR = 0
## I(STR^2) = 0
## I(STR^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
##     I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
```

Table 10.1:

	Dependent variable:				
	score				
	(I)	(II)	(III)	(IV)	(V)
STR	−1.718***	−0.689**	−0.641**	12.426	−1.018***
	(0.499)	(0.270)	(0.268)	(14.010)	(0.370)
I(STR^2)				−0.680	
				(0.737)	
I(STR^3)				0.011	
				(0.013)	
english		−0.411	−0.437	−0.434	
		(0.306)	(0.303)	(0.300)	
HiEL					−12.561
					(9.793)
lunch		−0.521***	−0.582***	−0.587***	−0.709***
		(0.078)	(0.097)	(0.104)	(0.091)
log(income)		16.529***			
		(3.146)			
income			−3.067	−3.382	−3.867
			(2.353)	(2.491)	(2.488)
I(income^2)			0.164*	0.174*	0.184**
			(0.085)	(0.089)	(0.090)
I(income^3)			−0.002**	−0.002**	−0.002**
			(0.001)	(0.001)	(0.001)
STR:HiEL					0.799
					(0.555)
Constant	739.621***	682.432***	744.025***	665.496***	759.914***
	(8.607)	(11.497)	(21.318)	(81.332)	(23.233)
Observations	220	220	220	220	220
R^2	0.067	0.676	0.685	0.687	0.686
Adjusted R^2	0.063	0.670	0.676	0.675	0.675
Residual Std. Error	14.646 (df = 218)	8.686 (df = 215)	8.607 (df = 213)	8.626 (df = 211)	8.621 (df = 212)
F Statistic	15.616*** (df = 1; 218)	112.284*** (df = 4; 215)	77.232*** (df = 6; 213)	57.803*** (df = 8; 211)	66.023*** (df = 7; 212)

Note: *p<0.1; **p

```
##
##   Res.Df Df      F  Pr(>F)
## 1    214
## 2    211  3 2.8565 0.03809 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(TestScore_MA_mod4,
                 c("I(STR^2)=0", "I(STR^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod4, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(STR^2) = 0
## I(STR^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
##     I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1    213
## 2    211  2 0.4463 0.6406
```

```
linearHypothesis(TestScore_MA_mod4,
                 c("I(income^2)=0","I(income^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod4, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + I(STR^2) + I(STR^3) + english + lunch + income +
##     I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    213
## 2    211  2 7.7487 0.0005657 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# F-tests model (5)
linearHypothesis(TestScore_MA_mod5,
                 c("STR=0","STR:HiEL=0"),
                 vcov. = vcovHC(TestScore_MA_mod5, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## STR = 0
## STR:HiEL = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
##     I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1    214
## 2    212  2 4.0062 0.0196 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(TestScore_MA_mod5,
                 c("I(income^2)=0","I(income^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod5, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
##     I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1    214
## 2    212  2 5.8468 0.003375 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(TestScore_MA_mod5,
                 c("HiEL=0","STR:HiEL=0"),
                 vcov. = vcovHC(TestScore_MA_mod5, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## HiEL = 0
## STR:HiEL = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + HiEL + HiEL:STR + lunch + income + I(income^2) +
##     I(income^3)
##
```

```
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1    214
## 2    212  2 1.5835 0.2077
```

```
# F-test Model (6)
linearHypothesis(TestScore_MA_mod6,
                 c("I(income^2)=0","I(income^3)=0"),
                 vcov. = vcovHC(TestScore_MA_mod6, type="HC1")
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## I(income^2) = 0
## I(income^3) = 0
##
## Model 1: restricted model
## Model 2: score ~ STR + lunch + income + I(income^2) + I(income^3)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1    216
## 2    214  2 6.5479 0.001737 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that, in terms of $\overline{R^2}$, specification (3) which uses a cubic to model the relationship between district income and test scores indeed does perform better the the linear-log specification (2). Using different F-tests on models (4) and (5), we cannot reject the hypothesis that there is no nonlinear relationship between student teacher ratio and test score and also that the share of english learners has an influence on the relationship of interest. Furthermore, regression (6) shows that percentage of english learners can be omitted as a regressor. Because changes in specification made in regressions (4) to (6) do not lead to substantially different results than those of regression (3), we choose model (3) as the most suitable specification.

In compariosn to the California data, we observe the following results:

1. Controlling for the students background characteristics in model specification (2) reduces the coefficient of interest (student-teacher ratio) by roughly 60%. The estimated coefficients are close.

2. The coefficient on student-teacher ratio is always significantly different from zero at the level of 1% for both data sets. This holds for all model specifications under consideration in both studies.

3. The share of english learners in the school district has no important influence on the impact on test score of a change in the student-teacher ratio in both studies.

The biggest difference is that, in contrast to the California results, we do not find evience of a nonlinear relationship between test scores and the student teacher ratio for the Massachusetts data since the correspoding F-tests on model (4) do not reject.

As pointed out by the authors, the test scores for California and Massachusetts have different units because the tests are different. Thus the estimated coefficients on the student-teacher ratio in both regressions cannot be compared before standardizing test scores to the same units as

$$\frac{Testscore - \overline{TestScore}}{\sigma_{TestScore}}$$

for all observations in both datasets and reruning the regressions of interest using the standardized data. One can show that the coefficient on student-teacher ratio in the regression using standardized test scores is the coefficient of the original regression devided by the standard deviation of test scores.

For model (3) of the Massachusetts data, the estimated coeffiecient on the student-teacher ratio is -0.64. A reduction of the student-teacher ratio by two students is predicted to increase test scores by $-2 \cdot (-0.64) = 1.28$ points. Thus we can compute the effect of a reduction of student-teacher ratio by two students on the standardized test scores as follows:

```
TestScore_MA_mod3$coefficients[2] / sd(MASchools$score) * (-2)
```

```
##        STR
## 0.08474001
```

Thus for Massachussetts, the predicted increase of test scores due to a reduction of the student-teacher ratio by two students is 0.085 standard deviation of the distribution of the observed distribution of test scores.

Beholding the linear specification (2) for California, the estimated coefficient on the student-teacher ratio is -0.73 so the predicted increase of test scores induced by a reduction of the student-teacher ratio by two students is $-0.73 * (-2) = 1.46$. We use `R` to compute the predicted change in standard deviation units:

```
TestScore_mod2$coefficients[2] / sd(CASchools$score) * (-2)
```

```
##        STR
## 0.07708103
```

This shows that the the predicted increase of test scores due to a reduction of the student-teacher ratio by two students is 0.077 standard deviation of the distribution of the observed distribution of test scores for the California data.

In terms of standardized test scores, the predicted change is essentially the same for school districts in California and Massachusetts.

Altogether, the results are in favour of the presumption that inference made using data on Californian elementary school districts is externally valid.

Internal Validity of the Study

External validity of the study **does not** ensure their internal validity. Though the model specification choosen improves upon a simple linear regression model, internal validity may still be violated due to some of the threats listed in Key Concept 9.7 threats. These threats are:

- Omitted variable bias
- Misspecification of functional form
- Errors in variables
- Sample selection issues
- Simultaneous causality
- Heteroskedasticity
- Correlation of errors across observations

Consult the book for a in depth discussion of these threats in view of the test score studies.

Summary

We have found that there *is* a statistically significant effect of the student-teacher ratio on test scores though it is quite small. However, it remains unclear if we have indeed estimated the causal effect of inerest since — although our approach includes control variables, takes into account nonlinearities in the population regression function and statistical inference is made using robust standard errors — the results might still be biased for example if there are omitted factors which we have not considered. Thus *internal validity* of the study remains questionable. As we have concluded from comparison with the analysis of the Massachusetts dataset, this result is *externally valid*.

The following chapters adress techniques that can be remedies to all the threats to internal validity named in Key Concept 9.7 if multiple regression alone is insufficient. This includes regression using panel data and approaches that employ instrumental variables.

Chapter 11

Regression with Panel Data

Regression using panel data may mititgate omitted variable bias when there is no information on variables available that correlate with both the regressors of interest and the independent variable if these variables are constant in the time dimension or across entities. Therefore, provided that panel data is availabe, panel regression methods may improve upon multiple regression models which, as discussed in chapter 9, produce results that are not internally valid in such a setting.

This chapter covers the following topics:

- Notation for panel data
- Fixed effects regression using time and/or entity fixed effects
- Computation of standard errors in fixed effects regression models

Following the book, for applications we make use of the data set `Fatalities` from the `AER` package which is a panel data set reporting annually state level observations on US traffic fatalities for 1982 through 1988. The applications are concered with the question if there are effects of alcohol taxes and drunk driving laws on road fatalities and, if present, *how strong* these effects are.

For this purpose we will introduce a convenient `R` function that enables to estimate linear panel regression models, the function `plm()` which comes with the package `plm`. Usage of `plm()` is very similar as for the `lm()` function which we have used throughout the previous five chapters for estimation of simple and multiple regression models.

11.1 Panel Data

Key Concept 10.1

Notation for Panel Data

In contrast to cross-section data where we have observations on n subjects (entities), panel data has observations on n entities at $T \geq 2$ time periods. This is denoted

$$(X_{it}, Y_{it}),\ i = 1, \dots, n \quad \text{and} \quad t = 1, \dots, T.$$

The index i refers to the entity beeing observed while t refers to the time period.

Sometimes panel data is also called longitudinal data as it adds a temporal dimension the what we call cross-sectional data. Let us have a glimpse at the data set `Fatalities` by checking its structure and listing the first few observations.

```
# load package and data
library(AER)
data(Fatalities)

# obtain dimension and inspect structure
is.data.frame(Fatalities)
```

```
## [1] TRUE
```

```
dim(Fatalities)
```

```
## [1] 336  34
```

```
str(Fatalities)
```

```
## 'data.frame':    336 obs. of  34 variables:
##  $ state       : Factor w/ 48 levels "al","az","ar",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ year        : Factor w/ 7 levels "1982","1983",..: 1 2 3 4 5 6 7 1 2 3 ...
##  $ spirits     : num  1.37 1.36 1.32 1.28 1.23 ...
##  $ unemp       : num  14.4 13.7 11.1 8.9 9.8 ...
##  $ income      : num  10544 10733 11109 11333 11662 ...
##  $ emppop      : num  50.7 52.1 54.2 55.3 56.5 ...
##  $ beertax     : num  1.54 1.79 1.71 1.65 1.61 ...
##  $ baptist     : num  30.4 30.3 30.3 30.3 30.3 ...
##  $ mormon      : num  0.328 0.343 0.359 0.376 0.393 ...
##  $ drinkage    : num  19 19 19 19.7 21 ...
##  $ dry         : num  25 23 24 23.6 23.5 ...
##  $ youngdrivers: num  0.212 0.211 0.211 0.211 0.213 ...
##  $ miles       : num  7234 7836 8263 8727 8953 ...
##  $ breath      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ jail        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
##  $ service     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 2 ...
##  $ fatal       : int  839 930 932 882 1081 1110 1023 724 675 869 ...
##  $ nfatal      : int  146 154 165 146 172 181 139 131 112 149 ...
##  $ sfatal      : int  99 98 94 98 119 114 89 76 60 81 ...
##  $ fatal1517   : int  53 71 49 66 82 94 66 40 40 51 ...
##  $ nfatal1517  : int  9 8 7 9 10 11 8 7 7 8 ...
##  $ fatal1820   : int  99 108 103 100 120 127 105 81 83 118 ...
##  $ nfatal1820  : int  34 26 25 23 23 31 24 16 19 34 ...
##  $ fatal2124   : int  120 124 118 114 119 138 123 96 80 123 ...
##  $ nfatal2124  : int  32 35 34 45 29 30 25 36 17 33 ...
##  $ afatal      : num  309 342 305 277 361 ...
##  $ pop         : num  3942002 3960008 3988992 4021008 4049994 ...
##  $ pop1517     : num  209000 202000 197000 195000 204000 ...
##  $ pop1820     : num  221553 219125 216724 214349 212000 ...
##  $ pop2124     : num  290000 290000 288000 284000 263000 ...
##  $ milestot    : num  28516 31032 32961 35091 36259 ...
##  $ unempus     : num  9.7 9.6 7.5 7.2 7 ...
##  $ emppopus    : num  57.8 57.9 59.5 60.1 60.7 ...
##  $ gsp         : num  -0.0221 0.0466 0.0628 0.0275 0.0321 ...
```

```
# list first few observations
head(Fatalities)
```

```
##   state year spirits unemp   income   emppop  beertax baptist  mormon
## 1    al 1982    1.37  14.4 10544.15 50.69204 1.539379 30.3557 0.32829
```

```
## 2    al 1983    1.36  13.7 10732.80 52.14703 1.788991 30.3336 0.34341
## 3    al 1984    1.32  11.1 11108.79 54.16809 1.714286 30.3115 0.35924
## 4    al 1985    1.28   8.9 11332.63 55.27114 1.652542 30.2895 0.37579
## 5    al 1986    1.23   9.8 11661.51 56.51450 1.609907 30.2674 0.39311
## 6    al 1987    1.18   7.8 11944.00 57.50988 1.560000 30.2453 0.41123
##   drinkage     dry youngdrivers    miles breath jail service fatal nfatal
## 1    19.00 25.0063     0.211572 7233.887     no   no      no   839    146
## 2    19.00 22.9942     0.210768 7836.348     no   no      no   930    154
## 3    19.00 24.0426     0.211484 8262.990     no   no      no   932    165
## 4    19.67 23.6339     0.211140 8726.917     no   no      no   882    146
## 5    21.00 23.4647     0.213400 8952.854     no   no      no  1081    172
## 6    21.00 23.7924     0.215527 9166.302     no   no      no  1110    181
##   sfatal fatal1517 nfatal1517 fatal1820 nfatal1820 fatal2124 nfatal2124
## 1     99        53          9        99         34       120         32
## 2     98        71          8       108         26       124         35
## 3     94        49          7       103         25       118         34
## 4     98        66          9       100         23       114         45
## 5    119        82         10       120         23       119         29
## 6    114        94         11       127         31       138         30
##    afatal     pop  pop1517  pop1820  pop2124 milestot unempus emppopus
## 1 309.438 3942002 208999.6 221553.4 290000.1    28516     9.7     57.8
## 2 341.834 3960008 202000.1 219125.5 290000.2    31032     9.6     57.9
## 3 304.872 3988992 197000.0 216724.1 288000.2    32961     7.5     59.5
## 4 276.742 4021008 194999.7 214349.0 284000.3    35091     7.2     60.1
## 5 360.716 4049994 203999.9 212000.0 263000.3    36259     7.0     60.7
## 6 368.421 4082999 204999.8 208998.5 258999.8    37426     6.2     61.5
##           gsp
## 1 -0.02212476
## 2  0.04655825
## 3  0.06279784
## 4  0.02748997
## 5  0.03214295
## 6  0.04897637
```

```
# summarize variables 'state' and 'year'
summary(Fatalities[,c(1,2)])
```

```
##      state       year
##  al     :  7   1982:48
##  az     :  7   1983:48
##  ar     :  7   1984:48
##  ca     :  7   1985:48
##  co     :  7   1986:48
##  ct     :  7   1987:48
##  (Other):294   1988:48
```

We find that the data set consists of 336 observations on 34 variables. Notice that the variable `state` is a factor variable with 48 levels (one for each of the 48 contigous US states). The varaible `year` is also a factor variable that has 7 levels identifying the time period when the observation was made which gives us $7 \times 48 = 336$ observations in total. Since all variables are observed for all entities and time periods we say the the panel is **balanced**. If there were missing data for at least one entities in at least one time period we would call the panel **unbalanced**.

Example: Traffic Deaths and Alcohol Taxes

Coming to the question of how the alcohol taxes and traffic fatalities are related, we start by reproducing figure 10.1 of the book. First we estimate simple regressions using data for years 1982 and 1988 that model the relationship between beer tax (adjusted for 1988 dollars) and the traffic fatility rate. Beforehand we define the latter as the number of fatalitites per 10000 inhabitants and choose subsets of the observations made for years 1982 and 1988. Afterwards, we plot both subsets of data and the corresponding estimated regression functions.

```
# define fatality rate
Fatalities$fatal_rate <- Fatalities$fatal / Fatalities$pop * 10000

# subset data
Fatalities1982 <- subset(Fatalities, year == "1982")
Fatalities1988 <- subset(Fatalities, year == "1988")
```

```
# estimate simple regression models using 1982 and 1988 data
library(lmtest)
library(sandwich)

fatal1982_mod <- lm(fatal_rate ~ beertax, data = Fatalities1982)
fatal1988_mod <- lm(fatal_rate ~ beertax, data = Fatalities1988)

coeftest(fatal1982_mod, vcov. = vcovHC(fatal1982_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.01038    0.14957 13.4408   <2e-16 ***
## beertax      0.14846    0.13261  1.1196   0.2687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(fatal1988_mod, vcov. = vcovHC(fatal1988_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  1.85907    0.11461 16.2205 < 2.2e-16 ***
## beertax      0.43875    0.12786  3.4314  0.001279 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

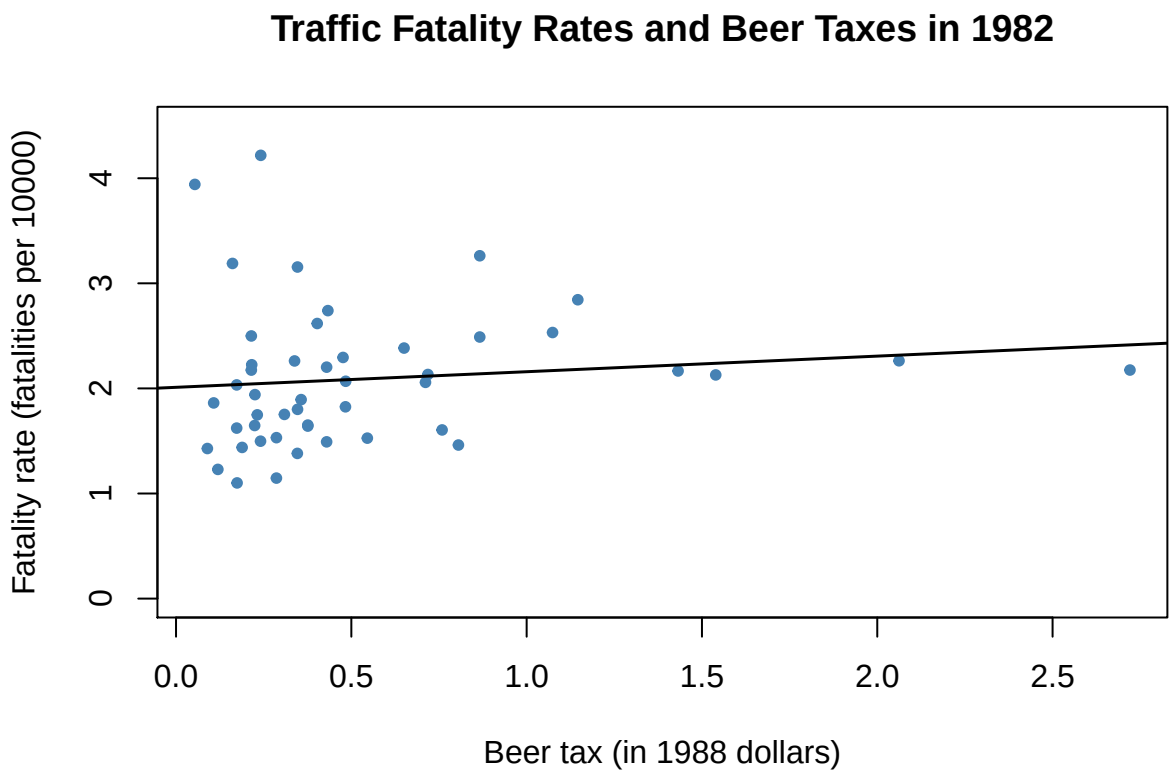
The estimated regression functions are:

$$\widehat{FatalityRate} = \underset{(0.15)}{2.01} + \underset{(0.13)}{0.15} \times BeerTax \quad \text{(1982 data)},$$

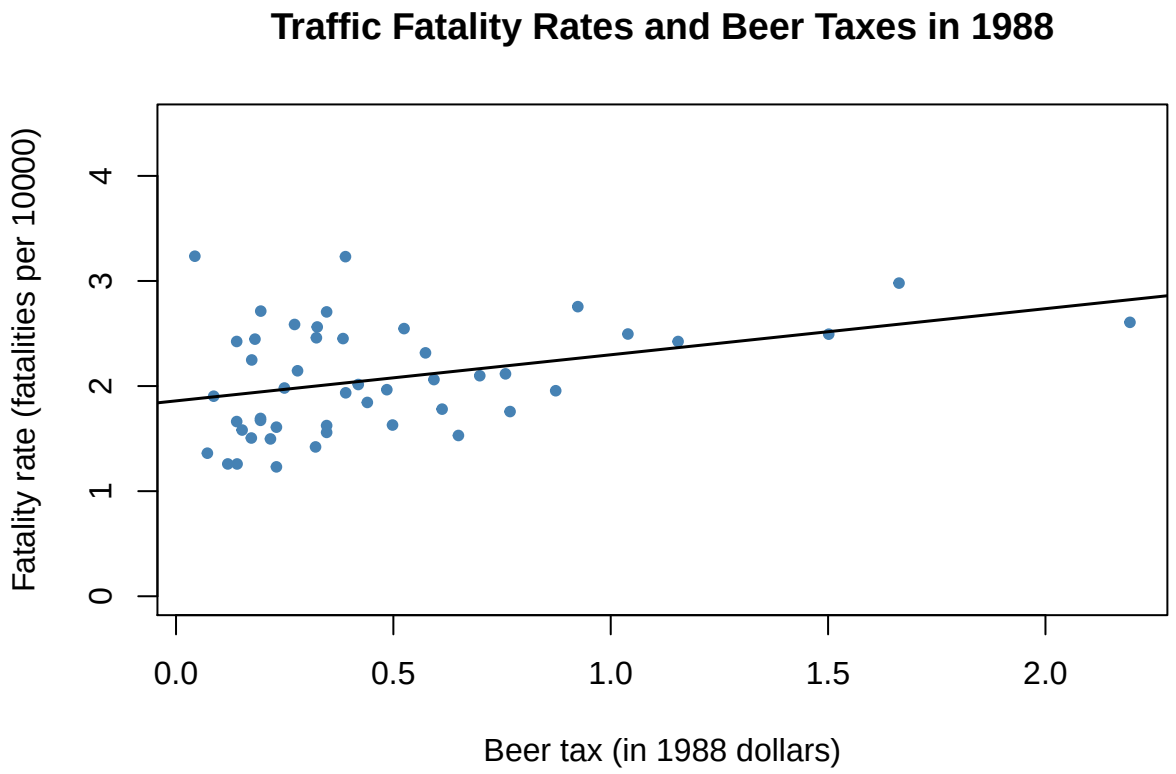
$$\widehat{FatalityRate} = \underset{(0.11)}{1.86} + \underset{(0.13)}{0.44} \times BeerTax \quad \text{(1988 data)}$$

```
# plot observations of interest and add estimated regression line for 1982 data
plot(x = Fatalities1982$beertax,
     y = Fatalities1982$fatal_rate,
```

```
     xlab = "Beer tax (in 1988 dollars)",
     ylab = "Fatality rate (fatalities per 10000)",
     main = "Traffic Fatality Rates and Beer Taxes in 1982",
     ylim = c(0, 4.5),
     pch = 20,
     col = "steelblue")

abline(fatal1982_mod, lwd=1.5)
```




```
# plot observations of interest and add estimated regression line for 1988 data
plot(x = Fatalities1988$beertax,
     y = Fatalities1988$fatal_rate,
     xlab = "Beer tax (in 1988 dollars)",
     ylab = "Fatality rate (fatalities per 10000)",
     main = "Traffic Fatality Rates and Beer Taxes in 1988",
     ylim = c(0, 4.5),
     pch = 20,
     col = "steelblue")

abline(fatal1988_mod, lwd = 1.5)
```



In both plots, each point represents observations of beer tax and fatality rate for a given state in the respective year. The regression results indicate a postive relationship between beer tax and fatality rate for both years wherby the estimated coefficient on beer tax for the 1988 data is almost three times as large as for the 1988 data set. This is contrary to our expectations: alcohol taxes are supposed to *lower* the rate of traffic fatalities. As known from chapter 6, this is possibly due to omitted variable bias since both models do not include any covariates (e.g. economic conditions) which could be corrected for using a multiple regression approach. However, both models cannot account for omitted *unobservable* factors that differ from state to state but can be assumed constant over the observation span like the populations attitude towas drunk driving. As shown in the next section, panel data allow us to hold such factors constant.

11.2 Panel Data with Two Time Periods: "Before and Afer" Comparisons

Suppose there are only $T = 2$ time periods $t = 1982, 1988$. This allows us to analyze differences in changes of the the fatality rate from year 1982 to 1988. We start by considering the population regression function

$$FatalityRate_{it} = \beta_0 + \beta_1 BeerTax_{it} + \beta_2 Z_i + u_{it}$$

where the Z_i are state specific characteristics that differ between states but are *constant over time*. For $t = 1982, 1988$ we have

$$\begin{aligned} FatalityRate_{i1982} &= \beta_0 + \beta_1 BeerTax_{i1982} + \beta_2 Z_i + u_{i1982}, \\ FatalityRate_{i1988} &= \beta_0 + \beta_1 BeerTax_{i1988} + \beta_2 Z_i + u_{i1988}. \end{aligned}$$

We can eliminate the Z_i by regressing the difference in the fatality rate between 1988 and 1982 on the difference in beer tax between those years:

$$FatalityRate_{i1988} - FatalityRate_{i1982} = \beta_1 (BeerTax_{i1988} - BeerTax_{i1982}) + u_{i1988} - u_{i1982}$$

Using this regression model we can obtain an estimate for β_1 without worrying about a possible bias due to omission of the Z_i since these influences are eliminated from the model. Let us use R to estimate a regression based on the differenced data and plot the estimated regression function.

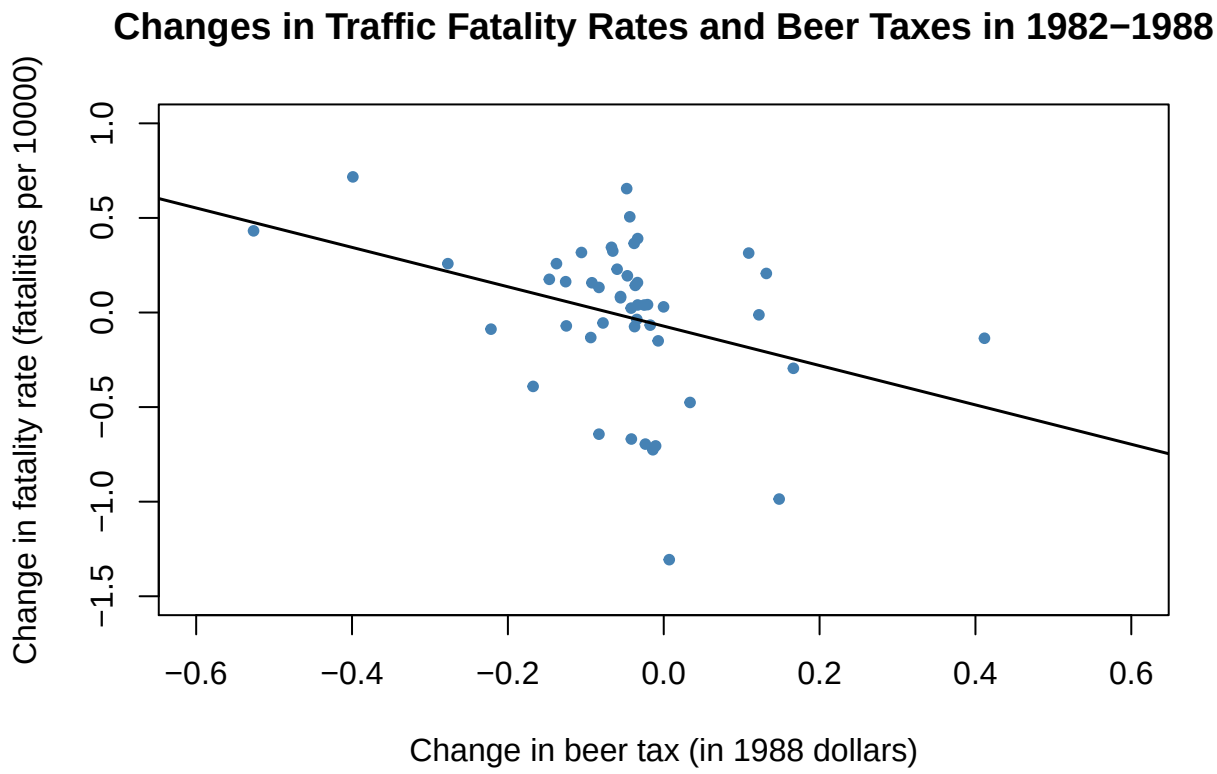
```
# differences
diff_fatal_rate <- Fatalities1988$fatal_rate - Fatalities1982$fatal_rate
diff_beertax <- Fatalities1988$beertax - Fatalities1982$beertax
# estimate regression on differenced data
fatal_diff_mod <- lm(diff_fatal_rate ~ diff_beertax)

coeftest(fatal_diff_mod, vcov = vcovHC(fatal_diff_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.072037   0.065355 -1.1022 0.276091
## diff_beertax -1.040973   0.355006 -2.9323 0.005229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain the OLS estimated regression function

$$\widehat{FatalityRate_{i1988} - FatalityRate_{i1982}} = \underset{(0.065)}{-0.072} \underset{(0.36)}{-1.04} \times (BeerTax_{i1988} - BeerTax_{i1982}).$$

```
# plot differenced data
plot(x = diff_beertax,
     y = diff_fatal_rate,
     xlab = "Change in beer tax (in 1988 dollars)",
     ylab = "Change in fatality rate (fatalities per 10000)",
     main = "Changes in Traffic Fatality Rates and Beer Taxes in 1982-1988",
     xlim = c(-0.6, 0.6),
     ylim = c(-1.5, 1),
     pch = 20,
     col = "steelblue")

# add regression line to plot
abline(fatal_diff_mod, lwd = 1.5)
```



The intercept allows for a change in the mean fatality rate in the time between 1982 and 1988 in the absence of a change in the beer tax. We observe that the estimated coefficient on beer tax is now negative and significantly different from zero at the level of 5%. Its interpretation is that raising the beer tax by \$1 causes trafic fatalities to decrease by 1.04 per 10000 people. This is rather large as the average fatality rate is approximately 2 persons per 10000 people.

```
# compute mean fatality rate over all states for all time periods
mean(Fatalities$fatal_rate)
```

```
## [1] 2.040444
```

Again, this outcome is likely to be a consequence of omitting factors that influence the fatality rate, are correlated with the beer tax *and* change over time. The message is that we need to be more careful and control for such factors before drawing conclusion about the effect of a raise in beer taxes.

Further, note that the approach presented in this section discards information for years 1983 to 1987. A method that allows to use data for more than $T = 2$ time periods and allows us to add control variables is the fixed effects regression approach.

11.3 Fixed Effects Regression

Consider the panel regression model

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \beta_2 Z_i + u_{it}$$

where the Z_i are unobserved time-invariant heterogeneities across entities $i = 1, \dots, n$. We are interested in estimating β_1, the effect on Y of a change in X holding constant Z. Letting $\alpha_i = \beta_0 + \beta_2 Z_i$ we obtain

$$Y_{it} = \alpha_i + \beta_1 X_{it} + u_{it}. \tag{11.1}$$

Having invidual specific intercepts α_i, $i = 1, \ldots, n$ where each of which can be understood as the fixed effect of entity i, (11.1) is called the *fixed effects regression model*. The variation in the α_i, $i = 1, \ldots, n$ comes from the Z_i. (11.1) can be rewritten as a regression model containing $n - 1$ dummy regressors and a constant:

$$Y_{it} = \beta_i + \beta_1 X_{it} + \gamma_2 D2_i + \gamma_3 D3_i + \cdots + \gamma_n Dn_i + u_{it}. \tag{11.2}$$

Model (11.2) has n different intercepts — one for every entity. Models (11.1) and (11.2) are equivalnt representations of the fixed effects regression model.

The model can be generalized to contain more that just one determinant of Y that are correlated with X and change over time. Key Concept 10.2 presents the generalized fixed effects regression model.

Key Concept 10.2

The Fixed Effects Regression Model

The fixed effects regression model is

$$Y_{it} = \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \alpha_i + u_{it} \tag{11.3}$$

with $i = 1, \ldots, n$ and $t = 1, \ldots, T$. The α_i are entity-specific intercepts that capture heterogeneities across entities. An equivalent representation of this model is given by

$$Y_{it} = \beta_0 + \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \gamma_2 D2_i + \gamma_3 D3_i + \cdots + \gamma_n Dn_i + u_{it} \tag{11.4}$$

where the $D2_i, D3_i, \ldots, Dn_i$ are dummy variables.

Estimation and Inference

Software packages use a so-called "entity-demeaned" OLS algorithm that is computationally more efficient than estimating regression models with $k + n$ regressors as need for models (11.3) and (11.4).

Taking averages on both sides of (11.1) we obtain

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} Y_{it} &= \beta_1 \frac{1}{n}\sum_{i=1}^{n} X_{it} + \frac{1}{n}\sum_{i=1}^{n} a_i + \frac{1}{n}\sum_{i=1}^{n} u_{it} \\ \overline{Y} &= \beta_1 \overline{X}_i + \alpha_i + \overline{u}_i \end{aligned}$$

substraction from (11.1) gives us

$$Y_{it} - \overline{Y}_i = \beta_1 (X_{it} - \overline{X}_i) + (u_{it} - \overline{u}_i) \tag{11.5}$$

$$\widetilde{Y}_{it} = \beta_1 \widetilde{X}_{it} + \widetilde{u}_{it}. \tag{11.6}$$

In this regression model, the OLS estimate of the parameter of interest β_1 is equal to the estimate obtained using (11.2) — without the need to estimate $n-1$ dummies and an intercept.

Thus, we have two ways to estimate β_1 in the fixed effects regression:

1. OLS regression of the dummy regression model (11.2)
2. OLS regression using the entity demeaned data (11.6)

Provided the fixed effects regression assumptions stated in Key Concept 10.3 hold, the sampling distribution of the OLS estimator in the fixed effects regression model has a normal distribution in large samples. The variance can be estimated and we can compute standard errors, t-statistics and confidence intervals for model coefficients. In the next section, we will see how to estimate a fixed effects model using `R` and how to obtain a model summary that makes use heteroskedasticity robust standard errors. Thereby we leave aside complicated formulas of the estimators. See chapter 10.5 and appendix 10.2 for a discussion of the theoretical aspects.

Application to Traffic Deaths

Following Key Concept 10.2, the simple fixed effects model for estimation of the relation between traffic fatality rates and the beer taxes is

$$\widehat{FatalityRate}_{it} = \beta_1 BeerTax_{it} + StateFixedEffects + u_{it}, \tag{11.7}$$

the regression of the traffic fatalaty rate on beer tax and 48 binary regressor — one for each state.

We can simply use the function `lm()` to obtain an estimate for β_1.

```
fatal_fe_lm_mod <- lm(fatal_rate ~ beertax + state - 1, data = Fatalities)
fatal_fe_lm_mod
```

```
##
## Call:
## lm(formula = fatal_rate ~ beertax + state - 1, data = Fatalities)
##
## Coefficients:
## beertax  stateal  stateaz  statear  stateca  stateco  statect  statede
## -0.6559   3.4776   2.9099   2.8227   1.9682   1.9933   1.6154   2.1700
## statefl  stategа  stateid  stateil  statein  stateia  stateks  stateky
##  3.2095   4.0022   2.8086   1.5160   2.0161   1.9337   2.2544   2.2601
## statela  stateme  statemd  statema  statemi  statemn  statems  statemo
##  2.6305   2.3697   1.7712   1.3679   1.9931   1.5804   3.4486   2.1814
## statemt  statene  statenv  statenh  statenj  statenm  stateny  statenc
##  3.1172   1.9555   2.8769   2.2232   1.3719   3.9040   1.2910   3.1872
## statend  stateoh  stateok  stateor  statepa  stateri  statesc  statesd
##  1.8542   1.8032   2.9326   2.3096   1.7102   1.2126   4.0348   2.4739
## statetn  statetx  stateut  statevt  stateva  statewa  statewv  statewi
##  2.6020   2.5602   2.3137   2.5116   2.1874   1.8181   2.5809   1.7184
## statewy
##  3.2491
```

As discussed in the previous section, it is also possible to estimate β_1 by

$$\widetilde{FatalityRate} = \beta_1 \widetilde{BeerTax}_{it} + u_{it},$$

an OLS regression based on entity demeaned data.

```
# demeaned data
Fatalities_demeaned <- with(Fatalities,
            data.frame(fatal_rate = fatal_rate - ave(fatal_rate, state),
            beertaxs = beertax - ave(beertax, state)))

# estimate regression
summary(lm(fatal_rate ~ beertax - 1, data = Fatalities_demeaned))
```

Equivalently it possible to use `plm()` from the package with the same name. If not already done, install the package `plm` using `install.packages("plm")` and attach it by calling `library(plm)`.

```
# install and load the 'plm' package
## install.packages("plm")
library(plm)
```

As for `lm()` we have to specify the regression formula and the data to be used in our call of `plm()`. Additionally, it is required to pass a vector of names of entity and time id variables to the `index` argument. For `Fatalities`, the id variable for entities is named `state` and the time id variable is `year`. Since the fixed effects estimator is also called the *within* estimator, we set `model = "within"`. The `coeftest()` function allows to obtain significance based on robust standard errors.

```
# estimate the fixed effects regression with plm()
fatal_fe_mod <- plm(fatal_rate ~ beertax,
                    data = Fatalities,
                    index = c("state", "year"),
                    model = "within")

# summary using robust standard errors
coeftest(fatal_fe_mod, vcov. = vcovHC(fatal_fe_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
##         Estimate Std. Error t value Pr(>|t|)
## beertax -0.65587    0.28880  -2.271  0.02388 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Agian, we find that the estimated coefficient is -0.6559. Notice that `plm()` uses the entity-demeaned OLS algorithm as no coefficients for dummy variables are reported. The estimated regression function is

$$\widehat{FatalityRate} = \underset{(0.29)}{-0.66} \times BeerTax + StateFixedEffects. \tag{11.8}$$

We conclude that the coefficient on $BeerTax$ is nagative and highly significant. The interpretation is that the estimated reduction in traffic fatalities due to a \$1 increase in the real beer tax is 0.66 per 10000 people which is still pretty high. Though including state fixed effects eliminates the risk of a bias due to omitted factors that vary across states but not over time, we suspect that there are other omitted varibales that vary over time and thus cause a bias.

11.4 Regression with Time Fixed Effects

Controlling for variables that are constant across entities but vary over time can be realized by including time fixed effects. If there are **only time fixed effects**, the fixed effects regression model becomes

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \delta_2 B2_t + \cdots + \delta_T BT_t + u_{it},$$

where only $T - 1$ dummies are included ($B1$ is omitted) since the model includes an intercept. This model eliminates only omitted variables bias from unobserved varaibles that evolve over time but are constant across entities.

In some applications it is meaningful to include entity and time fixed effects. The **entity and time fixed effects** regression model is

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \gamma_2 D2_i + \cdots + \gamma_n DT_i + \delta_2 B2_t + \cdots + \delta_T BT_t + u_{it}.$$

The combined model allows to eliminate bias from unobservables that change over time but are constant over entities and controls for factors that differ across entities but are constant over time. Such models can be estimated using and OLS algorithm that is implemented in R.

The following code chunk shows how to estimate the combined entity and time fixed effects model of the relation between fatalities and beer tax,

$$FatalityRate_{it} = \beta_1 BeerTax_{it} + StateEffects + TimeFixedEffects + u_{it}$$

using both, `lm()` and `plm()`. It is straightforward estimate this regression with `lm()` since it is just an extension of (11.7). We only have to adjust the `formula` argument by adding the additional regressor `year` for time fixed effects. In our call of `plm()` we set another argument `effect = "twoways"` for inclusion of entity *and* time dummies.

```
# Estimate combind time and entity fixed effects regression model
# via lm()
fatal_tefe_lm_mod <- lm(fatal_rate ~ beertax + state + year - 1, data = Fatalities)
fatal_tefe_lm_mod
```

```
##
## Call:
## lm(formula = fatal_rate ~ beertax + state + year - 1, data = Fatalities)
##
## Coefficients:
##  beertax   stateal   stateaz   statear   stateca   stateco   statect
## -0.63998   3.51137   2.96451   2.87284   2.02618   2.04984   1.67125
##  statede   statefl   stategа   stateid   stateil   statein   stateia
##  2.22711   3.25132   4.02300   2.86242   1.57287   2.07123   1.98709
##  stateks   stateky   statela   stateme   statemd   statema   statemi
##  2.30707   2.31659   2.67772   2.41713   1.82731   1.42335   2.04488
##  statemn   statems   statemo   statemt   statene   statenv   statenh
##  1.63488   3.49146   2.23598   3.17160   2.00846   2.93322   2.27245
##  statenj   statenm   stateny   statenc   statend   stateoh   stateok
##  1.43016   3.95748   1.34849   3.22630   1.90762   1.85664   2.97776
##  stateor   statepa   stateri   statesc   statesd   statetn   statetx
##  2.36597   1.76563   1.26964   4.06496   2.52317   2.65670   2.61282
##  stateut   statevt   stateva   statewa   statewv   statewi   statewy
##  2.36165   2.56100   2.23618   1.87424   2.63364   1.77545   3.30791
## year1983  year1984  year1985  year1986  year1987  year1988
## -0.07990  -0.07242  -0.12398  -0.03786  -0.05090  -0.05180
```

```
# via plm()
fatal_tefe_mod <- plm(fatal_rate ~ beertax,
                      data = Fatalities,
                      index = c("state", "year"),
                      model = "within",
                      effect = "twoways")

coeftest(fatal_tefe_mod, vcov = vcovHC(fatal_tefe_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
##         Estimate Std. Error t value Pr(>|t|)
## beertax -0.63998    0.35015 -1.8277  0.06865 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that since we exclude the intercept, `lm()` does estimate coefficients for $(n-1)+(T-1) = 47+6 = 53$ binary variables! Agina, `plm()` does only report the estimated coefficient on $BeerTax$.

The estimated regression function is

$$\widehat{FatalityRate} = \underset{(0.35)}{-0.64} \times BeerTax + StateEffects + TimeFixedEffects. \tag{11.9}$$

With an estimate of -0.66, the result is close to the estimated coefficient for the regression model including only entity fixed effects. Unsurprisingly, the coefficient is less precisely estimated as before but significantly different from zero at the level of 10%.

In view of (11.8) and (11.9) we conclude that the estimated relationship between traffic fatalities and the real beer tax is not affected by omitted variable bias due to factors that are constant either over time or across states.

11.5 The Fixed Effects Regression Assumptions and Standard Errors for Fixed Effects Regression

This section focusses on the entity fixed effects regression model and presents model assumptions that need to hold in order for OLS to produce unbiased estimates that are asymptotically normally distributed for large n. These assumptions are an extension of the assumptions made for the multiple regression model (see Key Concept 6.4) and are given in Key Concept 10.3. We will also briefly discuss standard errors in fixed effects regression models which differ from standard errors in multiple regression as the regression error can exhibit **serial correlation** in panel models.

Key Concept 10.3

The Fixed Effects Regression Assumptions

In the fixed effects regression model

$$Y_{it} = \beta_1 X_{it} + \alpha_i + u_{it} \quad , \quad i = 1, \dots, n, \ t = 1, \dots, T,$$

we assume that

1. the error term u_{it} has conditional mean zero, that is $E(u_{it}|u_{i1}, u_{i2}, \dots, u_{iT})$.

2. $(X_{i1}, X_{i2}, \dots, X_{i3}, u_{i1}, \dots, u_{iT})$, $i = 1, \dots, n$ are i.i.d. draws from their joint distribution.
3. Large outliers are unlikely i.e. (X_{it}, u_{it}) have nonzero finite fourth moments.
4. there is no perfect multicollinearity.

When there are multiple regressors, X_{it} is replaced by $X_{1,it}, X_{2,it}, \dots, X_{k,it}$.

The first assumption is that the error is uncorrelated with *all* observations on variable X for entity i over time. If this assumption is violated, we have omitted variables bias. The second assumption ensures that variables are i.i.d. across entities $i = 1, \dots, n$. Notice that this *does not* require observations to be uncorrelated *within* an entity. We say that the X_{it} are allowed to be **autocorrelated** or **serially correlated** within entities. This is a common property of time series data. The same is allowed for errors u_{it}. Consult chapter 10.5 for a detailed explanation why autocorrelation is a plausible characteristic in panel applications. The second assumption is granted if entities are selected by simple random sampling. The third and fourth assumptions are analagous to the multiple regression assumptions made in Key Concept 6.4.

Standard Errors for Fixed Effects Regression

Similar as for heteroskedasticity, autocorrelation invalidates usual of regression errors invalidates usual standard error fomulas and also the way heteroskedasticity-robust standard errors are computed since these formulas are derived under the assumption that there is no autocorrelation. In the context of heteroskedasticity *and* autocorrelation so-called **heteroskedasticity and autocorrelation-consistent (HAC) standard errors** need to be used. **Clustered standard errors** belong to these type of standard errors. They allow for heteroskedasticity *and* autocorrelated errors within an entity but *not for any kind of correlation across* entities.

As shown in the examples throughout this chapter, it is fairly easy to specify usage of clustered standard errors in regression summaries procuced by function like `coeftest()` in conjunction with `vcovHC()` from the `sandwich` package. Conveniently, `vcovHC()` recognizes panel model objects and computes clusterd standard errors by default.

The regressions conducted during this chapter are a good examples why usage of clustered standard errors is crucial in fixed effects models. For example, consider the entity and time fixed effects regression model for fatalities. Since `fatal_tefe_lm_mod` is an object of class `lm`, `coeftest()` does not compute clustered standard errros but uses robust standard errors that are only valid in the absence of autocorrelated errors.

```
# check class of the model (lm) object
class(fatal_tefe_lm_mod)
```

```
## [1] "lm"
```

```
# Summary based on heteroskedasticity-robust standard errors (no adjustment for autocorrelation)
coeftest(fatal_tefe_lm_mod, vcov = vcovHC(fatal_tefe_lm_mod, type = "HC1"))[1,]
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
## -0.6399800  0.2547149 -2.5125346  0.0125470
```

```
# check class of the (plm) model object
class(fatal_tefe_mod)
```

```
## [1] "plm"        "panelmodel"
```

```
# Summary based on clusterd standard errors (adjustment for autocorrelation + heteroskedasticity)
coeftest(fatal_tefe_mod, vcov = vcovHC(fatal_tefe_mod, type = "HC1"))
```

```
##
## t test of coefficients:
##
```

```
##         Estimate Std. Error t value Pr(>|t|)
## beertax -0.63998    0.35015 -1.8277  0.06865 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The outcomes deviate rather strongly: imposing no autocorrelation we obtain a standard error of 0.25, implying significance of β_1, the coefficient on $BeerTax$ at the level of 5%, whereas the clusterd standard error (0.35) leads to acceptance of the hypothesis $H_0 : \beta_1 = 0$ at the same level of significance (see (11.9)). Consult Appendix 10.2 of the book for insights on the computation of clustered standard errors.

11.6 Drunk Driving Laws and Traffic Deaths

There are two major sources of omitted variable bias that are not accounted for by all of our models of the relation between trafic fatalitites and beer taxes that we have considered so far: economic conditions and driving laws. Fortunately, `Fatalities` has recordings on state specific legal drinking age (`drinkage`), punishment (`jail`, `service`) and various economic indicators like unemployment rate (`unemp`) and per capita income (`income`). We may use this data to extend the preceeding analysis.

At first, we define variables according to the regression results presented in table 10.1 of the book.

```
# Discretize the minimum legal drinking age
Fatalities$drinkagec <- cut(Fatalities$drinkage,
                            breaks = 18:22,
                            include.lowest = TRUE,
                            right = FALSE
                            )

# Set minimum drinking age [21,22] to be the baseline level
Fatalities$drinkagec <- relevel(Fatalities$drinkagec, "[21,22]")

# Mandadory jail or community service?
Fatalities$punish <- with(Fatalities, factor(jail == "yes" | service == "yes",
                                             labels = c("no", "yes")
                                             )
                          )

# Observations on all variables for 1982 and 1988 only
Fatalities_1982_1988 <- Fatalities[with(Fatalities, year == 1982 | year == 1988), ]
```

Next, we estimate all seven models using `plm()`.

```
fatalities_mod1 <- lm(fatal_rate ~ beertax, data = Fatalities)

fatalities_mod2 <- plm(fatal_rate ~ beertax + state, data = Fatalities)

fatalities_mod3 <- plm(fatal_rate ~ beertax + state + year,
                       index = c("state","year"),
                       model = "within",
                       effect = "twoways",
                       data = Fatalities)

fatalities_mod4 <- plm(fatal_rate ~ beertax + state + year + drinkagec
                       + punish + miles + unemp + log(income),
                       index = c("state","year"),
                       model = "within",
```

```
                        effect = "twoways",
                        data = Fatalities)

fatalities_mod5 <- plm(fatal_rate ~ beertax + state + year + drinkagec
                        + punish + miles,
                        index = c("state","year"),
                        model = "within",
                        effect = "twoways",
                        data = Fatalities)

fatalities_mod6 <- plm(fatal_rate ~ beertax + year + drinkage
                        + punish + miles + unemp + log(income),
                        index = c("state","year"),
                        model = "within",
                        effect = "twoways",
                        data = Fatalities)

fatalities_mod7 <- plm(fatal_rate ~ beertax + state + year + drinkagec
                        + punish + miles + unemp + log(income),
                        index = c("state","year"),
                        model = "within",
                        effect = "twoways",
                        data = Fatalities_1982_1988)
```

Yet again we may use `stargazer()` to generate a comprehensive tabular presentation of the results.

```
library(stargazer)
library(sandwich)

rob_se <- list(
  sqrt(diag(vcovHC(fatalities_mod1, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod2, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod3, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod4, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod5, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod6, type="HC1"))),
  sqrt(diag(vcovHC(fatalities_mod7, type="HC1")))
)

stargazer(fatalities_mod1, fatalities_mod2, fatalities_mod3, fatalities_mod4, fatalities_mod5, fatalitie
          digits = 3,
          type = "latex",
          se = rob_se,
          model.numbers = FALSE,
          column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)", "(7)")
          )
```

Dependent variable:

fatal_rate

OLS

panel

linear

(1)

(2)

(3)

(4)

(5)

(6)

(7)

beertax

0.365***

-0.656**

-0.640*

-0.445

-0.690**

-0.456

-0.926***

(0.053)

(0.289)

(0.350)

(0.291)

(0.345)

(0.301)

(0.337)

drinkagec[18,19)

0.028

-0.010

0.037

(0.068)

(0.081)

(0.101)

drinkagec[19,20)

-0.018

-0.076

-0.065

(0.049)

(0.066)

(0.097)

drinkagec[20,21)
0.032
-0.100*
-0.113
(0.050)
(0.055)
(0.123)
drinkage
-0.002
(0.021)
punishyes
0.038
0.085
0.039
0.089
(0.101)
(0.109)
(0.101)
(0.161)
miles
0.00001
0.00002*
0.00001
0.0001***
(0.00001)
(0.00001)
(0.00001)
(0.00005)
unemp
-0.063***
-0.063***
-0.091***
(0.013)
(0.013)
(0.021)
log(income)

1.816***

1.786***

0.996

(0.624)

(0.631)

(0.666)

Constant

1.853***

(0.047)

Observations

336

336

336

335

335

335

95

R2

0.093

0.041

0.036

0.360

0.066

0.357

0.659

Adjusted R2

0.091

-0.120

-0.149

0.217

-0.134

0.219

0.157

Residual Std. Error

0.544 (df = 334)

F Statistic

34.394*** (df = 1; 334)

12.190*** (df = 1; 287)

10.513*** (df = 1; 281)

19.194*** (df = 8; 273)

3.252*** (df = 6; 275)

25.423*** (df = 6; 275)

9.194*** (df = 8; 38)

Note:

p<0.1; ***p<0.05;*** p<0.01

While coloumns (2) and (3) recap the results (11.8) and @red(eq:cbnfemod), coloumn (1) presents an estimate of the coefficient of interest in the naive OLS regression of the fatality rate on beer tax without any fixed effects. We obtain a *positive* estimate for the coefficient on beer tax that is likely to be heavily biased. Notice that the model fit is rather bad ($\overline{R^2} = 0.091$), too. The sign of the estimate changes as we extend the model by both entity and time fixed effects in models (2) and (3). Furthermore $\overline{R^2}$ increases substantially as fixed effects are included in the model equation. Nonetheless, as discussed before, the magnitudes of both estimates are by far too high.

Thus, model specifications (4) to (7) include covariates that shall capture the effect of overall state economic conditions as well as the legal framework. These covariates are

- `unemp`: a numeric variable stating the state specific unemployment rate.
- `log(income)`: the logarithm of real per capita income (in prices of 1988).
- `miles`: average miles per driver.
- `drinkage`: state specifiy minimum legal drinking age.
- `drinkagc`: a discretized version of `drinkage` that classifies states into four categories of minimal drinking age: 18, 19, 20, 21 and older. R denotes this as `[18,19)`, `[19,20)`, `[20,21)` and `[21,22]`. These categories are included as dummy regressors wherby `[21,22]` is chosen to be the reference category.
- `punish`: a dummy variable with levels `yes` and `no` that measures if drunk driving is severely punished by mandatory jail time or mandatory community service (first conviction).

Considering (4) as the base specification, we observe four interesting results:

1. Including the covariates does not lead to a major reduction of the estimated effect of the beer tax. Plus, the coefficient is not significantly different from zero at the level of 5% as the estimate is rather imprecise.

2. The minimum legal drinking age *does not* have an effect on traffic fatalities: none of the three dummy variables is significantly different from zero at any level of significance common in practice. Moreover, an F-Test of the joint hypothesis that all three coefficients are zero does not reject.

```
# Test if legal drinking age has no explanatory power
linearHypothesis(fatalities_mod4,
                 test = "F",
                 c("drinkagec[18,19)=0","drinkagec[19,20)=0","drinkagec[20,21)"),
                 vcov. = vcovHC(fatalities_mod4, type = "HC2"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## drinkagec[18,19) = 0
## drinkagec[19,20) = 0
## drinkagec[20,21) = 0
```

```
##
## Model 1: restricted model
## Model 2: fatal_rate ~ beertax + state + year + drinkagec + punish + miles +
##     unemp + log(income)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1    276
## 2    273  3 0.3514 0.7882
```

3. There is no evidence that punishment for first offenders has a deterring effects on drunk driving: the corresponding coefficient is not significant at the 10% level.

4. The economic variables have explanatory power in explaning traffic fatalities. We can check that the employment rate and per capita income are jointly significant at the level of 0.1%.

```
# Test if economic indicators have no explanatory power
linearHypothesis(fatalities_mod4,
                 test = "F",
                 c("log(income)", "unemp"),
                 vcov. = vcovHC(fatalities_mod4, type = "HC2"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(income) = 0
## unemp = 0
##
## Model 1: restricted model
## Model 2: fatal_rate ~ beertax + state + year + drinkagec + punish + miles +
##     unemp + log(income)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    275
## 2    273  2 30.482 1.125e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model (5) omitts economic factors. The regression result supports the presumption that economic indicators should remain part of the model as the coefficient on beer tax is sensitive to inclusion of the latter.

Results for model (6) are in favour of the presumption that legal drinking age has little explanatory power and that the coefficient of interest is not senstive to changes in the functional form of the relation between drinking age and traffic fatalities.

Estimation of spefcification (7) reveals that reducing the amount of available information inflates standard errors but does not lead to drastic changes in coefficient estimates.

Summary

We have not found evidence that menacing severe punishments and increasing minimum drinking ages are appropriate measures for reducing traffic fatalities due to drunk driving. Nonetheless we conclude that there is some effect of alcohol taxes on traffic fatalities which, however, is estimated imprecisely and cannot be

interpreted as the causal effect of interest as there seems to be omitted variable bias. This bias persists even though we used a panel approach that controls for entity specific and time fixed effects unobservables. A possible source for this bias are omitted variables that change both across entities and over time.

A powerful method that can be used if common panel regression approaches fail is instrumental variables regression. We will return to this issue in Chapter 12.

Chapter 12

Regression with a Binary Dependent Variable

In this chapter, we discuss a special case of regression models that aim to explain a limited dependent variable — a dependent variable with a limit range — by multiple regressors.

In particulat we consider models where the dependent variable is binary. We will see that in such models, the regression function can be interpreted as a conditional probability function.

We review the following concepts:

- The linear probability model
- Probit models
- Logit model
- Maximum likelihood estimation of nonlinear regression models

Of course, we will also see how to estimate abovementioned models using `R` and discuss an application where we explore the question whether there is racial discrimination in the US mortgage market.

12.1 Binary Dependent Variables and the Linear Probability Model

Key Concept 11.1

The Linear Probabilty Model

The linear regression model

$$Y_i = \beta_0 + \beta_1 + X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + u_i$$

with a binary depnendent variable Y_i is called the linear probability model as

$$E(Y|X_1, X_2, \ldots, X_k) = P(Y = 1|X_1, X_2, \ldots, X_3)$$

such that

$$P(Y = 1|X_1, X_2, \ldots, X_k) = \beta_0 + \beta_1 + X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki}.$$

Thus, the coefficient β_j can be interpreted as the change in the probability that $Y = 1$, holding constant the other $k - 1$ regressors. Just as in common multiple regression, the β_j can be estimated using OLS and the robust standard error formulas can be used for hypothesis testing and computation of confidence intervals.

In most linear probability models, R^2 has no meaningful interpretation since the regression line can never fit the data perfectly if the depdent variable is binary and the regressors are continuous. This can be seen in the application below.

We emphasize that it is *essential* to use robust standard errors since the u_i in a linear probability model are always heteroskedastic.

Following the book, we start by loading the dataset `HMDA` which provides data that relate to mortgage applications filed in Boston in the year of 1990.

```
# load AER package and attach data
library(AER)
data(HMDA)
```

We continue by inspecting the first few observations and compute summary statistics afterwards.

```
# Inspect data
head(HMDA)
```

```
##   deny pirat hirat     lvrat chist mhist phist unemp selfemp insurance
## 1   no 0.221 0.221 0.8000000     5     2    no   3.9      no        no
## 2   no 0.265 0.265 0.9218750     2     2    no   3.2      no        no
## 3   no 0.372 0.248 0.9203980     1     2    no   3.2      no        no
## 4   no 0.320 0.250 0.8604651     1     2    no   4.3      no        no
## 5   no 0.360 0.350 0.6000000     1     1    no   3.2      no        no
## 6   no 0.240 0.170 0.5105263     1     1    no   3.9      no        no
##   condomin afam single hschool
## 1       no   no     no     yes
## 2       no   no    yes     yes
## 3       no   no     no     yes
## 4       no   no     no     yes
## 5       no   no     no     yes
## 6       no   no     no     yes
```

```
summary(HMDA)
```

```
##   deny          pirat            hirat            lvrat        chist
##  no :2095   Min.   :0.0000   Min.   :0.0000   Min.   :0.0200   1:1353
##  yes: 285   1st Qu.:0.2800   1st Qu.:0.2140   1st Qu.:0.6527   2: 441
##             Median :0.3300   Median :0.2600   Median :0.7795   3: 126
##             Mean   :0.3308   Mean   :0.2553   Mean   :0.7378   4:  77
##             3rd Qu.:0.3700   3rd Qu.:0.2988   3rd Qu.:0.8685   5: 182
##             Max.   :3.0000   Max.   :3.0000   Max.   :1.9500   6: 201
##  mhist    phist          unemp        selfemp    insurance  condomin
##  1: 747   no :2205   Min.   : 1.800   no :2103   no :2332   no :1694
##  2:1571   yes: 175   1st Qu.: 3.100   yes: 277   yes:  48   yes: 686
##  3:  41              Median : 3.200
##  4:  21              Mean   : 3.774
##                      3rd Qu.: 3.900
##                      Max.   :10.600
##   afam      single     hschool
##  no :2041   no :1444   no :  39
##  yes: 339   yes: 936   yes:2341
##
##
##
##
```

The variable we are interesting in modelling is `deny`, an indicator for whether an applicants mortgage application has been accepted (`deny = no`) or denied (`deny = yes`). A regressors that can be presumed to have power in explaining whether a mortgage application has been denied or accepted is `pirat`, the size of the anticipated total monthly loan payments relative to the the applicant's income. It is straightforward to translate this into the simple regression approach

$$deny = \beta_0 + \beta_1 P/I \ ratio + u. \tag{12.1}$$

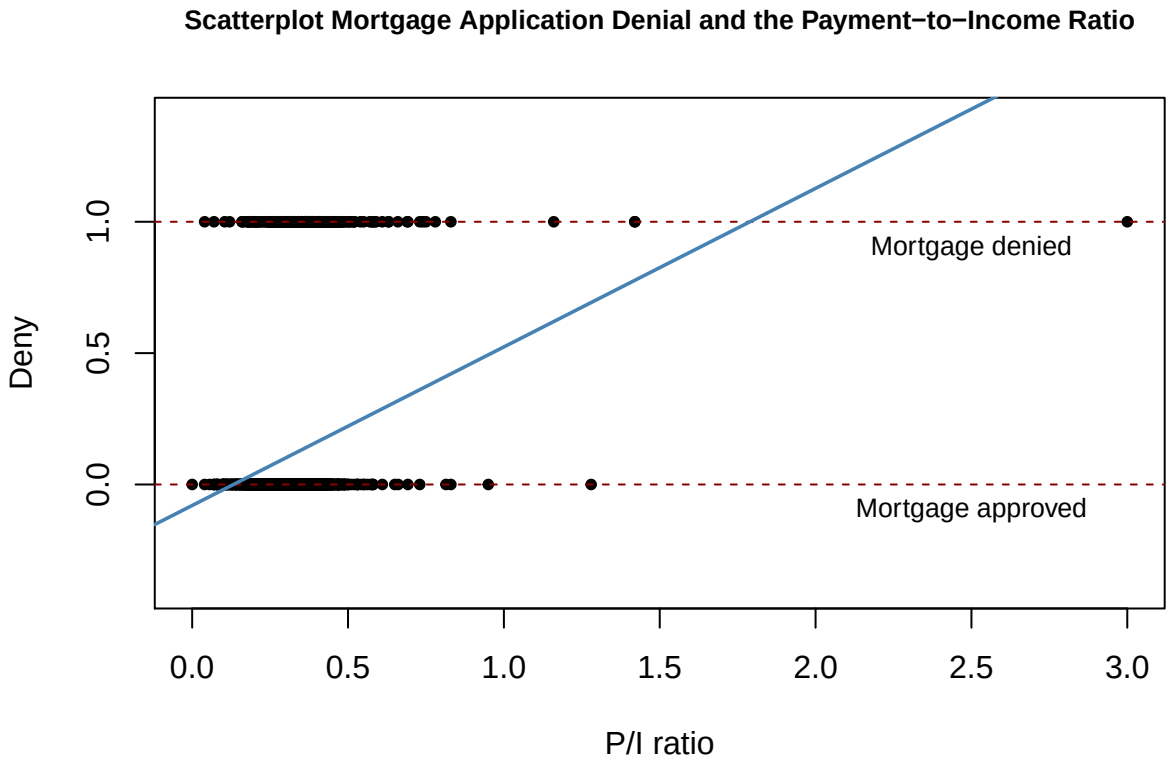
We can estimate this model just as any other linear regression model using `lm()`. Before we do, `deny` must be converted to a numeric variable using `as.numeric()`. Note that `as.numeric(HMDA$deny)` will turn `deny = no` into `deny = 1` and `deny = yes` into `deny = 2`, so using `as.numeric(HMDA$deny)-1` we obtain levels `0` and `1`.

```
# convert 'deny' to numeric
HMDA$deny <- as.numeric(HMDA$deny) - 1

# estimate a simple linear probabilty model
denymod1 <- lm(deny ~ pirat, data = HMDA)
denymod1
```

```
##
## Call:
## lm(formula = deny ~ pirat, data = HMDA)
##
## Coefficients:
## (Intercept)        pirat
##    -0.07991      0.60353
```

We plot the data and the regression line to reproduce Figure 11.1 of the book.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
     main = "Scatterplot Mortgage Application Denial and the Payment-to-Income Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4,1.4),
     cex.main = 0.8
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line
abline(denymod1,
       lwd=1.8,
       col = "steelblue")
```



We observe that, according to the estimated model equation, a $P/Iratio$ of 1 is associated with an expected probability of mortgage application denial of roughly 50%. The model indicates that there is a postive relation between P/I *ratio* and the probabilty of a denied mortgage application i.e. individuals with a high ratio of loan payments to income are more likely to be rejected.

We may use `coeftest` to obtain robust standard errors for both coefficient estimators.

```
# model coefficient summary
coeftest(denymod1, vcov. = vcovHC(denymod1, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.079910   0.031967 -2.4998   0.01249 *
## pirat        0.603535   0.098483  6.1283 1.036e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression line is

$$\widehat{deny} = \underset{(0.032)}{-0.080} + \underset{(0.098)}{0.604} P/I \ ratio.$$

The true coefficient on P/I *ratio* is statistically different from 0 at the 1% level. Its estimate can be interpreted as follows: a 1% increase in P/I *ratio* leads to an increase in the probability of a loan denial by $0.604 * 0.01 = 0.00604 \approx 0.6\%$.

Following the book we augment the approach (12.1) by an additional regressor *black* which equals 1 if the applicant is an African American and equals 0 if the applicant is white. Such a specification is the baseline for investigation of the question whether there is racial discrimination in the mortgage market: if being black has a significant (positive) influence on the probability of a loan denial when we control for factors that allow for objective assessment of an applicants credit worthyness, this is an indicator for discrimination.

```
# rename variable 'afam'
colnames(HMDA)[colnames(HMDA) == "afam"] <- "black"

# estimate the model
denymod2 <- lm(deny ~ pirat + black, data = HMDA)
coeftest(denymod2, vcov. = vcovHC(denymod2))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.090514   0.033430 -2.7076  0.006826 **
## pirat        0.559195   0.103671  5.3939 7.575e-08 ***
## blackyes     0.177428   0.025055  7.0815 1.871e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So the estimated regression function is

$$\widehat{deny} = \underset{(0.029)}{-0.091} + \underset{(0.089)}{0.559} P/I\ ratio + \underset{(0.025)}{0.177} afam. \tag{12.2}$$

The coefficient on *black* is positive and significantly different from zero at the 0.01% level. The interpretation is that, holding constant the $P/I\ ratio$, being black increases the probability of a mortgage application denial by about 17.7%. This suggests racial discrimination. However, the estimation might suffer from omitted variable bias so this could be a premature conclusion.

12.2 Probit and Logit Regression

The linear probability model has a major flaw: it assumes the conditional probability function to be linear. This does not allow the probability of observing $Y = 1$ conditional on some regressor to lie between 0 and 1. We can easily see this in our reproduction of Figure 11.1 of the book: for $P/I\ ratio \geq 1.75$, model (12.1) predicts the probability of a mortgage application denial to be bigger than 1. For applications with $P/I\ ratio$ close to 0, the predicted probability of denial is even negative so the model has no meaningful interpretation here.

This circumstance demands for approaches that use a *nonlinear* function to model the conditional probability function of a binary dependent variable: probit and logit regression models.

Probit Regression

In probit regression, the cumulative standard normal distribution function Φ is used to model the regression function when the dependent variable is binary:

$$E(Y|X) = P(Y = 1|X) = \Phi(\beta_0 + \beta_1 X) \tag{12.3}$$

$\beta_0 + \beta_1 X$ in (12.3) plays the role of a quantile z. Remember that

$$\Phi(z) = P(Z \leq z)\ ,\ Z \sim \mathcal{N}(0, 1)$$

such that the probit coefficient β_1 in (12.3) is the change in z associated with a one unit change in X. Note that, although the effect on z of a change in X is linear, the link between z and the dependent variable Y is nonlinear since Φ is a nonlinear function.

Since in a probit models the dependent variable is a nonlinear function of the regressors, the coefficient on X has no easy interpretation. According to Key Concept 8.1, the expected change in the probability that $Y = 1$ due to a change in P/I *ratio* can be computed in the following manner:

1. Compute the predicted probability that $Y = 1$ for the original value of X.
2. Compute the predicted probability that $Y = 1$ for $X + \Delta X$.
3. Compute the difference between both predicted probabilities.

Of course we can generalize (12.3) to a probit regression with multiple regressors to mitigate the risk of facing omitted variable bias. The core knowledge on probit regression is summarized in Key Concept 11.2

Key Concept 11.2

Probit Model, Predicted Probabilities and Estimated Effects

$$P(Y = 1|X_1, X_2, \dots, X_k) = \Phi(\beta_0 + \beta_1 + X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)$$

is the population probit model with multiple regressors where Y is a binary varible, $X_1, X_2, \dots, X_k$ are regressors and Φ is the cumulative standard normal distribution function.

The predicted probability that $Y = 1$ given $X_1, X_2, \dots, X_k$ can be calculated in two steps:

1. Compute $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$
2. Look up $\Phi(z)$ using a normal distribution table or by calling `pnorm()`.

β_j is the effect on z of a one unit change in regressor X_j, holding constant all other $k - 1$ regressors.

The effect on the predicted probability of a change in a regressor can be computed as shown in Key Concept 8.1.

In R, probit models can be estimated using the function `glm()` from package `stats`. Using the argument `family` we specify that we want to use a probit link function.

```
# estimate the simple probit model
denyprobit <- glm(deny ~ pirat,
                  family = binomial(link = "probit"),
                  data = HMDA)

coeftest(denyprobit, vcov. = vcovHC(denyprobit, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -2.19415    0.18901 -11.6087 < 2.2e-16 ***
## pirat        2.96787    0.53698   5.5269 3.259e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

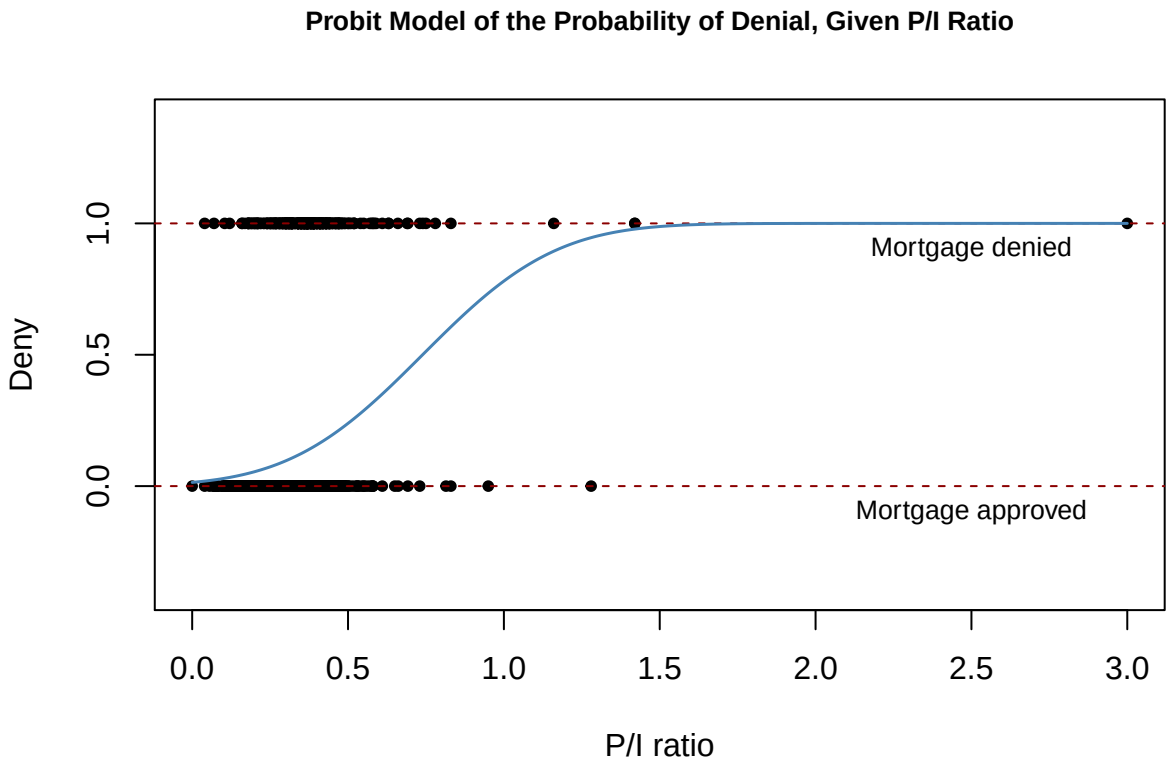
The estimated model equation is

$$\widehat{P(deny|P/I \ ratio)} = \Phi(\underset{(0.19)}{-2.19} + \underset{(0.54)}{2.97} P/I \ ratio). \tag{12.4}$$

Just as in the linear probability model we find that the relation between the probability of denial and the payments-to-income ratio is positive and that corresponding coefficient is highly significant.

The following code chunk reproduces Figure 11.2 of the book.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
     main = "Probit Model of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4,1.4),
     cex.main = 0.85
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line
x <- seq(0, 3, 0.01)
y <- predict(denyprobit, list(pirat = x), type="response")

lines(x, y, lwd = 1.5, col = "steelblue")
```



Notice that the estimated regression function has a stretched "S" shape which typical for the c.d.f. of a of continuous random variable with symmetric p.d.f. like a normal distributed random variable. The function is clearly nonlinear and flattens out for large and small values of P/I *ratio*. More important, it ensures that

predicted conditional probabilities of a denial lie between 0 and 1.

We may use `predict()` to compute the predicted change in the denial probability when $P/I\ ratio$ is increased from 0.3 to 0.4.

```
# 1. compute predictions for P/I ratio = 0.3,0.4
predictions <- predict(denyprobit,
                       newdata = data.frame("pirat" = c(0.3, 0.4)),
                       type = "response"
                       )

# 2. Compute difference in probabilities
diff(predictions)
```

```
##          2
## 0.06081433
```

We find that an increase in the payment-to-income ratio from 0.3 to 0.4 is predicted to increase the probability of denial by approximately 6.2 percentage points.

We continue by using a probit model to estimate the effect of race on the probability of a mortgage application denial.

```
denyprobit2 <- glm(deny ~ pirat + black,
                   family = binomial(link = "probit"),
                   data = HMDA)

coeftest(denyprobit2, vcov. = vcovHC(denyprobit2, type = "HC1"))
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -2.258787   0.176608 -12.7898 < 2.2e-16 ***
## pirat        2.741779   0.497673   5.5092 3.605e-08 ***
## blackyes     0.708155   0.083091   8.5227 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model equation is

$$\widehat{P(deny|P/I\ ratio, black)} = \Phi(\underset{(0.18)}{-2.26} + \underset{(0.50)}{2.74}\, P/I\ ratio + \underset{(0.08)}{0.71}\, black). \tag{12.5}$$

While all coefficients are highly significant, both the estimated coefficients on the payments-to-income ratio and the indicator for African American descent are positive. Again, the coefficients are difficult to interpret but they indicate that first, being black applicants have a higher higher probability of denial than white applicants, holding constant the payments-to-income ratio and second, applicants with a high payments-to-income ratio face a higher risk of being rejected.

How big is the predicted difference in denial probility between two hypothetical applicants with the same payments-to-income ratio? As before, we may use `predict()` to compute this difference.

```
# 1. compute predictions for P/I ratio = 0.3,0.4
predictions <- predict(denyprobit2,
                       newdata = data.frame("black" = c("no", "yes"), "pirat" = c(0.3,0.3)),
                       type = "response"
```

```
                        )

# 2. Compute difference in probabilities
diff(predictions)
```

```
##         2
## 0.1578117
```

In this case, the difference in denial probabilities is 15.8%.

Logit Regression

Key Concept 11.3

Logit Regression

The logit regression population function is

$$\begin{aligned} P(Y = 1|X_1, X_2, \dots, X_k) &= F(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k) \\ &= \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}}. \end{aligned}$$

The idea is similar do probit regression except that a different c.d.f. is used:

$$F(x) = \frac{1}{1 + e^{-x}}$$

is the c.d.f. of a standard logistically distributed random variable.

Key Concept 11.3 summarizes the main differences between logit and probit regression. As for probit regression, there is no simple interpretation of coefficients and it is best to consider predicted probabilitites or differences in predicted probabilities. Here again, t-statitics and confidence intervals based on large sample normal approximation can be computed as usual.

It is fairly easy to estimate a logit regression model using R.

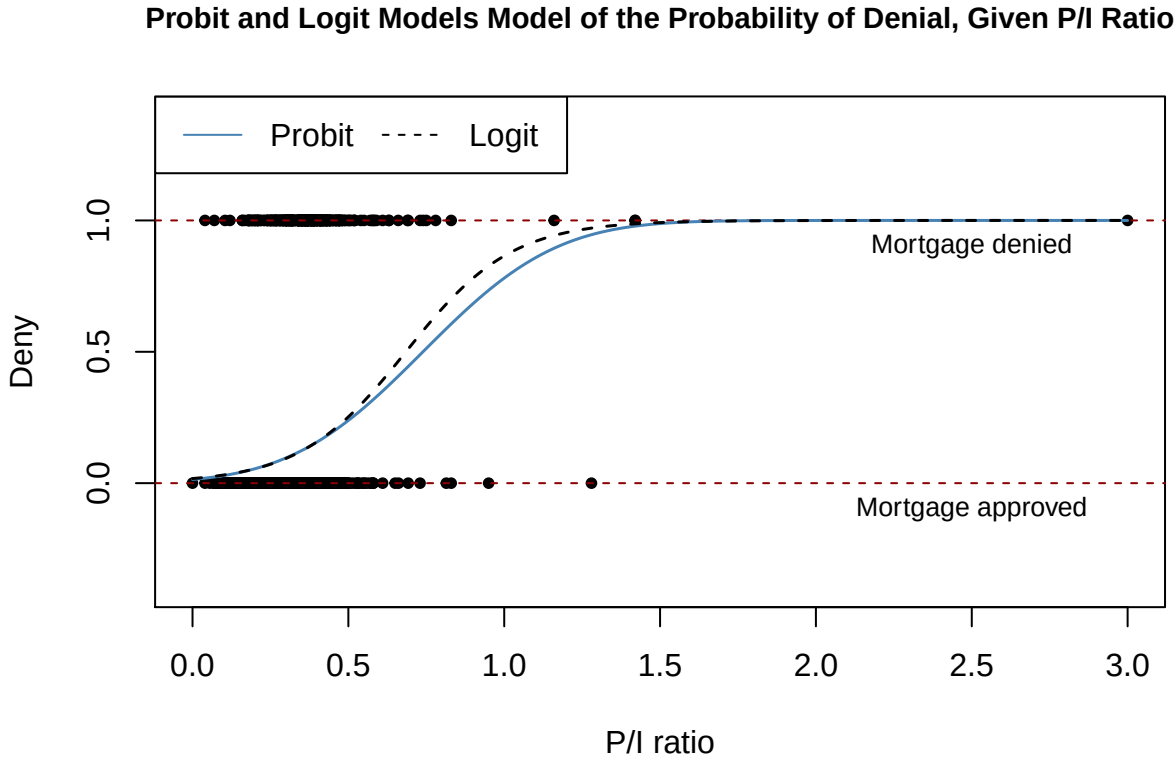
```
denylogit <- glm(deny ~ pirat,
                 family = binomial(link = "logit"),
                 data = HMDA)

coeftest(denylogit, vcov. = vcovHC(denylogit, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -4.02843    0.35898 -11.2218 < 2.2e-16 ***
## pirat        5.88450    1.00015   5.8836 4.014e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The subsequent code chunk reproduces figure 11.3 of the book.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
```

```
     main = "Probit and Logit Models Model of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.9
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line of probit and logit models
x <- seq(0, 3, 0.01)
y_probit <- predict(denyprobit, list(pirat = x), type="response")
y_logit <- predict(denylogit, list(pirat = x), type="response")

lines(x, y_probit, lwd = 1.5, col = "steelblue")
lines(x, y_logit, lwd = 1.5, col = "black", lty = 2)

# add a legend
legend("topleft",
       horiz = TRUE,
       legend = c("Probit", "Logit"),
       col = c("steelblue", "black"),
       lty = c(1, 2))
```



Notice that both models produce very similar estimates of the probability that a mortgage application will be denied depending on the applicants payment-to-income ratio.

Following the book we expand the simple logit model of mortage denial with the additional regressor *black* and estimate the model.

```
denylogit2 <- glm(deny ~ pirat + black,
                  family = binomial(link = "logit"),
                  data = HMDA)

coeftest(denylogit2, vcov. = vcovHC(denylogit2, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -4.12556    0.34597 -11.9245 < 2.2e-16 ***
## pirat        5.37036    0.96376   5.5723 2.514e-08 ***
## blackyes     1.27278    0.14616   8.7081 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain

$$\widehat{P(deny = 1|P/I ratio, black)} = F(\underset{(0.35)}{-4.13} + \underset{(0.96)}{5.37}\, P/I\ ratio + \underset{(0.15)}{1.27}\, black). \tag{12.6}$$

As for the probit model (12.5) all model coefficients are highly significant and we obtain postive estimate for the coefficients on P/I *ratio* and *black*. For comparison purposes we compute the predicted probability of denial for two hypothetical applicants that differ in race and have a P/I *ratio* of 0.3.

```
# 1. compute predictions for P/I ratio = 0.3
predictions <- predict(denylogit2,
                       newdata = data.frame("black" = c("no", "yes"), "pirat" = c(0.3,0.3)),
                       type = "response"
                      )

# 2. Compute difference in probabilities
diff(predictions)
```

```
##         2
## 0.1492945
```

We find that the white applicant faces a denial probability of only 7.5% percent while the African American is rejected with a probability of 22.4% with makes a difference of 14.9%.

Comparison of Models

The linear probability model, the probit model and the logit model deliver only approximations to the unknown population regression function $E(Y|X)$. It is not unabigious to decide which model one should use in practice. The linear probability model has the clear drawbacks of not beeing able to capture the nonlinear nature of the population regression function and predicting probabilities outside the interval of $[0, 1]$ for extreme regressor values. Probit and logit models are hard to interpret but can capture nonlinearities. Both models produce predictions of probabilities that lie inside $[0, 1]$. Predictions of probit and logit models are often close to each other. The book suggests to use the method that is easiest to use in the statistical

software of choice. As we have seen, it is equally easy to estimate probit and logit model using R. We can therefore give no general guide which method to use.

12.3 Estimation and Inference in the Logit and Probit Models

So far nothing has been said about *how* logit and probit models are estimated by statistical software. The reason why this is interesting is that both models are *nonlinear in the parameters* and thus cannot be estimated using OLS. Instead one uses a concept called *Maximum Liklihood Estimation*. Another approach is nonlinear least squares (NLS) estimation.

Nonlinear Least Squares

Consider the multiple regression probit model

$$E(Y|X_1,\dots,X_k) = P(Y=1|X_1,\dots,X_k) = \Phi(\beta_0+\beta_1X_1+\cdots+\beta_kX_k). \tag{12.7}$$

Similarly to OLS, NLS estimates the parameters $\beta_0,\beta_1,\dots,\beta_k$ by minimizing the sum of squared mistakes

$$\sum_{i=1}^{n}\left[Y_i-\Phi(b_0+b_1X_{1i}+\cdots+b_kX_{ki})\right]^2.$$

NLS estimation is a consistent approach that produces estimates which are normally distributed in large samples. In R there are functions like `nls()` from package `stats` that provide algorithms for solving nonlinear least squares problems. However, NLS is inefficient, meaning that there are estimation techniques that have a smaller variance which is why we will not dwell any further on this topic.

Maximum Likelihood Estimation

In maximum likelihood estimation (MLE) we seek to estimate unknown parameters choosing them such that the likelihood of drawing the sample observed is maximized. This probability is given by the likelihood function, the joint probability distribution of the data treated as a function of the unknown parameters. Put differently maximum likelihood of unkown parameters are values that are the most likely one to have produced the data observed. MLE is more efficient than NLS.

As the MLE is normally distributed in large samples, statistical inference for coefficients in nonlinear models like logit and probit regression can be made using the same tools that are used for linear regression models: we can compute t-statistics and confidence intervals.

Many software packages use an MLE algorithm for estimation of nonlinear models. The function `glm()` uses an algorithm named *Iteratively Reweighted Least Squares*.

Measures of Fit

It is important to be aware that the usual R^2 and $\overline{R^2}$ are **invalid** for nonlinear regression models. The reason for this is simple: both measure are derived under the assumption that the relation between the dependent and the explanatory variable(s) is linear. This obviously does not hold for probit and logit models such that R^2 does not fall between 0 and 1 and there is no meaningful interpretation. However, statistical softare often reports these measures anyhow.

For example, consider the summary of the model object `denyprobit` produced by `summary()` for (12.4), the probit regression of *deny* on *P/I rat*.

```
summary(denyprobit)
```

```
##
## Call:
## glm(formula = deny ~ pirat, family = binomial(link = "probit"),
##     data = HMDA)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4140  -0.5281  -0.4750  -0.3900   2.8159
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.1941     0.1378 -15.927  < 2e-16 ***
## pirat         2.9679     0.3858   7.694 1.43e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1744.2  on 2379  degrees of freedom
## Residual deviance: 1663.6  on 2378  degrees of freedom
## AIC: 1667.6
##
## Number of Fisher Scoring iterations: 6
```

Besides reporting wrong standard errors (remember that it is inevitable to use heteroskedasticity robust standard errors for probit and logit models), the output also reports R^2 and $\overline{R^2}$ although neither of these are valid for this model.

The take away message is that one should be aware which measures are reported by functions like `summary()` and whether these measures are adequate for the model at hand.

There are many measures of fit for nonlinear regression models and there is no consensus which one should be reported. The situation is even more complicated because not there is no measure of fit that generally applicable. For models with a binary response variable like *deny* one could use the following rule:
If $Y_i = 1$ and $P(Y_i|\widehat{X_{i1}, \dots, X_{ik}}) > 0.5$ or if $Y_i = 0$ and $P(Y_i|\widehat{X_{i1}, \dots, X_{ik}}) < 0.5$, consider the Y_i as correctly predicted. Otherwise Y_i is said to be incorrectly predicted. The measure of fit is the share of correctly predicted observations. The downside of such an approach is that it does not yield information on the quality of the prediction.

An alternative are so called pseudo-R^2 measures. In order to measure quality of fit, these measures compare the value of the maximized (log-)likelihood of the model with all regressors (the *full model*) to the likelihood of a model with no regressors (*null model*, regression on a constant).

Consider a probit regression. The pseudo-R^2 is given by

$$\text{pseudo-}R^2 = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}$$

where $f_j^{max} \in [0, 1]$ denotes the maximized likelihood for model j.

The reasoning behind this is that the maximized likelihood increases as additional regressors are added to the model, similarly to thr decrease in SSR when regressors are added in a linear regression model. If the full model has a similar maximized likelihood than the null model, the full model does not really improve upon on a model that uses only the information in the dependent variable and pseudo-$R^2 \approx 0$. If the full model fits

the data very good, the maximized likelihood shoud be close 1 such that $\ln(f_{full}^{max}) \approx 0$ and pseudo-$R^2 \approx 1$. See Appendix 11.2 of the book for more on MLE and pseudo-R^2 measures.

`summary()` does not report pseudo-R^2 for models estimated by `glm()` but we can use the entries *residual deviance* (`deviance`) and *null deviance* (`null.deviance`). These are computed as

$$\text{deviance} = -2 \times \left[\ln(f_{saturated}^{max}) - \ln(f_{full}^{max})\right]$$

and

$$\text{null deviance} = -2 \times \left[\ln(f_{saturated}^{max}) - \ln(f_{null}^{max})\right]$$

where $f_{saturated}^{max}$ is the maximized likelihood for a model that assumes each observations has its own parameter ($n+1$ parameters to be estimated which leads to a perfect fit). For models with binrary dependent variables, it holds that

$$\text{pseudo-}R^2 = 1 - \frac{\text{deviance}}{\text{null deviance}} = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}$$

```
pseudoR2 <- 1 - (denyprobit2$deviance) / (denyprobit2$null.deviance)
pseudoR2
```

```
## [1] 0.08594259
```

Another way to obtain the pseudo-R^2 is to estimate the null model using `glm()` and extract maximized log-likelihoods for both the null and the full model using the function `logLik()`

```
# compute null model
denyprobit_null <- glm(formula = deny ~ 1, family = binomial(link = "probit"), data = HMDA)

1 - logLik(denyprobit2)[1]/logLik(denyprobit_null)[1]
```

```
## [1] 0.08594259
```

12.4 Application to the Boston HMDA Data

Models (12.5) and (12.6) indicate that denial rates are higher for African American applicants holding constant payment-to-income ratio. Both results could be wrong due to omitted variable bias. In order to obtain a more trustworthy estimate of the effect of being black on the probability of a mortgage application denial we estimate a linear probability model as well as several logit and probit models that control for financial variables and additional applicant characteristics which are likely to determine the probability of denial and differ between black and white applicants.

Sample averages as shown in table 11.1 of the book can be easily reproduced using the functions `mean()` (as usual for numeric variables) and `prop.table()` (for factor variables).

```
# Mean P/I ratio
mean(HMDA$pirat)
```

```
## [1] 0.3308136
```

```
# inhouse expense-to-total-income ratio
mean(HMDA$hirat)
```

```
## [1] 0.2553461
```

```
# loan-to-value ratio
mean(HMDA$lvrat)
```

```
## [1] 0.7377759
```

```
# consumer credit score
mean(as.numeric(HMDA$chist))
```

```
## [1] 2.116387
```

```
# mortgage credit score
mean(as.numeric(HMDA$mhist))
```

```
## [1] 1.721008
```

```
# public bad credit record
mean(as.numeric(HMDA$phist)-1)
```

```
## [1] 0.07352941
```

```
# denied mortgage insurance
prop.table(table(HMDA$insurance))
```

```
##
##         no        yes
## 0.97983193 0.02016807
```

```
# self-employed
prop.table(table(HMDA$selfemp))
```

```
##
##        no       yes
## 0.8836134 0.1163866
```

```
# single
prop.table(table(HMDA$single))
```

```
##
##        no       yes
## 0.6067227 0.3932773
```

```
# high school dimploma
prop.table(table(HMDA$hschool))
```

```
##
##         no        yes
## 0.01638655 0.98361345
```

```
# unemployment rate
mean(HMDA$unemp)
```

```
## [1] 3.774496
```

```
# condominium
prop.table(table(HMDA$condomin))
```

```
##
##        no       yes
## 0.7117647 0.2882353
```

```
# black
prop.table(table(HMDA$black))
```

```
##
##       no      yes
```

```
## 0.857563 0.142437
```

```
# deny
prop.table(table(HMDA$deny))
```

```
##
##         0         1
## 0.8802521 0.1197479
```

See Chapter 11.4 of the book or use the R's help function for more info on variables contained in the `HMDA` dataset.

We transform loan-to-value ratio (`lvrat`) into a factor variable where

$$lvrat = \begin{cases} \text{low} & \text{if } lvrat < 0.8, \\ \text{medium} & \text{if } 0.8 \leq lvrat \leq 0.95, \\ \text{high} & \text{if } lvrat > 0.95. \end{cases}$$

```
# define low, medium and high loan-to-value ratio
HMDA$lvrat <- factor(
  ifelse(HMDA$lvrat < 0.8, "low",
  ifelse(HMDA$lvrat >= 0.8 & HMDA$lvrat <= 0.95, "medium", "high")),
  levels = c("low", "medium", "high")
  )

# convert credit scores to numeric
HMDA$mhist <- as.numeric(HMDA$mhist)
HMDA$chist <- as.numeric(HMDA$chist)
```

In the next step we reproduce the estimation results presented in table 11.2 of the book.

```
# estimate all 6 models for the denial probability
lpm_HMDA <- lm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
               + insurance + selfemp, data = HMDA)

logit_HMDA <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                  + insurance + selfemp,
                  family = binomial(link = "logit"), data = HMDA)

probit_HMDA_1 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                     + insurance + selfemp,
                     family = binomial(link = "probit"), data = HMDA)

probit_HMDA_2 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                     + insurance + selfemp + single + hschool + unemp,
                     family = binomial(link = "probit"), data = HMDA)

probit_HMDA_3 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist
                     + phist + insurance + selfemp + single + hschool + unemp + condomin
                     + I(mhist==3) + I(mhist==4) + I(chist==3) + I(chist==4) + I(chist==5)
                     + I(chist==6),
                     family = binomial(link = "probit"), data = HMDA)

probit_HMDA_4 <- glm(deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist
```

```
                    + insurance + selfemp + single + hschool + unemp,
                    family = binomial(link = "probit"), data = HMDA)
```

Just as in previous chapters, we use the package `sandwich` for computation of heteroskedasticity-robust standard errors of the coefficient estimators in all models and store these in a a list which is then supplied as the argument `se` in `stargazer()`.

```
library(stargazer)
library(sandwich)

rob_se <- list(
  sqrt(diag(vcovHC(lpm_HMDA, type = "HC1"))),
  sqrt(diag(vcovHC(logit_HMDA, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_1, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_2, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_3, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_4, type = "HC1")))
)

stargazer(lpm_HMDA, logit_HMDA, probit_HMDA_1, probit_HMDA_2, probit_HMDA_3, probit_HMDA_4,
          digits = 3,
          type = "latex",
          se = rob_se,
          model.numbers = FALSE,
          column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)")
          )
```

Dependent variable:

deny

OLS

logistic

probit

(1)

(2)

(3)

(4)

(5)

(6)

blackyes

0.084***

0.688***

0.389***

0.371***

0.363***

0.246

(0.023)

(0.183)
(0.099)
(0.100)
(0.101)
(0.479)
pirat
0.449***
4.764***
2.442***
2.464***
2.622***
2.572***
(0.114)
(1.332)
(0.673)
(0.654)
(0.665)
(0.728)
hirat
-0.048
-0.109
-0.185
-0.302
-0.502
-0.538
(0.110)
(1.298)
(0.689)
(0.689)
(0.715)
(0.755)
lvratmedium
0.031**
0.464***
0.214***
0.216***

0.215**

0.216***

(0.013)

(0.160)

(0.082)

(0.082)

(0.084)

(0.083)

lvrathigh

0.189***

1.495***

0.791***

0.795***

0.836***

0.788***

(0.050)

(0.325)

(0.183)

(0.184)

(0.185)

(0.185)

chist

0.031***

0.290***

0.155***

0.158***

0.344***

0.158***

(0.005)

(0.039)

(0.021)

(0.021)

(0.108)

(0.021)

mhist

0.021*

0.279**

0.148**

0.110

0.162

0.111

(0.011)

(0.138)

(0.073)

(0.076)

(0.104)

(0.077)

phistyes

0.197***

1.226***

0.697***

0.702***

0.717***

0.705***

(0.035)

(0.203)

(0.114)

(0.115)

(0.116)

(0.115)

insuranceyes

0.702***

4.548***

2.557***

2.585***

2.589***

2.590***

(0.045)

(0.576)

(0.305)

(0.299)

(0.306)

(0.299)
selfempyes
0.060***
0.666***
0.359***
0.346***
0.342***
0.348***
(0.021)
(0.214)
(0.113)
(0.116)
(0.116)
(0.116)
singleyes
0.229***
0.230***
0.226***
(0.080)
(0.086)
(0.081)
hschoolyes
-0.613***
-0.604**
-0.620***
(0.229)
(0.237)
(0.229)
unemp
0.030*
0.028
0.030
(0.018)
(0.018)
(0.018)
condominyes

-0.055

(0.096)

I(mhist == 3)

-0.107

(0.301)

I(mhist == 4)

-0.383

(0.427)

I(chist == 3)

-0.226

(0.248)

I(chist == 4)

-0.251

(0.338)

I(chist == 5)

-0.789*

(0.412)

I(chist == 6)

-0.905*

(0.515)

blackyes:pirat

-0.579

(1.550)

blackyes:hirat

1.232

(1.709)

Constant

-0.183***

-5.707***

-3.041***

-2.575***

-2.896***

-2.543***

(0.028)

(0.484)

(0.250)

(0.350)

(0.404)

(0.370)

Observations

2,380

2,380

2,380

2,380

2,380

2,380

R2

0.266

Adjusted R2

0.263

Log Likelihood

-635.637

-636.847

-628.614

-625.064

-628.332

Akaike Inf. Crit.

1,293.273

1,295.694

1,285.227

1,292.129

1,288.664

Residual Std. Error

0.279 (df = 2369)

F Statistic

85.974*** (df = 10; 2369)

Note:

p<0.1; ***p<0.05;*** p<0.01

Models (1), (2) and (3) are base specifications that include several financial control variables. They differ only in the way they model the denial probability. (1) is linear probability model, (2) is a logistic regression and (3) uses the probit approach.

Since model (1) is a linear model, the coefficients have direct interpretation. For example, an increase in the consumer credit score by 1 unit is estimated to increase the probability of a loan denial by about 0.031

percentage points. We also find that having a high loan-to-value ratio is predicted to be not conductive for credit approval: the coefficient for a loan-to-value ratio higher than 0.95 is 0.189 so clients with this property ceteris paribus are estimated to face an almost 19% larger risk of denial than those with a low loan-to-value ratio. For the linear probability model, the coefficient on black is estimated to be 0.084 which indicates the denial probability for African Americans is 8.4% larger than for white applicants with the same characteristics except for race. Apart from housing expense-to-income ratio and the mortgage credit score, all coefficients are significant.

Models (2) and (3) provide similar evidence that there is racial discrimination in the US mortgage market. All coefficients except for housing expense-to-income ratio (which is not significantly different from zero) are signiicant at 1% level. As discussed above, the nonlinearity makes interpretation of coefficient estimates more difficult than for model (1). In order to make a statement about the effect of beeing black, we need to compute the estimated denial probability for two individuals that differ only in race. For the comparison we consider two individuals that share mean values for all numeric regressors. For all qualitative variables we assign the property that is the most representative for the data at hand. As an example, consider self-employment: we have seen that about 88% of all individuals in the sample are not self-employed such that we set `selfemp = no`. Using this approach, the estimate for the effect on the denial probability of being African American obtained when using the logit model (2) is about 4%. The next code chunk shows how to apply this approach for models (1) to (7) using `R`.

```
# regressor values for average person, black = "yes"
new <- data.frame(
  "pirat" = mean(HMDA$pirat),
  "hirat" = mean(HMDA$hirat),
  "lvrat" = "low",
  "chist" = mean(HMDA$chist),
  "mhist" = mean(HMDA$mhist),
  "phist" = "no",
  "insurance" = "no",
  "selfemp" = "no",
  "black" = c("no","yes"),
  "single" = "no",
  "hschool" = "yes",
  "unemp" = mean(HMDA$unemp),
  "condomin" = "no"
)

# differnce predicted by the LPM
predictions <- predict(lpm_HMDA, newdata = new)
diff(predictions)
```

```
##          2
## 0.08369674
```

```
# differnce predicted by the logit model
predictions <- predict(logit_HMDA, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.04042135
```

```
# difference predicted by probit model (3)
predictions <- predict(probit_HMDA_1, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.05049716
```

```
# difference predicted by probit model (4)
predictions <- predict(probit_HMDA_2, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.03978918
```

```
# difference predicted by probit model (5)
predictions <- predict(probit_HMDA_3, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.04972468
```

```
# difference predicted by probit model (6)
predictions <- predict(probit_HMDA_4, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.03955893
```

The estimates of the impact on the denial probability of being black are similar for models (2) and (3). In particular, it is interesting that the magnitude of the estimated effects is much smaller than for probit and logit models that do not control for financial characteristics (see section 11.2). This indicates that these simple models produce biased estimates due to the omitted variables.

Regressions (4) to (6) use regression specifications that include different sets of applicant characteristics and credit rating indicator variables as well as interactions. However, most of the corresponding coefficients are not significant and the estimates of the coefficient on `black` obtained for these models as well as the estimated difference in denial probabilities do not deviate much from those obtained for the similar specifications (2) and (3).

An intersting question related to racial discrimination can be investigated using the probit model (6) where the interactions `blackyes:pirat` and `blackyes:hirat` are added to the the specification of model (4). If the coefficient on `blackyes:pirat` was different from zero, the effect of the payment-to-income ratio on the denial probability would be different for black and white applicants. Similarly, a non-zero coefficient on `blackyes:hirat` would indicate that loan officers weight the risk of bankrupt associated with a high loan-to-value ratio differently for black and white mortgage applicants. We can test whether these coefficients are jointly significant at the 5% level using an F-Test.

```
linearHypothesis(probit_HMDA_4,
                 test = "F",
                 c("blackyes:pirat=0","blackyes:hirat=0"),
                 vcov = vcovHC(probit_HMDA_4, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## blackyes:pirat = 0
## blackyes:hirat = 0
##
## Model 1: restricted model
## Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +
##     insurance + selfemp + single + hschool + unemp
##
## Note: Coefficient covariance matrix supplied.
##
```

```
##   Res.Df Df      F Pr(>F)
## 1   2366
## 2   2364  2 0.2616 0.7698
```

Since p-value ≈ 0.77 for this test, the null cannot be rejected. Nonetheless we cannot reject the hypothesis that there is no racial discrimination at all since the corresponding F-test has a p-value of about 0.002.

```
linearHypothesis(probit_HMDA_4,
                 test = "F",
                 c("blackyes=0","blackyes:pirat=0","blackyes:hirat=0"),
                 vcov = vcovHC(probit_HMDA_4, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## blackyes = 0
## blackyes:pirat = 0
## blackyes:hirat = 0
##
## Model 1: restricted model
## Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +
##     insurance + selfemp + single + hschool + unemp
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1   2367
## 2   2364  3 4.8886 0.002168 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary

Models (1) to (6) provide evidence that there is an effect of being African American on the probability of a mortgage application denial: in all specifications, the effect is estimated to be positive (ranging from 4% to 5% for the fictive individuals considered) and significantly different from zero at the 1% level. While the linear probability model seems to overestimate this effect slightly, it still can be used as an approximation to an intrinsic nonlinear relationship.

See Chapters 11.4 and 11.5 of the book for a discussion of external and internal validity of this study and some concluding remarks on regression models where the dependent variable is binary.

Chapter 13

Instrumental Variables Regression

As discussed in Chapter 9, regression models may suffer from problems like omitted variables, measurement errors and simultaneous causality. If so, the error term is correlated with the regressor of interest and there is a bias in estimation of the corresponding coefficients. If data is available, what we have assumed up till now, omitted varibales can be included in the regression to mitigate the risk of biased estimation of the causal effect of interest. However, if omitted factors cannot be measured, multiple regression cannot solve the problem. The same issue arises if there is simultaneous causality. When causality runs from X to Y and vice versa, there will be an estimation bias that cannot be corrected for by multiple regression.

A general technique for obtaining a consistent estimator of the coefficient of interest is instrumental variables (IV) regression. In this chapter we focus on a powerful IV regression tool called two stage least squares (TSLS). The first sections briefly recap the general mechanics and assumptions of IV regression and show how to apply TSLS estimation using `R`. Next, IV regression is used for estimating the elasticity of demand for cigarettes — a classical example where multiple regression fails to do the job because of simultaneous causality.

13.1 The IV Estimator with a Single Regressor and a Single Instrument

Cosider the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i \quad , \quad i = 1, \dots, n \tag{13.1}$$

where the error term u_i is correlated with the regressor X_i (we say that X is *endogenous*) such that OLS is inconsistent for the true β_1. In the most simple case, IV regression uses a single instrumental variable Z to obtain a consistent estimator for β_1.

Z must satisfy two condictions to be a valid instrument:

1. Instrument relevance condition:

X and its instrument Z *must be* correlated: $corr(Z_i, X_i) \neq 0$.

2. Instrument exogeneity condition:

The instrument Z *must not be* correlated with the error term u: $corr(Z_i, u_i) = 0$.

The Two Stage Least Squares Estimator

As can be guessed from its name, TSLS has to stages. In the first stage, the variation in the endogenous regressor X is decomposed into a problem-free component that is explained by the instrument Z and a problematic component that is correlated with the error u_i. The second stage uses the problem-free component of the variation in X to estimate β_1.

The first stage regression model is

$$X_i = \pi_0 + \pi_1 Z_i + \nu_i$$

where $\pi_0 + \pi_1 Z_i$ is the component of X_i that is explained by Z_i while ν_i is the component that cannot be explained by Z_i and exhibits correlation with u_i.

Using OLS estimates $\widehat{\pi}_0$ and $\widehat{\pi}_1$ we obtain predicted values $\widehat{X}_i \quad , \quad i = 1, \dots, n$. If Z is a valid instrument, the $\widehat{X}_i$ are problem-free in the sense that $\widehat{X}$ is exogeneous in a regression of Y on $\widehat{X}$ what is done in the second stage regression. The second stage produces $\widehat{\beta}_0^{TSLS}$ and $\widehat{\beta}_1^{TSLS}$, the TSLS estimates of β_0 and β_1.

For the case of a single instrument one can show that the TSLS estimator of β_1 is

$$\widehat{\beta}_1^{TSLS} = \frac{s_{ZY}}{s_{ZX}} = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \overline{Y})(Z_i - \overline{Z})}{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(Z_i - \overline{Z})} \tag{13.2}$$

which is nothing but the ratio of the sample covariance between Z and Y to the sample covariance between Z and X.

As shown in Appendix 12.3 of the book, (13.2) is a consistent estimator for β_1 in (13.1) under the assumption that Z is a valid instrument. Furthermore, as for every other OLS estimator that we have considered so far, the central limit theorem implies that $\widehat{\beta}_1^{TSLS}$ can be approximated by a normal distribution if the sample size is large. This allows us to use t-statistics and confidence intervals which are also computed by `R` functions relevant to us. A more detailed argument on the large-sample distribution of the TSLS estimator is sketched out in Appendix 12.3, too.

Application to the Demand For Cigarettes

The relation between demand and price of commodities is a simple yet widespread problem in many branches of economics. The field of health economics is inter alia concernced with the study of how health-affecting behaviour of individuals is influenced by the health-care system and regulation policy. The probably most prominent example in public policy debates is smoking as it is related to numerous illnesses and negative externalities.

It is plausible that cigarette consumption can be reduced by taxing cigarettes more heavily. The question is how much taxes must be increase to reach a noticeable reduction in cigarette consumption, e.g. 20 percentage points. Economists use elasticities to answer this kind of question. Since the price elasticity for demand of cigarettes is unknown, it must be estimated. As discussed in the box Who Invented Instrumental Variables Regression presented in Chapter 12.1 of the book, an OLS regression of log quantity on log price cannot be used to estimate the effect of interest since there is simultaneous causality between demand and supply. Instead, IV regression can be used.

We use the data `CigarettesSW` which comes with the package `AER`. It is a panel data set that contains observations on cigarette consumption and several economic indicators for all 48 continental US States from 1985 to 1995. Following the book, in what follows we consider data for the cross section of states in 1995 only.

We start by loading the package, attaching the data set and getting an overview.

```
library(AER)
data("CigarettesSW")
summary(CigarettesSW)
```

```
##      state       year         cpi          population
##  AL     : 2   1985:48   Min.   :1.076   Min.   :  478447
##  AR     : 2   1995:48   1st Qu.:1.076   1st Qu.: 1622606
##  AZ     : 2             Median :1.300   Median : 3697472
##  CA     : 2             Mean   :1.300   Mean   : 5168866
##  CO     : 2             3rd Qu.:1.524   3rd Qu.: 5901500
##  CT     : 2             Max.   :1.524   Max.   :31493524
##  (Other):84
##      packs             income               tax             price
##  Min.   : 49.27   Min.   :  6887097   Min.   :18.00   Min.   : 84.97
##  1st Qu.: 92.45   1st Qu.: 25520384   1st Qu.:31.00   1st Qu.:102.71
##  Median :110.16   Median : 61661644   Median :37.00   Median :137.72
##  Mean   :109.18   Mean   : 99878736   Mean   :42.68   Mean   :143.45
##  3rd Qu.:123.52   3rd Qu.:127313964   3rd Qu.:50.88   3rd Qu.:176.15
##  Max.   :197.99   Max.   :771470144   Max.   :99.00   Max.   :240.85
##
##       taxs
##  Min.   : 21.27
##  1st Qu.: 34.77
##  Median : 41.05
##  Mean   : 48.33
##  3rd Qu.: 59.48
##  Max.   :112.63
##
```

See `?CigarettesSW` for a detailed description of variables.

We are interested in estimating β_1 in

$$\log(Q_i^{cigarettes}) = \beta_0 + \beta_1 \log(P_i^{cigarettes}) + u_i \tag{13.3}$$

where $Q_i^{cigarettes}$ is the number of cigarette packs per capita sold and $P_i^{cigarettes}$ is the after tax average real price per pack of cigarettes in state i.

The instrumental variable we are going to use is $SalesTax$, the portion of taxes on cigarettes arising from the general sales tax. $SalesTax$ is measured in dollars per pack. The idea is that $SalesTax$ is a relevant instrument as it is included in the after tax average price per pack. Also it is plausible that $SalesTax$ is exogeneous since the sales tax does not influence to quantity sold directly but indirectly through the price.

Before computing the TSLS estimate for β_1 we have to do some transformations in order to obtain deflated cross section data for the year 1995.

We also compute the sample correlation between the sales tax and price per pack. Notice that the sample correlation is a consistent estimator of the pupulation. The estimate of approximately 0.614 indicates that $SalesTax$ and $P_i^{cigarettes}$ exhibit moderately positive correlation which is in accordance with our expectations: higher sales taxes lead to higher prices. However, a correlation analysis like this is not sufficient for checking whether the instrument is relevant. We will later come back to the issue of checking weather an instrument is relevant and exogenous.

```
# compute real per capita prices
CigarettesSW$rprice <- with(CigarettesSW, price/cpi)
```

```
# SalesTax
CigarettesSW$salestax <- with(CigarettesSW, (taxs - tax)/cpi)

# check correlation between sales tax and price
cor(CigarettesSW$salestax, CigarettesSW$price)
```

```
## [1] 0.6141228
```

```
# generate a subset for the year 1995
c1995 <- subset(CigarettesSW, year == "1995")
```

The first stage regression is

$$\log(P_i^{cigarettes}) = \pi_0 + \pi_1 SalesTax_i + \nu_i.$$

We estimate this model in `R` using `lm()`. In the second stage we run a regression of $\log(Q_i^{cigarettes})$ on $\log(\widehat{P_i^{cigarettes}})$ to obtain $\widehat{\beta}_0^{TSLS}$ and $\widehat{\beta}_1^{TSLS}$.

```
# load the sandwich package for robust standard errors
library(sandwich)

# Stage 1 regression
cig_s1 <- lm(log(rprice) ~ salestax, data = c1995)
coeftest(cig_s1, vcov = vcovHC(cig_s1, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 4.6165463  0.0289177 159.6444 < 2.2e-16 ***
## salestax    0.0307289  0.0048354   6.3549 8.489e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the first stage regression we obtain

$$\log(\widehat{P_i^{cigarettes}}) = \underset{(0.03)}{4.62} + \underset{(0.005)}{0.031} SalesTax_i$$

which also predicts the relation between sales tax price per cigarettes to be postive. How much of the observed variation in $\log(P^{cigarettes})$ is explained by the instrumental variable $SalesTax$? This can be answered by looking at the regression's R^2 which states that about 47% of the variation in after tax prices is explained by the variation of the sales tax across states.

```
summary(cig_s1)$r.squared
```

```
## [1] 0.4709961
```

We continue by saving $\log(\widehat{P_i^{cigarettes}})$, the fitted values predicted by the first stage regression `cig_s1`, to the variable `lcigp_pred`.

```
# Save predicted values
lcigp_pred <- cig_s1$fitted.values
```

Next, we run the second stage regression which gives the TSLS estimates we seek.

```
# Stage 2 regression
cig_s2 <- lm(log(c1995$packs) ~ lcigp_pred)
coeftest(cig_s2, vcov = vcovHC(cig_s2))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  9.71988    1.70304  5.7074 7.932e-07 ***
## lcigp_pred  -1.08359    0.35563 -3.0469  0.003822 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus estimating the model equation (13.3) using TSLS yields

$$\widehat{\log(Q_i^{cigarettes})} = \underset{(1.70)}{9.72} + \underset{(0.36)}{1.08} \log(P_i^{cigarettes}) \tag{13.4}$$

where we write $\log(P_i^{cigarettes})$ instead of $\widehat{\log(P_i^{cigarettes})}$ for convenience.

The function `ivreg()` from the package `AER` carries out TSLS automatically. It is used similarly as `lm()`. Instruments can be added to the usual specification of the regression using a | seperating the model equation from the instruments. Thus, for the regression and hand the correct formula is `log(packs) ~ log(rprice) | salestax`.

```
# TSLS using ivreg()
cig_ivreg <- ivreg(log(packs) ~ log(rprice) | salestax, data = c1995)

coeftest(cig_ivreg, vcov = vcovHC(cig_ivreg, type = "HC1"))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  9.71988    1.52832  6.3598 8.346e-08 ***
## log(rprice) -1.08359    0.31892 -3.3977  0.001411 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that the coefficient estimates obtain coincide for both approaches.

Two notes on calculation of TSLS standard errors

1. We have demonstrated that running the individual regressions for each stage of TSLS using lm() leads to the same coefficient estimates as when using ivreg(). However, the standard errors reported for the second-stage regression, e.g. by coeftest() or summary(), are *invalid*: they *do not* adjust for using predictions from the first-stage regression as regressors in the second-stage regression! Fortunately, ivreg() performs the necessary adjustment automatically. This is another advantage over manual step-by-step estimation.

2. Just as in mutliple regression, it is important to compute heteroskedasticity-robust standard errors as we have done above using vcovHC().

The TSLS estimate for β_1 in (13.4) means that an increase in cigarette prices by one percent reduces cigarette consumption by roughly 1.08 percentage points which is fairly elastic. However, we know that this estimate might not be trusworthy although we used IV estimation: there still might be a bias due to omitted variables. Thus a multiple IV regression approach is needed.

13.2 The General IV Regression Model

The simple IV regression model is easily extended to a multiple regression model which we refer to as the general IV regression model. In this model we distinguish between four types of variables: the dependent variable, included exogenous variables, included endogenous variables and instrumental variables. Key Concept 12.1 summarizes the model and the common terminology. See Chapter 12.2 of the book for a more comprehensive discussion of the individual components of the general model.

Key Concept 12.1

The General Instrumental Variables Regression Model and Terminology

$$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \beta_{k+1} W_{1i} + \beta_{k+r} W_{ri} + u_i, \tag{13.5}$$

$i = 1, \ldots, n$ is the general intrumental variables regression model where

- Y_i is the dependent variable
- $\beta_1, \ldots, \beta_k$ are k unknown regression coefficients
- $X_{1i}, \ldots, X_{ki}$ are k endogenous regressors which are potentially correlated with u_i
- $W_{1i}, \ldots, W_{ri}$ are r exogenous regressors which are uncorrelated with u_i
- u_i is the error term
- $Z_{1i}, \ldots, Z_{mi}$ are m instrumental variables

We say that the coefficients are overidentified if $m > k$ thats is if there are more instruments than regressors. If $m < k$, the coefficients are underidentified and when $m = k$ they are exactly identified. For estimation of the IV regression model exact identification or overidentification is required.

While computing both stages of TSLS individually is not a big deal in the simple regression model with a single endogenous regressor (13.1), Key Concept 12.2 clarifies why resorting to TSLS estimation functions like `ivreg()` is much more convenient when the set of potentially endogenous regressors (and instruments) is large.

Estimating regression models with TSLS using multiple instruments by means of `ivreg()` is straightforward. There are, however, some subtleties when it comes to correct specification of the regression formula that should be mentioned.

Assume that you want to estimate the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + W_1 + u$$

where X_1 and X_2 are endogenous regressors that shall be instrumented by Z_1, Z_2 and Z_3 and W_1 is an exogenous regressor. The corresponding data is available in a `data.frame` with coloumn names `y`, `x1`, `x1`, `w1`, `z1`, `z2` and `z3`. It might be tempting to specify the argument `formula` in your call of `ivreg()` as `y ~ x1 + x2 + w1 | z1 + z2 + z3` which is wrong. As explained in the documentation of `ivreg()` (see `?ivreg`), it is necessary to list *all* exogenous variables as instruments, that is joining them by `+`'s on the right of `|`: `y ~ x1 + x2 + w1 | w1 + z1 + z2 + z3` where `w1` is instrumenting itself.

If there is a large number of exogenous variables it may be convenient to provide an update formula with a `.` (this includes all variables except for the dependent variable) right after the `|` and to exclude all endogenous variables using `-`.

Key Concept 12.2

Two Stage Least Squares

Similarly to the simple IV regression model, the gerneral IV model (13.5) can be estimated using the TSLS estimator which is computed in two stages:

1. **First-stage regression(s)**: Run an OLS regression for each of the endogenous variables $(X_{1i}, \ldots, X_{ki})$ on all instrumental variables $(Z_{1i}, \ldots, Z_{mi})$ and all exogenous variables $(W_{1i}, \ldots, W_{ri})$ and an intercept. Compute the fitted values $(\widehat{X}_{1i}, \ldots, \widehat{X}_{ki})$.

2. **Second-stage regression**: OLS regression of the dependent variable on predicted values of all endogenous regressors and all exogenous variables and an intercept. This delivers $\widehat{\beta}_0^{TSLS}, \ldots, \widehat{\beta}_{k+r}^{TSLS}$ which are called the TSLS estimates.

In the general IV regression model the instrument relevance and exogeneity assumptions are just as in the simple regression model with a single endogenous regressor and only one instrument. See Key Concept 12.3 for a recap using terminology of gerneral IV regression.

Key Concept 12.3

Two Conditions for Valid Instruments

For $Z_{1i}, \ldots, Z_{mi}$ to be a set of valid instruments, the following two conditions must be fulfilled:

1. **Instrument Relevance**: If there are k endogenous variables, r exogenous variables and $m \geq k$ instruments Z and the $\widehat{X}_{1i}^*, \ldots, \widehat{X}_{ki}^*$ are the predicted values from the k population stage one regressions, it *must hold that*

$$(\widehat{X}_{1i}^*, \ldots, \widehat{X}_{ki}^*, W_{1i}, \ldots, W_{ri}, 1)$$

   are not perfectly multicollinear. 1 denotes the constant regressor which equals 1 for all observations.

Note: If there is only one endogenous regressor X_i, there must be at least one non-zero coefficient on the Z's and the W's in the population regression for this condition to be valid: If all of these coefficients are zero, all the $\widehat{X}_i^*$ are just the mean of X such that there is perfect multicollinearity.

2. **Instrument Exogeneity**:

All the m instruments are uncorrelated with the error term:

$$corr(Z_{1i}, u_i) = 0, \ldots, corr(Z_{mi}, u_i) = 0$$

One can show that if the IV regression assumptions presented in Key Concept 12.4 hold, the TSLS estimator in the general IV model (13.5) is consistent and normally distributed when the sample size is large. To get an idea of the proof for this statement, see Appendix 12.3 of the book for a disussion of the special case with a single regressor, a single instrument and no exogenous variables. The reasoning behind this carries over to the general IV model. A more complicated explanation for the general case can be found in Chapter 18 of the book.

For the purpose of this book it is sufficient to bear in mind that validity of the assumptions stated in Key Concept 12.4 allows us to obtain valid statistical inference using `R` functions used to compute t-Tests, F-Tests and confidence intervals for model coefficients.

Key Concept 12.4

The IV Regression Assumptions

For the general IV regression model in Key Concept 12.1 we assume that

1. $E(u_i|W_{1i}, \ldots, W_{ri}) = 0$
2. $(X_{1i}, \ldots, X_{ki}, W_{1i}, \ldots, W_{ri}, Z_{1i}, \ldots, Z_{mi})$ are i.i.d. draws from their joint distribution
3. All X's, W's, Z's and Y's have nonzero finite fourth moments i.e. outliers are unlikely
4. The Z's are valid instruments (see Key Concept 12.3)

Application to the Demand for Cigarettes

The estimated elasticity of demand for cigarettes in (13.1) is 1.08. Although (13.1) was estimated using IV regression it is plausible that this IV estimate is biased: in this model, the TSLS estimator is inconsistent for the true β_1 if the instrument (the real sales tax per pack) correlates with the error term. This is likely to be the case since there are economic facors like state income which impact the demand for cigarettes and correlate with the sales tax. States with high personal income tend to generate tax revenues more by income taxes and less by sales taxes. Consequently, state income needs to be included in the regression model.

$$\log(Q_i^{cigarettes}) = \beta_0 + \beta_1 \log(P_i^{cigarettes}) + \beta_2 \log(Income_i) + u_i \tag{13.6}$$

Before estimating (13.6) using `ivreg()` we define $Income$ as real per capita income `rincome` and add it to the data set.

```
# add rincome to the data set
CigarettesSW$rincome <- with(CigarettesSW, income/population/cpi)
c1995 <- subset(CigarettesSW, year == "1995")
```

```
# estimate the model
cig_ivreg2 <- ivreg(log(packs) ~ log(rprice) + log(rincome) | log(rincome) + salestax,
  data = c1995)

coeftest(cig_ivreg2, vcov = vcovHC(cig_ivreg2, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   9.43066    1.25939  7.4883 1.935e-09 ***
## log(rprice)  -1.14338    0.37230 -3.0711  0.003611 **
## log(rincome)  0.21452    0.31175  0.6881  0.494917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain

$$\widehat{\log(Q_i^{cigarettes})} = \underset{(1.26)}{9.42} - \underset{(0.37)}{1.14} \log(P_i^{cigarettes}) + \underset{(0.31)}{0.21} \log(Income_i). \tag{13.7}$$

Following the book we add the cigarette-specific taxes ($cigtax_i$) as a further instrumental variable and estimate again using TSLS.

```
# add cigtax to the data set
CigarettesSW$cigtax <- with(CigarettesSW, tax/cpi)
c1995 <- subset(CigarettesSW, year == "1995")
```

```
cig_ivreg3 <- ivreg(log(packs) ~ log(rprice) + log(rincome) |
                      log(rincome) + salestax + cigtax, data = c1995)

coeftest(cig_ivreg3, vcov = vcovHC(cig_ivreg3, type = "HC1"))
```

```
##
## t test of coefficients:
##
```

```
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   9.89496    0.95922 10.3157 1.947e-13 ***
## log(rprice)  -1.27742    0.24961 -5.1177 6.211e-06 ***
## log(rincome)  0.28040    0.25389  1.1044    0.2753
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the two instrumental variables $salestax_i$ and $cigtax_i$ we have $m = 2$ and $k = 1$ so the coefficient on the endogenous regressor $\log(P_i^{cigarettes})$ is *overidentified*. The TSLS estimate of the regression function (13.6) is

$$\widehat{\log(Q_i^{cigarettes})} = \underset{(0.96)}{9.89} - \underset{(0.25)}{1.28}\log(P_i^{cigarettes}) + \underset{(0.25)}{0.28}\log(Income_i). \tag{13.8}$$

Should we trust the estimates presented in (13.7) or rather rely on (13.8)? Comparing both equations we find that the estimation based on both instruments is more precise since in (13.8) all standard errors reported are smaller than in (13.7). In fact, the standard error for the estimator of the demand elasticity is only two thirds of the standard error when the sales tax is the only instrument used. This is due to more information being used in estimation (13.8). *If* the instruments are *valid*, (13.8) can be considered more reliable.

Without any inference on validity of both instruments it is not sensible to make a statement like that. This stresses why checking instrument validity is essential. Chapter 12.3 briefly disscusses guidelines in checking instrument validity and presents approaches that allow to test for instrument relevance and exogeneity under certain conditions. These are then used in an application to the demand for cigarettes in Chapter 12.4.

13.3 Checking Instrument Validity

Instrument Relevance

Instruments that explain little variation in the endogenous regressor X are called **weak instrument**. Weak instruments provide little information about the variation in X that is exploited by IV regression to estimate the effect of interest: the estimate of the coefficient on the endogenous regressor is estimated inaccurately. Moreover, weak instruments cause the distribution of the estimator to deviate considerably from a normal distribution even in large samples such that known methods for obtaining inference about the true coefficient on X may be invalid. See Chapter 12.3 and Appendix 12.4 of the book for a more detailed argument on the undesirable consequences of using weak instruments in IV regression.

Key Concept 12.5

A Rule of Thumb for Checking for Weak Instruments

Consider the case of a *single* endogenous regressor X and m exogenous instruments $Z_1, \dots, Z_m$. If the coefficients on all instruments in the population first-stage regression of a TSLS estimation are zero, the instruments do not explain any of the variation in the X which clearly violates assumption 1 of Key Concept 12.2. Though the latter case is unlikely to be encountered in practice, we should ask ourselves to what extend the assumption of instrument relevance should be fulfilled.

While this is hard to answer for general IV reggression, in the case of a *single* endogenous regressor X one may use the following rule of thumb:

Compute the F-statistic which corresponds to the hypothesis that the coefficients on $Z_1, \dots, Z_m$ are all zero in the first-stage regression. If the F-statistic is less than 10 the instruments are weak such that the TSLS estimate of the coefficient on X is biased and no valid statistical inference about its true value can be made. See also Appendix 12.5 of the book.

The rule of thumb stated in Key Concept 12.5 is easily implemented in R. One simply has to run the first stage regression using `lm()` and subsequently compute the heteroskedasticity-robust F-statistic by means of `linearHypothesis()`. This is part of the application to the demand for cigarettes discussed in Chapter 12.4.

If instruments are weak

There are two options if instruments are weak:

1. Discard the weak instruments and/or find stronger instruments. While the former is only an option if the unknown coefficients remain identified when the weak instruments are discarded, the latter can be very difficult and even may require a redesign of the whole study.
2. Stick with the weak instruments but use methods that improve upon TSLS in this scenario, for example limited information maximum likelihood estimation, see Appendix 12.5 of the book.

When the Assumption of Instrument Exogeneity is Violated

If there is correlation between an instrument and the error term IV regression is not consistent estimator (this is shown in Appendix 12.4 of the book). As econometricians we wish to use the available information for gaining statistical inference whether an instrument is exogenous or not. The overidentifying restrictions test (also called the J-test) is an approach to test the hypothesis that *additional* instruments are exogenous under the assumption that the set of instruments consists of enough valid instruments to identify the coefficients of interest. For the J-test to be applicable we there need to be more instruments than endogenous regressors. The idea is summarized in Key Concept 12.5.

Key Concept 12.5

J-Statistic / Overidentifying Restriction Test

Take $\widehat{u}_i^{TSLS}$, $i = 1, \dots, n$, the residuals of the TSLS estimation of the general IV regression model (13.5). Run the OLS regression

$$\widehat{u}_i^{TSLS} = \delta_0 + \delta_1 Z_{1i} + \cdots + \delta_m Z_{mi} + \delta_{m+1} W_{1i} + \cdots + \delta_{m+r} W_{ri} + e_i \tag{13.9}$$

and test the hypothesis

$$H_0 : \delta_1 = 0, \dots, \delta_m = 0$$

which states that all instruments are exogenous. This can be done using the corresponding F-statistic by computing

$$J = mF.$$

This test is the overidentifying restrictions test and the statistic is called the J-statistic with

$$J \sim \chi^2_{m-k}$$

in large samples and the assumption of homoskedasticity. The degress of freedom $m - k$ state the degree of overidentification since this is the number of instruments m minus the number of endogenous regressors k.

It is important to note that the J-statistic discussed in Key Concept 12.5 is only χ^2_{m-k} distributed when the error term e_i in the regression (13.9) is homoskedastic. Discussion of the heteroskedasticity-robust J-statistic is beyond the scope of this chapter the reader is refered to Section 18.7 for a theoretical argument.

As for the procedure shown in Key Concept 12.5, further interest lies in applying the J-test using the function `linearHypothesis()` in R which will be shown in context of estimation of the demand elasticity for cigarettes discussed in the next chapter.

13.4 Application to the Demand for Cigarettes

Are the general sales tax and the cigarette-specific tax valid instruments? If not the TSLS attempt to estimate the demand elasticity for cigarettes discussed in Chapter 12.2 is not meaningful and we need to rethink the design of the study. As discussed in Chapter 12.1 both variables are likely do be relevant but whether they are exogenous is a different question.

In the book it is advocated that cigarette-specific taxes could be endogenous because there might be state specific historical factors like economic importance of the tabacco farming and cigarette production industry that enforces its interest in low cigarette specific taxes. Since it is plausible that tabacco growing states have higher rates of smoking than others this would lead to endogeneity of cigarette specific taxes. If data about the size on the tabacco and cigarette industry would be available we could solve this potential issue by including the information in the regression which is not the case.

However, since the role of the influence of the tabacco and cigarette industry is a factor that can be assumed to differ across states but not over time we may exploit the panel structure of `CigarettesSW` instead: as shown in Chapter 10.2, regression using data on *changes* between two time periods eliminates such state specific and time invariant effects. Following the book we consider changes in variables between 1985 and 1995 that is we are interested in estimating the *long-run elasticity* of the demand for cigarettes.

The model to be estimated by TSLS using the general sales tax and the cigarette-specific sals tax as instruments hence is

$$\log(Q_{i,1995}^{cigarettes}) - \log(Q_{i,1995}^{cigarettes}) = \beta_0 + \beta_1 \left[\log(P_{i,1995}^{cigarettes}) - \log(P_{i,1985}^{cigarettes})\right] \tag{13.10}$$

$$+ \beta_2 \left[\log(Income_{i,1995}) - \log(Income_{i,1985})\right] + u_i. \tag{13.11}$$

Prior to the estimation using `ivreg()` we define changes from 1985 to 1995 for the dependent variable, the regressors and both instruments.

```
# subset data for year 1985
c1985 <- subset(CigarettesSW, year == "1985")

# define changes in variables
packsdiff <- log(c1995$packs) - log(c1985$packs)

pricediff <- log(c1995$price/c1995$cpi) - log(c1985$price/c1985$cpi)

incomediff <- log(c1995$income/c1995$population/c1995$cpi) -
log(c1985$income/c1985$population/c1985$cpi)

salestaxdiff <- (c1995$taxs - c1995$tax)/c1995$cpi - (c1985$taxs - c1985$tax)/c1985$cpi

cigtaxdiff <- c1995$tax/c1995$cpi - c1985$tax/c1985$cpi
```

We perform three estimations of (13.11) using `ivreg()`:

1. 2SLS using only the difference in the sales taxes between 1985 and 1995 as the instrument.
2. 2SLS using only the dfference in the cigarette-specific sales taxes 1985 and 1995 as the instrument.
3. 2SLS using both the difference in the sales taxes 1985 and 1995 and the dfference in the cigarette-specific sales taxes 1985 and 1995 as instruments.

```
# estimate models
cig_ivreg_diff1 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff + salestaxdiff)
```

```
cig_ivreg_diff2 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff + cigtaxdiff)

cig_ivreg_diff3 <- ivreg(packsdiff ~ pricediff + incomediff | incomediff + salestaxdiff + cigtaxdiff)
```

As usual we use `coeftest()` in conjunction with `vcovHC()` to obtain robust coefficient summaries for all three models.

```
# robust coefficient summary 1
coeftest(cig_ivreg_diff1, vcov = vcovHC(cig_ivreg_diff1, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.117962   0.068217 -1.7292   0.09062 .
## pricediff   -0.938014   0.207502 -4.5205 4.454e-05 ***
## incomediff   0.525970   0.339494  1.5493   0.12832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# robust coefficient summary 2
coeftest(cig_ivreg_diff2, vcov = vcovHC(cig_ivreg_diff2, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.017049   0.067217 -0.2536    0.8009
## pricediff   -1.342515   0.228661 -5.8712 4.848e-07 ***
## incomediff   0.428146   0.298718  1.4333    0.1587
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# robust coefficient summary 3
coeftest(cig_ivreg_diff3, vcov = vcovHC(cig_ivreg_diff3, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.052003   0.062488 -0.8322    0.4097
## pricediff   -1.202403   0.196943 -6.1053 2.178e-07 ***
## incomediff   0.462030   0.309341  1.4936    0.1423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We proceed by generating a tabulated summary of the estimation results using `stargazer()`.

```
# gather robust standard errors in a list
rob_se <- list(
  sqrt(diag(vcovHC(cig_ivreg_diff1, type = "HC1"))),
  sqrt(diag(vcovHC(cig_ivreg_diff2, type = "HC1"))),
  sqrt(diag(vcovHC(cig_ivreg_diff3, type = "HC1")))
)
```

2SLS Estimates of the Long-Term Elasticity of the Demand for Cigarettes using Panel Data

Dependent variable: 1985-1995 difference in log per pack price

IV: salestax

IV: cigtax

IVs: salestax, cigtax

(1)

(2)

(3)

pricediff

-0.94***

-1.34***

-1.20***

(0.21)

(0.23)

(0.20)

incomediff

0.53

0.43

0.46

(0.34)

(0.30)

(0.31)

Constant

-0.12*

-0.02

-0.05

(0.07)

(0.07)

(0.06)

Observations

48

48

48

R2

0.55

0.52

0.55

Adjusted R2

0.53

0.50

0.53

Residual Std. Error (df = 45)

0.09

0.09

0.09

We observe negative estimates of the coefficient on `pricediff` that are quite different in magnitude. Which one should we trust most? This hinges only on the validity of instruments used. To assess this we compute F-statistics for the first-stage regression of all three models to check instrument relevance.

```
# First-stage regressions
mod_relevance1 <- lm(pricediff ~ salestaxdiff + incomediff)
mod_relevance2 <- lm(pricediff ~ cigtaxdiff + incomediff)
mod_relevance3 <- lm(pricediff ~ incomediff + salestaxdiff + cigtaxdiff)
```

```
# check instrument relevance for model 1
linearHypothesis(mod_relevance1,
                 "salestaxdiff = 0",
                 vcov = vcovHC(mod_relevance1, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## salestaxdiff = 0
##
## Model 1: restricted model
## Model 2: pricediff ~ salestaxdiff + incomediff
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     46
## 2     45  1 33.674 6.119e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# check instrument relevance for model 2
linearHypothesis(mod_relevance2,
                 "cigtaxdiff = 0",
                 vcov = vcovHC(mod_relevance2, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## cigtaxdiff = 0
##
## Model 1: restricted model
## Model 2: pricediff ~ cigtaxdiff + incomediff
##
## Note: Coefficient covariance matrix supplied.
##
```

```
##   Res.Df Df      F    Pr(>F)
## 1     46
## 2     45  1 107.18 1.735e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# check instrument relevance for model 3
linearHypothesis(mod_relevance3,
                 c("salestaxdiff = 0", "cigtaxdiff = 0"),
                 vcov = vcovHC(mod_relevance3, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## salestaxdiff = 0
## cigtaxdiff = 0
##
## Model 1: restricted model
## Model 2: pricediff ~ incomediff + salestaxdiff + cigtaxdiff
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     46
## 2     44  2 88.616 3.709e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We also conduct the overidentifying restrictions test for model three which is the only model where the coefficient on the difference in log prices is overidentified ($m = 2$, $k = 1$) such that the J-statistic can be computed. To do this we take the residuals stored in `cig_ivreg_diff3` and regress them on both instruments and the presumably exogenous regressor `incomediff`. Yet again use `linearHypothesis()` to test jointly whether the coefficients on both instruments are zero which is necessary for the exogeneity assumption to be fulfilled. Note that with `test = "Chisq"` we obtain a chi-squared distributed test statistic instead of a F-statistic.

```
# compute J-statistic

cig_iv_OR <- lm(residuals(cig_ivreg_diff3) ~ incomediff + salestaxdiff + cigtaxdiff)

cig_OR_test <- linearHypothesis(cig_iv_OR,
                                c("salestaxdiff = 0", "cigtaxdiff = 0"),
                                test = "Chisq")
cig_OR_test
```

```
## Linear hypothesis test
##
## Hypothesis:
## salestaxdiff = 0
## cigtaxdiff = 0
##
## Model 1: restricted model
## Model 2: residuals(cig_ivreg_diff3) ~ incomediff + salestaxdiff + cigtaxdiff
##
##   Res.Df     RSS Df Sum of Sq Chisq Pr(>Chisq)
## 1     46 0.37472
```

```
## 2     44 0.33695  2  0.037769 4.932    0.08492 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Caution: In this case the p-Value reported by `linearHypothesis()` is wrong because the degrees of freedom are set to 2. This differs from the degree of overidetification ($m - k = 2 - 1 = 1$) so the J-statistic is χ^2_1 distributed instead of following a χ^2_2 distribution as assumed defaultly `linearHypothesis()`. We may compute the correct p-Value using `pchisq()`.

```
# compute correct p-value for J-statistic
pchisq(cig_OR_test[2,5], df = 1, lower.tail = FALSE)
```

```
## [1] 0.02636406
```

Since this value is smaller than 0.05 we reject the hypothesis that both instruments are exogenous at the level of 5%. This means one the following:

1. The sales tax is an invalid instrument.
2. The cigarettes-specific sales tax is an invalid instrument.
3. Both instruments are invaild.

In the book it is argued that the assumption of instrument exogeneity is more likely to hold for the gerneral sales tax (see Chapter 12.4 of the book) such that the IV estimate of the long-run elasticity of demand for cigarettes we consider the most trustworthy is -0.94, the estimate obtained using 2SLS with only the general sales tax as an instrument.

The interpretation of this estimate is that over a 10-year period, an increase in the average price per package by one percent is expected to decrease consumption by about 0.94 percentage points. This suggests that, in the long run, price increases can reduce cigarette consumption considerably.

13.5 Where Do Valid Instruments Come From?

Chapter 12.5 of the books presents a comprehensive discussion of approaches to find valid instruments in practice by the example of three research questions:

1. Does putting criminals in jail reduce crime?
2. Does cutting class sizes invrease test scores?
3. Does aggressive treatment of heart attacks prolong lives?

This section is not directly related to applications in R which is why we do not discuss the contents here. We encourage you to work through this on your own.

Summary

`ivreg()` from the package AER provides convenient functionalities to estimate IV regression models in R. It is an implementation of the 2SLS estimation approach.

Besides treating IV estimation, we have also discussed how to test for weak instruments and how to conduct an overidentifying restrictions test when there are more instrumental variables than endogenous regressors using R.

In an empirical application we have shown how `ivreg()` can be used to estimate the long-run elasticity of demand for cigarettes based on `CigarettesSW`, a panel data set on cigarette consumption and economic indicators for all 48 continental US states for time periods 1985 and 1995. Different sets of instruments were used in the process and it has been argued why using the general sales tax as the only instrument is the prefered choice here. The estimate of the demand elasticity deemed the most trustworthy is -0.94. This

estimate suggests that there is a remarkable negative long-run effect on cigarette consumption of increasing the average price per pack.

Chapter 14

Experiments and Quasi-Experiments

This chapter discusses statistical tools that are commonly applied in program evaluation which is a field of study where resarchers are interested in measuring the causal effects of programs, policies or other interventions / treatments. An optimal research design for this purpose is what statisticains call an ideal randomized controlled experiment. The basic idea is to randomly assign subjects in two different groups, one that receives the treatment (treatment group) and one that does not (control group) and to compare outcomes for both groups in order to get an estimate of the average treatment effect.

Note that *experimental* data is fundamentally different from *observational* data. For example, one might be interested in measuring how much the performance of students in a standardized test differs between two classes where one of which has a student-teacher ratio which is considered regular and the other one has fewer students using a randomized controlled experiment. The data produced by such an experiment would be different from e.g. the observed cross-section data on the students' performance used throughout Chapters 4 to 8 where class sizes are *not randomly assigned* to students but are the results of an economic dicision where educational objectives and budgetary aspects where balanced.

For economists, randomized controlled experiments are often difficult to implement or even infeasable. For example, due to ethic, moral and legal reasons it is practically impossible for a business owner to estimate the causal effect the productivity of workers of setting them under psychological stress using an experiment where workers are randomly assigned either to the treament group that is pressed under time or to the control group where work is under regular conditions, at best without knowlegde of beeing in an experiment.[1]

However sometimes it is the case that external cirumstances which appear random produce what is called a *quasi-experiment* or *natural experiment*. This "as if" randomness allows for estimation of causal effects that are of interest for economist using tools which are very similar to those valid for ideal randomized controlled experiments. These tools draw heavily on the theory of multiple regression and also on IV regression (see chapter 12). We will review the core aspects of these methods and demonstrate how to apply them in R using the STAR data set.[2]

14.1 Potential Outcomes, Causal Effects and Idealized Experiments

In the following we will briefly recap the idea of the average causal effect and how it can be estimated using the *differences estimator*. We advise You to read through Chapter 13.1 of the book for a better understanding.

[1] See the box *The Hawthorne Effect* on page 528 of the book.
[2] See the description of the data set.

Potential Outcomes and the average causal Effect

A potential outcome is the outcome for an individual under a potential treatment. For this individual, we say that the causal effect of the treatment is the difference in the potential outcome if the indiviual receives the treatment and the potential outcome if she does not. Since this causal effect may be different for different individuals and it is not possible to measure the causal effect for a single individual, one is interested in studying the **average causal effect** of the treatment, hence also called the **average treatment effect.**

In a ideal randomized controlled experiment the following conditions are fulfilled:

1. The subjects are selected at random from the population
2. The subjects are randomly assigned to treatment and control group

Condition 1. guarantees that the subjects' potential outcomes are drawn randomly from the same distribution such that the expected value of the causal effect in the sample is equal to the average causual effect in the distribution. Condition 2. ensures that receipt of treamtment is independent from the subjects potential outcomes. If both conditions are fulfilled, the expected causual effect is the expected outcome in the treatment group minus the expected outcome in the control group. Using conditional expecations we have

$$\text{Average causal effect} = E(Y_i|X_i = 1) - E(Y_i|X_i = 0)$$

where X_i is a binary treatment indicator.

The average causal effect can be estimated using the *differences estimator*, which is nothing but the OLS estimator in the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i \quad , \quad i = 1, \dots, n \tag{14.1}$$

where random assignment ensures $E(u_i|X_i) = 0$.

The OLS estimator in the regression model

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_{1i} + \cdots + \beta_{1+r} W_{ri} + u_i \quad , \quad i = 1, \dots, n \tag{14.2}$$

with additional regressors $W_1, \dots, W_r$ is called the *differences estimator with additional regressors*. It is assumed that conditional mean independence holds, i.e.

$$E(u_i|X_i, W_i) = E(u_i|W_i) = 0.$$

The *differences estimator with additional regressors* is more efficient than the *differences estimator* if the additional regressors explain some of the variation in Y_i.

14.2 Threats to Validity of Experiments

The concepts of internal and external validity discussed in Key Concept 9.1 are also applicable for studies based on experimental and quasi-experimental data. Chapter 13.2 of the book provides a sound explanation of the particular threats to internal and external validity of experiments including examples. We limit ourselves to a short repetition of the threats listed. Consult the book for a more detailed explanation.

Threats to Internal Validity

1. **Failure to randomize**

   If the subjects are not randomly assigned to the treatment group, then the outcomes will be contaminated with the effect of the subjects' individuals characteristics or preferences and it is not possible to obtain an unbiased estimate of the treatment effect. One can test for nonrandom assignment using a significance test (F-Test) on the coefficient in the regression model

   $$X_i = \beta_0 + \beta_1 W_{1i} + \cdots + \beta_2 W_{ri} + u_i \quad , \quad i = 1, \ldots, n.$$

2. **Failure to follow the treatment protocol**

   If subjects do not follow the treatment protocol, i.e. some subjects in the treatment group manage to avoid receiving the treatment and/or some subjects in the control group manage to receive the treatment (*partial compliance*), there is correlation between X_i und u_i such that the OLS estimator of the average treatment effect will be biased. If there are data on *both* treatment actually recieved (X_i) and initial random assignment (Z_i), IV regression of the models (14.1) and (14.2) is a remedy.

3. **Attrition**

   Attrition results in a nonrandomly selected sample. If subjects systematically drop out of the study after beeing assigned to the control or the treatment group, where systematic means that the reason of the drop out is related to the treatment, there will be correlation between X_i and u_i and hence bias in the OLS estimator of the treatment effect.

4. **Experimental effects**

   If human subjects in treatment group and/or control group know that they are in an experiment, they might adapt their behaviour in a way that prevents unbiased estimation of the treatment effect.

5. **Small sample sizes**

   As we know from the theory of linear regression, small sample sizes lead to imprecise estimation of coefficients in regression models and thus imply imprecise estimation of the causal effect. Furthermore, confidence intervals and hypothesis test may produce wrong inference when the sample size is small.

Threats to External Validity

1. **Nonrepresentative sample**

   If the population studied and the population of interest are not sufficiently similar, there is no justification in generalizing the results.

2. **Nonrepresentative program or policy**

   If the program or policy differs for the population studied differs considerably from the program (to be) applied to population(s) interest, the results cannot be generalized. For example, a small-scale programm with low funding might have different effects than the widely available scaled-up programm that is actually implemented. There are other factors like duration and the extend of monitoring that should be considered here.

3. **General equilibrium effects**

   If market and/or environmental conditions cannot be kept constant when an internally valid program is implemented broadly, the external validity may be doubted.

14.3 Experimental Estimates of the Effect of Class Size Reductions

Experimental Design and the Data Set

The Project *Sudent-Teacher Achievement Ratio* (*STAR*) was a large randomized controlled experiment with the aim of ascerting whether a class size reduction is effective in improving education. It has been conducted in 80 Tennesse elementary schools over a period of four years during the 1980s by the state Derpartment of Education.

In the first year, about 6400 students were randomly assigned into one of three interventions: small class (13 to 17 students per teacher), regular class (22 to 25 students per teacher), and regular-with-aide class (22 to 25 students with a full-time teacher's aide). Classroom teachers were also randomly assigned to the classes they would teach. The interventions were initiated as the students entered school in kindergarten and continued through third grade. Thus, we have the following structure of control and treatment groups across grades:

Table 14.1: Control and treatment groups in the STAR experiment

	K	1	2	3
Treatment 1	Small class	Small class	Small class	Small class
Treatment 2	Regular class + aide	Regular class + aide	Regular class + aide	Regular class + aide
Control	Regular class	Regular class	Regular class	Regular class

Each year, the students learning progress was assessed using the sum of the points scored on the math and reading portion of a standardized test (the Stanford Achievement Test).

The *STAR* data set is part of the package AER.

```
# Load the package AER and the STAR data set
library(AER)
data(STAR)
```

Using `head(STAR)` we find that there is a variety of factor variables that describe student and teacher characteristics as well as various school indicators, all of which seperately recorded for the four different grades. The data is in *wide format*, that is each variable has its own coloumn and for each student, the rows contain observations on these variables. Using `dim(STAR)` we find that there are a total of 11598 observations on 47 variables.

```
# get an overview
head(STAR, 2)
```

```
##      gender ethnicity   birth stark star1 star2   star3 readk read1 read2
## 1122 female      afam 1979 Q3  <NA>  <NA>  <NA> regular    NA    NA    NA
## 1137 female      cauc 1980 Q1 small small small   small   447   507   568
##      read3 mathk math1 math2 math3   lunchk lunch1   lunch2 lunch3 schoolk
## 1122   580    NA    NA    NA   564     <NA>   <NA>     <NA>   free    <NA>
## 1137   587   473   538   579   593 non-free   free non-free   free   rural
##      school1 school2  school3  degreek  degree1  degree2  degree3 ladderk
## 1122    <NA>    <NA> suburban     <NA>     <NA>     <NA> bachelor    <NA>
## 1137   rural   rural    rural bachelor bachelor bachelor bachelor  level1
##      ladder1    ladder2    ladder3 experiencek experience1 experience2
## 1122    <NA>       <NA>     level1          NA          NA          NA
```

```
## 1137  level1 apprentice apprentice           7           7           3
##      experience3 tethnicityk tethnicity1 tethnicity2 tethnicity3 systemk
## 1122          30        <NA>        <NA>        <NA>        cauc    <NA>
## 1137           1        cauc        cauc        cauc        cauc      30
##      system1 system2 system3 schoolidk schoolid1 schoolid2 schoolid3
## 1122    <NA>    <NA>      22      <NA>      <NA>      <NA>        54
## 1137      30      30      30        63        63        63        63
```

```
dim(STAR)
```

```
## [1] 11598    47
```

```
# get variable names
names(STAR)
```

```
##  [1] "gender"      "ethnicity"   "birth"       "stark"       "star1"
##  [6] "star2"       "star3"       "readk"       "read1"       "read2"
## [11] "read3"       "mathk"       "math1"       "math2"       "math3"
## [16] "lunchk"      "lunch1"      "lunch2"      "lunch3"      "schoolk"
## [21] "school1"     "school2"     "school3"     "degreek"     "degree1"
## [26] "degree2"     "degree3"     "ladderk"     "ladder1"     "ladder2"
## [31] "ladder3"     "experiencek" "experience1" "experience2" "experience3"
## [36] "tethnicityk" "tethnicity1" "tethnicity2" "tethnicity3" "systemk"
## [41] "system1"     "system2"     "system3"     "schoolidk"   "schoolid1"
## [46] "schoolid2"   "schoolid3"
```

Notice that a majority of the variable names contain a suffix (`k`, `1`, `2` or `3`) stating the grade which the respective variable is refering to. This facilitates regression analysis because it allows to adjust the `formula` argument in `lm()` for each grade by simply changing the variables' suffixes accordingly.

Inspecting the outcome produced by `head()` we see that some values recorded are `NA` and `<NA>`, i.e. there is no data of this variable for the student under consideration. This lies in the nature of the data: for example, take the first observation `STAR[1,]`. We find that the student entered the experiment in third grade in a regular class which is why the class size is recorded in `star3` and other class type indicator variables are `<NA>`. For the same reason there are no recordings of her math and reading score but for third grade. It is straightforward to get only the non-`NA`/`<NA>` recordings for her: we simply drop the `NA`s using `is.na()`.

```
# drop NA recordings for the first observation
STAR[1,!is.na(STAR[1,])]
```

```
##      gender ethnicity   birth   star3 read3 math3 lunch3  school3  degree3
## 1122 female      afam 1979 Q3 regular   580   564   free suburban bachelor
##      ladder3 experience3 tethnicity3 system3 schoolid3
## 1122  level1          30        cauc      22        54
```

In general it is not necessary to perform such action when doing regression analysis because `lm()` excludes incomplete cases, i.e. rows with missing data, by default. You should be aware that missing data may imply a small sample size and thus may lead to imprecise estimation and wrong inference which is, however, not not an issue for the study at hand since, as we will see below, sample sizes lie beyond 5000 observations for each regression conducted.

Analysis of the STAR Data

As can be seen from 14.1 there are two treatment groups in each grade, small classes with only 13 to 17 students and regular classes with 22 to 25 students and a teaching aide. Thus, two binary variables, each beeing an indicator for the respective treatment group, are introduced for the differences estimator to capture the treatment effect for both treatment groups seperately. This yields the population regression model

$$Y_i = \beta_0 + \beta_1 SmallClass_i + \beta_2 RegAide_i + u_i, \tag{14.3}$$

with test score Y_i, the small class indicator $SmallClass_i$ and $RegAide_i$, the indicator for a regular class with aide.

We can easily reproduce the results presented in table 13.1 of the book by performing the regression (14.3) for each grade seperately. Note that for each student, the dependent variable is simply the sum of the points scored in the math and reading portion of the test which is why we make use of the function `I()`.

```
# Compute differences Estimates for each grades
fmk <- lm(I(readk + mathk) ~ stark, data = STAR)
fm1 <- lm(I(read1 + math1) ~ star1, data = STAR)
fm2 <- lm(I(read2 + math2) ~ star2, data = STAR)
fm3 <- lm(I(read3 + math3) ~ star3, data = STAR)
```

```
# Obtain coefficient matrix using robust standard errors
coeftest(fmk, vcov = vcovHC(fmk, type= "HC0"))
coeftest(fm1, vcov = vcovHC(fm1, type= "HC0"))
coeftest(fm2, vcov = vcovHC(fm2, type= "HC0"))
coeftest(fm3, vcov = vcovHC(fm3, type= "HC0"))
```

```
##
## t test of coefficients:
##
##                     Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)       918.04289    1.63297 562.1930 < 2.2e-16 ***
## starksmall         13.89899    2.45346   5.6651 1.541e-08 ***
## starkregular+aide   0.31394    2.27039   0.1383      0.89
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## t test of coefficients:
##
##                     Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)       1039.3926     1.7842 582.5691 < 2.2e-16 ***
## star1small          29.7808     2.8305  10.5215 < 2.2e-16 ***
## star1regular+aide   11.9587     2.6514   4.5104 6.588e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## t test of coefficients:
##
##                     Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)       1157.8066     1.8146 638.0403 < 2.2e-16 ***
## star2small          19.3944     2.7110   7.1540 9.428e-13 ***
## star2regular+aide    3.4791     2.5441   1.3675    0.1715
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## t test of coefficients:
##
##                     Estimate Std. Error  t value  Pr(>|t|)
```

```
## (Intercept)       1228.50636    1.67959 731.4322 < 2.2e-16 ***
## star3small          15.58660    2.39544   6.5068 8.303e-11 ***
## star3regular+aide   -0.29094    2.27214  -0.1280    0.8981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We may gather the results and present them in a LaTeX or HTML table using `stargazer()`.

```
# compute robust standard errors for each model and gather them in a list
rob_se_1 <- list(
  sqrt(diag(vcovHC(fmk, type = "HC0"))),
  sqrt(diag(vcovHC(fm1, type = "HC0"))),
  sqrt(diag(vcovHC(fm2, type = "HC0"))),
  sqrt(diag(vcovHC(fm2, type = "HC0")))
)
```

```
library(stargazer)

stargazer(fmk,fm1,fm2,fm3,
  title = "Project STAR: Differences Estimates",
  header = FALSE,
  type = "latex",
  model.numbers = F,
  omit.table.layout = "n",
  digits = 2,
  column.labels = c("K", "1", "2", "3"),
  dep.var.caption  = "Grade",
  dep.var.labels.include = FALSE,
  se = rob_se_1)
```

Table 13.2: Project STAR - Differences Estimates

Grade

K

1

2

3

starksmall

13.90***

(2.45)

starkregular+aide

0.31

(2.27)

star1small

29.78***

(2.83)

star1regular+aide

11.96***

(2.65)
star2small
19.39***
(2.71)
star2regular+aide
3.48
(2.54)
star3small
15.59
star3regular+aide
-0.29
Constant
918.04***
1,039.39***
1,157.81***
1,228.51***
(1.63)
(1.78)
(1.81)
(1.81)
Observations
5,786
6,379
6,049
5,967
R2
0.01
0.02
0.01
0.01
Adjusted R2
0.01
0.02
0.01
0.01
Residual Std. Error

73.49 (df = 5783)

90.50 (df = 6376)

83.69 (df = 6046)

72.91 (df = 5964)

F Statistic

21.26*** (df = 2; 5783)

56.34*** (df = 2; 6376)

28.71*** (df = 2; 6046)

30.25*** (df = 2; 5964)

The estimates suggest that the class size reduction does improve performance of the students. Except for grade 1, the estimates of the coefficient on $SmallClass$ are roughly of the same magnitude (the estimates lie between 13.90 and 19.39 points) and statistically significant at the level of 1%. Furthermore we see that adding a teaching aide to a regular class has little, possibly zero, effect on the performance of the students.

Following the book, we continue by augmenting the population regression (14.3) by different sets of additional regressors for two reasons:

1. If the additional regressors explain some of the observed variation in the dependent variable we obtain more efficient estimates of the coefficients of interest.
2. If the treatment is not received at random due to failures to follows the treatment protocol (see section 13.3 of the book) the estimates obtained using the model specification (14.3) may be biased. Adding additional regressors may solve this problem.

In particular, we consider the following student and teacher characteristics

- *experience* — Teacher's years of experience
- *boy* — Student is a boy (dummy)
- *lunch* — Free lunch eligibility (dummy)
- *black* — Student is African-American (dummy)
- *race* — Student's race is other than black or white (dummy)
- schoolid — School indicator variables

in four different population regression specifications:

$$Y_i = \beta_0 + \beta_1 SmallClass_i + \beta_2 RegAide_i + u_i, \tag{14.4}$$

$$Y_i = \beta_0 + \beta_1 SmallClass_i + \beta_2 RegAide_i + \beta_3 experience_i + u_i, \tag{14.5}$$

$$Y_i = \beta_0 + \beta_1 SmallClass_i + \beta_2 RegAide_i + \beta_3 experience_i + \text{school indicators} + u_i \tag{14.6}$$

$$Y_i = \beta_0 + \beta_1 SmallClass_i + \beta_2 RegAide_i + u_i + \beta_3 experience_i + \beta_4 boy + \beta_5 lunch \tag{14.7}$$

$$+ \beta_6 black + \beta_7 race + \text{schoolid} + u_i \tag{14.8}$$

Beforehand to estimation, we apply some subsetting and some data wrangling. We do this using functions from the packages `dplyr` and `tidyr` which are both part of *tidyverse*, a collection of R packages designed for data science and handling big data sets.[3] The functions `%>%`, `transmute()` and `mutate()` are sufficient for us here. `%>%` allows to chain function calls together. `transmute()` allows to subset the data set by naming the variables to be kept and `mutate()` is convenient for adding new variables based existing ones while preserving the latter.

From looking at the regression models (14.4) to (14.7) we find that variables obviously needed to perform the analysis are `gender`, `ethnicity`, `stark`, `readk`, `mathk`, `lunchk`, `experiencek` and `schoolidk`. After dropping

[3] See the official site for more on *tidyverse* packages.

the remaining variables using `transmute()` we use `mutate()` to add three additional binary variables which are derivatives of existing ones: `black`, `race` and `boy`. They are generated using logical statements within the function `ifelse()`.

```
# load packages 'dplyr' and 'tidyr' for data wrangling functionalities
library(dplyr)
library(tidyr)

# generate subset with kindergarten data
STARK <- STAR %>%
      transmute(
                gender,
                ethnicity,
                stark,
                readk,
                mathk,
                lunchk,
                experiencek,
                schoolidk
                ) %>%
      mutate(
             black = ifelse(ethnicity == "afam", 1, 0),
             race = ifelse(ethnicity == "afam" | ethnicity == "cauc", 1, 0),
             boy = ifelse(gender == "male", 1, 0)
             )
```

```
# Estimate models
gradeK1 <- lm(I(mathk + readk) ~ stark + experiencek, data = STARK)
gradeK2 <- lm(I(mathk + readk) ~ stark + experiencek + schoolidk, data = STARK)
gradeK3 <- lm(I(mathk + readk) ~ stark + experiencek + boy + lunchk + black + race + schoolidk, data = S
```

Since the models `gradeK2` and `gradeK3` include a constant term, there are a total of 79 school indicator dummies. For brevity, we exclude their estimated coefficients in the output produced by `coeftest()` by subsetting the matrices.

```
# obtain robust inference on the significance of coefficients
coeftest(gradeK1, vcov. = vcovHC(gradeK1, type = "HC0"))
```

```
##
## t test of coefficients:
##
##                    Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)       904.72124    2.22157 407.2433 < 2.2e-16 ***
## starksmall         14.00613    2.44620   5.7257 1.082e-08 ***
## starkregular+aide  -0.60058    2.25352  -0.2665    0.7899
## experiencek         1.46903    0.16923   8.6808 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(gradeK2, vcov. = vcovHC(gradeK2, type = "HC0"))[1:4,]
```

```
##                      Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)       925.6748750  7.5981111 121.8296050 0.000000e+00
## starksmall         15.9330822  2.2251817   7.1603511 9.067413e-13
## starkregular+aide   1.2151960  2.0208171   0.6013389 5.476382e-01
## experiencek         0.7431059  0.1685504   4.4088046 1.058577e-05
```

```
coeftest(gradeK3, vcov. = vcovHC(gradeK3, type = "HC0"))[1:7,]
```

```
##                      Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)       937.6831330 14.2647435  65.7343143 0.000000e+00
## starksmall         15.8900507  2.1389984   7.4287344 1.259182e-13
## starkregular+aide   1.7869378  1.9467305   0.9179174 3.587012e-01
## experiencek         0.6627251  0.1646838   4.0242278 5.790802e-05
## boy               -12.0905123  1.6600732  -7.2831199 3.707612e-13
## lunchkfree        -34.7033021  1.9721158 -17.5969896 1.586414e-67
## black             -25.4305130  3.4724200  -7.3235707 2.752262e-13
```

We use `stargazer()` to gather all relevant information in a structured table.

```
# compute robust standard errors for each model and gather them in a list
rob_se_2 <- list(
  sqrt(diag(vcovHC(fmk, type = "HC0"))),
  sqrt(diag(vcovHC(gradeK1, type = "HC0"))),
  sqrt(diag(vcovHC(gradeK2, type = "HC0"))),
  sqrt(diag(vcovHC(gradeK3, type = "HC0")))
)
```

```
stargazer(fmk,fm1,fm2,fm3,
  title = "Table 13.3: Project STAR - Differences Estimates with Additional Regressors for Kindergarten
  header = FALSE,
  type = "latex",
  model.numbers = F,
  omit.table.layout = "n",
  digits = 2,
  column.labels = c("(1)", "(2)", "(3)", "(4)"),
  dep.var.caption  = "Test Score in Kindergarten",
  dep.var.labels.include = FALSE,
  se = rob_se_2)
```

Table 13.3: Project STAR - Differences Estimates with Additional Regressors for Kindergarten

Test Score in Kindergarten

(1)

(2)

(3)

(4)

starksmall

13.90***

14.01***

15.93***

15.89***

(2.45)

(2.45)

(2.23)

(2.14)

starkregular+aide

0.31

-0.60

1.22

1.79

(2.27)

(2.25)

(2.02)

(1.95)

experiencek

1.47***

0.74***

0.66***

(0.17)

(0.17)

(0.16)

boy

-12.09***

(1.66)

lunchkfree

-34.70***

(1.97)

black

-25.43***

(3.47)

race

8.50

(12.43)

Constant

918.04***

904.72***

925.67***

937.68***

(1.63)

(2.22)

(7.60)

(14.26)

School indicators?

no

no

yes

yes

Observations

5,786

5,766

5,766

5,748

R2

0.01

0.02

0.23

0.29

Adjusted R2

0.01

0.02

0.22

0.28

Residual Std. Error

73.49 (df = 5783)

73.08 (df = 5762)

65.08 (df = 5684)

62.66 (df = 5662)

F Statistic

21.26*** (df = 2; 5783)

39.86*** (df = 3; 5762)

21.41*** (df = 81; 5684)

27.36*** (df = 85; 5662)

While the results in column (1) of table Table 13.3 are just a repetition of the outcomes obtained for (14.3) before, columns (2) to (4) reveal that adding student characteristics and school fixed effects does not lead to significantly different estimates of the treatment effects. This result makes it more plausible that the estimates of the effects obainted using model (14.3) do not suffer from failure of random assignment. Note that there is some decrease in the standard errors and some increase in $\overline{R}^2$, implying that the estimates are more precise.

Because teachers where randomly assigned to classes, inclusion of school fixed effect allows us to estimate the effect of a teacher's experience on test scores of students in kindergarten. For regression (3) we find that the average effect of 10 years experience on test scores is predicted to be $10 * 0.74 = 7.4$ points. It is important to be aware of the fact that the other estimates on student characteristics used in regression (4) *do not* have causal interpretation due to nonrandom assignment.[4]

To answer the question whether the estimated effect presented in table 13.2 are large or small in a practical sense, we translate the predicted changes in test scores to units of standard deviation in order to allow for a comparison (see Section 9.4).

```
# compute sample standard deviations of test scores
SSD <- c(
        "K" = sd(na.omit(STAR$readk + STAR$mathk)),
        "1" = sd(na.omit(STAR$read1 + STAR$math1)),
        "2" = sd(na.omit(STAR$read2 + STAR$math2)),
        "3" = sd(na.omit(STAR$read3 + STAR$math3))
)

# translate effects of small classes to standard deviations
Small <- c(
           "K" = as.numeric(coef(fmk)[2]/SSD[1]),
           "1" = as.numeric(coef(fm1)[2]/SSD[2]),
           "2" = as.numeric(coef(fm2)[2]/SSD[3]),
           "3" = as.numeric(coef(fm3)[2]/SSD[4])
)

# adjust standard errors
SmallSE <- c(
             "K" = as.numeric(rob_se_1[[1]][2]/SSD[1]),
             "1" = as.numeric(rob_se_1[[2]][2]/SSD[2]),
             "2" = as.numeric(rob_se_1[[3]][2]/SSD[3]),
             "3" = as.numeric(rob_se_1[[4]][2]/SSD[4])
)

# translate effects of regular classes with aide to standard deviations
RegAide<- c(
            "K" = as.numeric(coef(fmk)[3]/SSD[1]),
            "1" = as.numeric(coef(fm1)[3]/SSD[2]),
            "2" = as.numeric(coef(fm2)[3]/SSD[3]),
            "3" = as.numeric(coef(fm3)[3]/SSD[4])
)

# adjust standard errors
RegAideSE <- c(
               "K" = as.numeric(rob_se_1[[1]][3]/SSD[1]),
               "1" = as.numeric(rob_se_1[[2]][3]/SSD[2]),
               "2" = as.numeric(rob_se_1[[3]][3]/SSD[3]),
               "3" = as.numeric(rob_se_1[[4]][3]/SSD[4])
)

# Gather the results in a data.frame and round
df <- t(
  round(
```

[4] See Chapter 13.3 of the book for a detailed discussion.

```
    data.frame(
      Small, SmallSE,RegAide, RegAideSE, SSD
      ),
    digits =  2
    )
  )
```

It is fairly easy to turn the data.frame `df` into a LaTeX table.

```
# generate simply LaTeX table using stargazer
stargazer(df,
          title = "Table 13.4: Estimated Class Size Effects (in Units of Standard Deviations)",
          type = "html",
          summary = FALSE)
```

Table 13.4: Estimated Class Size Effects (in Units of Standard Deviations)

K

1

2

3

Small

0.190

0.330

0.230

0.210

SmallSE

0.030

0.030

0.030

0.040

RegAide

0

0.130

0.040

0

RegAideSE

0.030

0.030

0.030

0.030

SSD

73.750

91.280

84.080

73.270

On the one hand, we find that the estimated effects of a small classes is largest for grade 1. As pointed out in the book, this is probably because student in the control group for grade 1 did poorly on the test for some unknown reason or simply due to random variation. On the other hand, the difference between the estimated effect of beeing in a small class and beeing in a regular classes with an aide is roughly 0.2 standard deviations for all grades. This leads to the conclusion that the effect of beeing in a regular sized class with an aide is zero and the effect of beeing in a small class is roughly the same for all grades.

The remainder of Chapter 13.3 in the book discusses to what extent these experimental estimates are comparable with observational estimates obtained using data on school districts in California and Massachusetts in Chapter 9. It turns out that the estimates are indeed very similar. Please refer to the aforementioned section in the book for the a discussion.

14.4 Quasi Experiments

In quasi-experiments, "as if" randomness is exploited to use methods similar to those that have been discussed in the previous Chapter. There are two types of quasi-experiments:[5]

1. Random variations in individual circumstances allows to view the treatment "as if" it was randomly determined.
2. The treatment is only partially determined by "as if" random variation.

The former allows to estimate the effect using either model (14.2), i.e. the *difference estimator with additional regressors*, or, if there is doubt that the "as if" randomness does not entirely ensure that there are no systematic differences between control and treatment group, using the *differences-in-differences* (DID) estimator. In the latter case, an IV approach for estimation of a model like (14.2) which uses the source of "as if" randomness in treatment assignment as the instrument may be applied.

Some more advanced techniques that are helpful in settings where the treatment assignment is (partially) determined by a threshold in a so-called running variable are *sharp regression discontinuity design* (RDD) and *fuzzy regression discontinuity design* (FRDD).

We will briefly review these techniques and, since the book does not provide any empirical examples in this section, we will use our own simulated data in a minimal example to discuss how DiD, RDD and FRDD can be applied in R.

The Differences-in-Differences Estimator

In quasi-experiments is often the case that the source of "as if" randomness in treatment assignment cannot entirely prevent systematic differences between control and treatment group. This problem was encountered by Card and Krüger (1994) who used geography as the "as if" random treatment assignment to study the effect on employment of workers in the fast-food restaurants caused by an increase in the state minimum wage in New Jersey in the year of 1992. Their idea was to use the fact that the increase in minimum wage applied to employees in New Jersey (treatment group) but not to those living in the neighbouring Pennsylvania (control group).

It is quite conceivable that such a wage hike is not correlated with other determinants of employment. However, there still might by some state-specific differences an thus differences between control and treatment group. This would render the *differences estimator* biased and inconsistent. Card and Krüger (1994) solved

[5] See chapter 13.4 of the book for some example studies that are based on quasi-experiments.

this by using a DID estimator: they collected data in February 1992 (before the treatment) and November 1992 (after the treatment) for the same restaurants and estimated the effect of the wage hike by analyzing differences in the differences in employment for New Jersey and Pensilvania before and after the increase.[6] The DID estimator is

$$\widehat{\beta}_1^{\text{diffs-in-diffs}} = (\overline{Y}^{\text{treatment,after}} - \overline{Y}^{\text{treatment,before}}) - (\overline{Y}^{\text{control,after}} - \overline{Y}^{\text{control,before}}) \tag{14.9}$$

$$= \Delta\overline{Y}^{\text{treatment}} - \Delta\overline{Y}^{\text{control}} \tag{14.10}$$

with

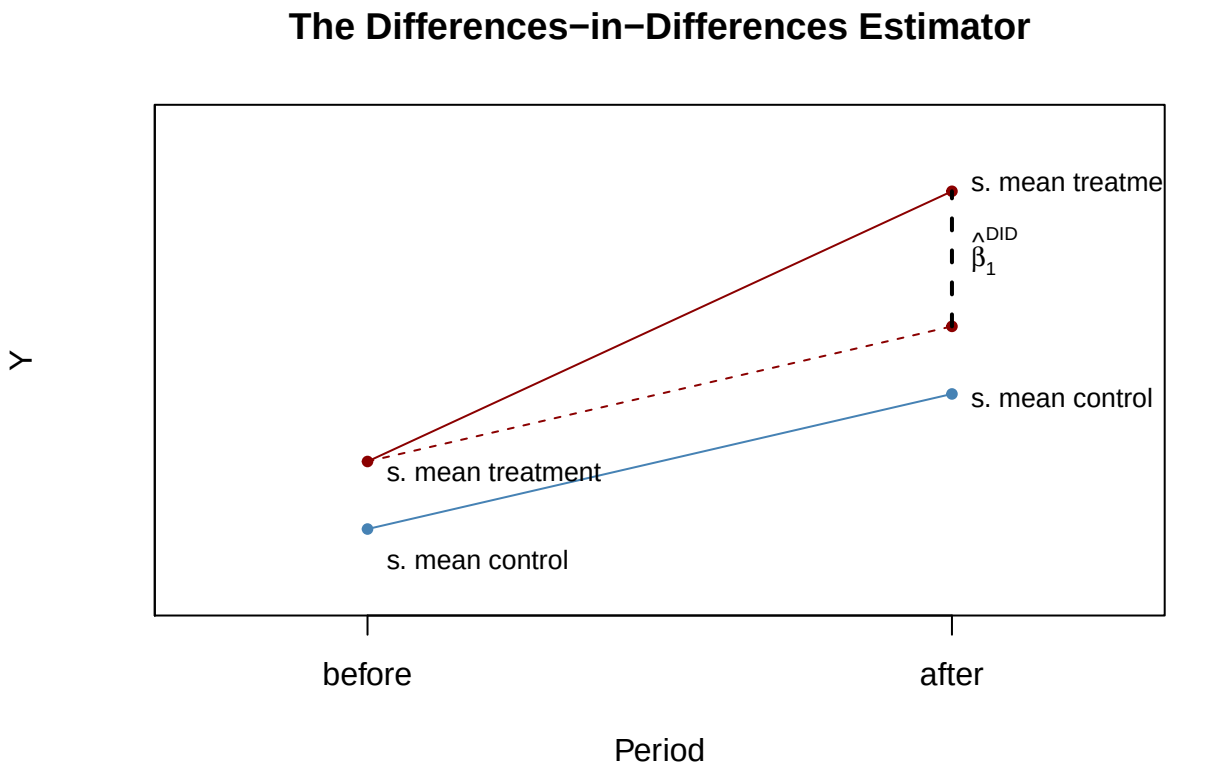
- $\overline{Y}^{\text{treatment,before}}$ - the sample average in the treatment group, before the treatment,
- $\overline{Y}^{\text{treatment,after}}$ - the sample average in the treatment group, after the treatment,
- $\overline{Y}^{\text{treatment,before}}$ - the sample average in the control group, before the treatment,
- $\overline{Y}^{\text{treatment,after}}$ - the sample average in the control group, after the treatment.

We may use R to reproduce figure 13.1 of the book.

```
# initialize plot and add control group
plot(c(0,1), c(6,8),
     type = "p",
     ylim = c(5,12),
     xlim = c(-0.3,1.3),
     main = "The Differences-in-Differences Estimator",
     xlab = "Period",
     ylab = "Y",
     col = "steelblue",
     pch = 20,
     xaxt = "n",
     yaxt = "n"
)
axis(1, at = c(0,1), labels = c("before","after"))
axis(2, at = c(0,13))

# add treatment group
points(c(0,1,1), c(7,9,11),
       col = "darkred",
       pch = 20
)

# add line segments
lines(c(0,1),c(7,11), col = "darkred")
lines(c(0,1),c(6,8), col = "steelblue")
lines(c(0,1),c(7,9), col = "darkred", lty = 2)
lines(c(1,1),c(9,11), col = "black", lty = 2, lwd = 2)

# add annotations
text(1, 10, expression(hat(beta)[1]^{DID}), cex = 0.8, pos = 4)
text(0, 5.5, "s. mean control", cex = 0.8 , pos = 4)
text(0, 6.8, "s. mean treatment", cex = 0.8 , pos = 4)
```

[6] See also the box *What is the Effect on Employment of the Minimum Wage?* in Chapter 13.4 of the book.

```
text(1, 7.9, "s. mean control", cex = 0.8 , pos = 4)
text(1, 11.1, "s. mean treatment", cex = 0.8 , pos = 4)
```



The DID estimator (14.10) can als be written in regression notation: $\widehat{\beta}_1^{\text{DID}}$ is the OLS estimator of β_1 in the model

$$\Delta Y_i = \beta_0 + \beta_1 X_i + u_i \tag{14.11}$$

where ΔY_i denotes the difference in pre- and post-treatment outcomes of individual i and X_i is the treatment indicator.

Adding additional regressors that measure pre-treatment characteristics to (14.11) we obtain

$$\Delta Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_{1i} + \cdots + \beta_{1+r} W_{ri} + u_i, \tag{14.12}$$

the *difference-in-differences estimator* with additional regressors. The additional regressors may lead to a more precise estimate of β_1.

We will keep things simple and focus on estimation of the treatment effect using DID in the most simplest case, that is a control and a treatment group observed for two time periods — one before and one after the treatment. In particular, we will see that there are three different ways to proceed.

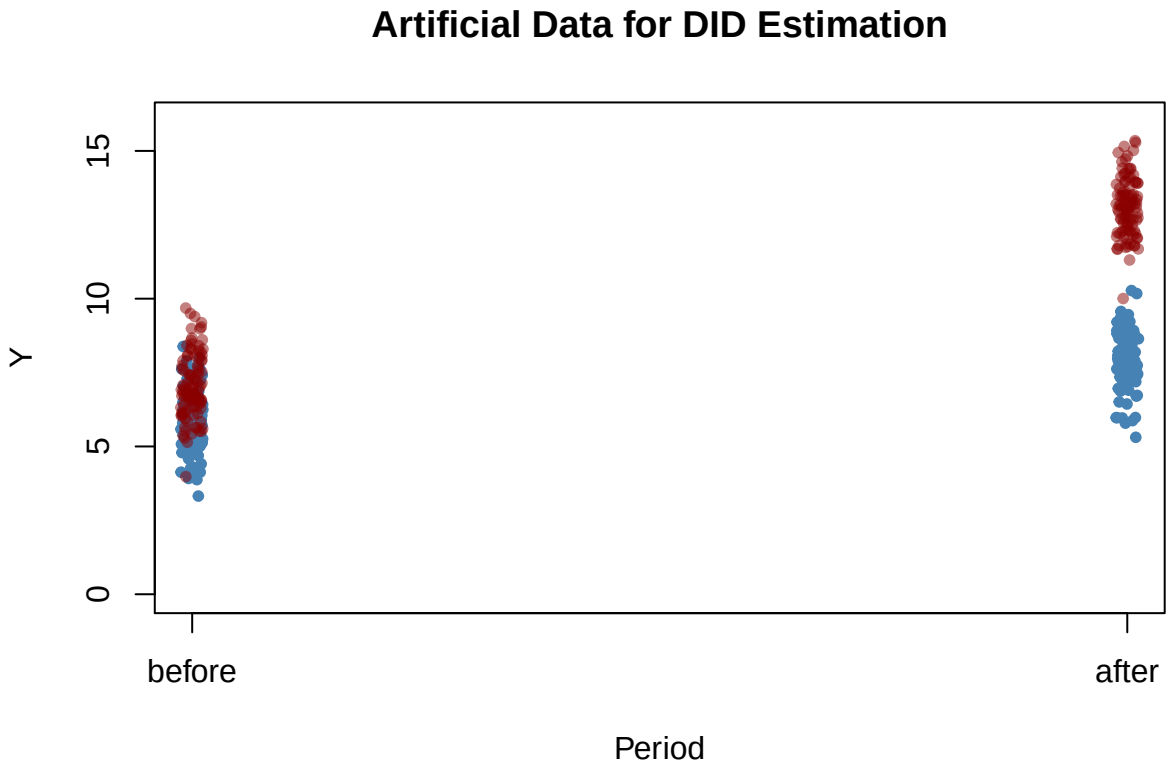
First, we simulate pre- and post-treatment data using R.

```
# sample size
n <- 200

# treatment effect
```

```
TEffect <- 4

# treatment dummy
TDummy <- c(rep(0, n/2), rep(1, n/2))

# simulate pre- and post-treatment values of the dep. variable
y_pre <- 7 + rnorm(n)
y_pre[1:n/2] <- y_pre[1:n/2] - 1
y_post <- 7 + 2 + TEffect * TDummy + rnorm(n)
y_post[1:n/2] <- y_post[1:n/2] - 1
```

We continue by plotting the data. The function `jitter()` is used to add some artificial dispersion in the horizontal component of the points so that there is less overplotting. `alpha()` from the package `scales` allows to adjust the opacity of colors.

```
library(scales)

pre <- rep(0, length(y_pre[TDummy==0]))
post <- rep(1, length(y_pre[TDummy==0]))

# plot control group in t=1
plot(jitter(pre, 0.6),
     y_pre[TDummy==0],
     ylim = c(0,16),
     col = "steelblue",
     pch = 20,
     xlim = c(0,1),
     ylab = "Y",
     xlab = "Period",
     xaxt = "n",
     main = "Artificial Data for DID Estimation"
     )
axis(1, at = c(0,1), labels = c("before","after"))

# add treatment group in t=1
points(jitter(pre, 0.6),
       y_pre[TDummy==1],
       col = alpha("darkred", 0.5),
       pch = 20
       )

# add control group in t=2
points(jitter(post, 0.6),
       y_post[TDummy==0],
       col = "steelblue",
       pch = 20
       )

# add treatment group in t=2
points(jitter(post, 0.6),
       y_post[TDummy==1],
       col = alpha("darkred", 0.5),
       pch = 20
       )
```



Notice that both observations from control and treatment group have a higher sample mean after the treatment but that the increase is stronger for the treatment group. Using DID we may estimate how much of that difference in the difference between groups is due to the treatment.

It is straightforward to compute the DID estimate in the fashion of (14.10).

```
# Compute the DID estimator for the treatment effect 'by hand'
mean(y_post[TDummy==1]) - mean(y_pre[TDummy==1]) -
(mean(y_post[TDummy==0]) - mean(y_pre[TDummy==0]))
```

```
## [1] 3.960268
```

Notice that the estimate is close to 4, the value chosen as the treatment effect `TEffect` above. Since (14.11) is a simple linear model, we may perform OLS estimation of this regression specification using `lm()`.

```
# Compute the DiD estimator using a linear model
lm(I(y_post - y_pre) ~ TDummy)
```

```
##
## Call:
## lm(formula = I(y_post - y_pre) ~ TDummy)
##
## Coefficients:
## (Intercept)       TDummy
##       2.104        3.960
```

We find that the estimates conincide. Furthermore, one can show that the DID estimate obained by estimating specification (14.11) OLS is the same as the OLS estimate of β_{TE} in

$$Y_i = \beta_0 + \beta_1 D_i + \beta_2 Period_i + \beta_{TE}(Period_i \times D_i) \tag{14.13}$$

where D_i is the binary treatment indicator, $Period_i$ is a binary indicator for the after treatment period and the $Period_i \times D_i$ is the interaction of both.

As for (14.11), estimation of (14.13) using R is straightforward. See Chapter 8 for a discussion of interaction terms in regression models.

```
# prepare Data for DID regression using interaction term
d <- data.frame("Y" = c(y_pre,y_post),
                "Treatment" = TDummy,
                "Period" = c(rep("1", n), rep("2", n))
                )

# estimate the model
lm(Y ~ Treatment * Period, data = d)
```

```
##
## Call:
## lm(formula = Y ~ Treatment * Period, data = d)
##
## Coefficients:
##       (Intercept)          Treatment            Period2
##             5.858              1.197              2.104
## Treatment:Period2
##             3.960
```

As expected, the estimate of the coefficient on the interaction of the treatment dummy and the time dummy coincides with the estimates obtained using the approach (14.10) and OLS estimation of (14.11).

Regression Discontinuity Estimators

Consider the model

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_i + u_i \tag{14.14}$$

and let

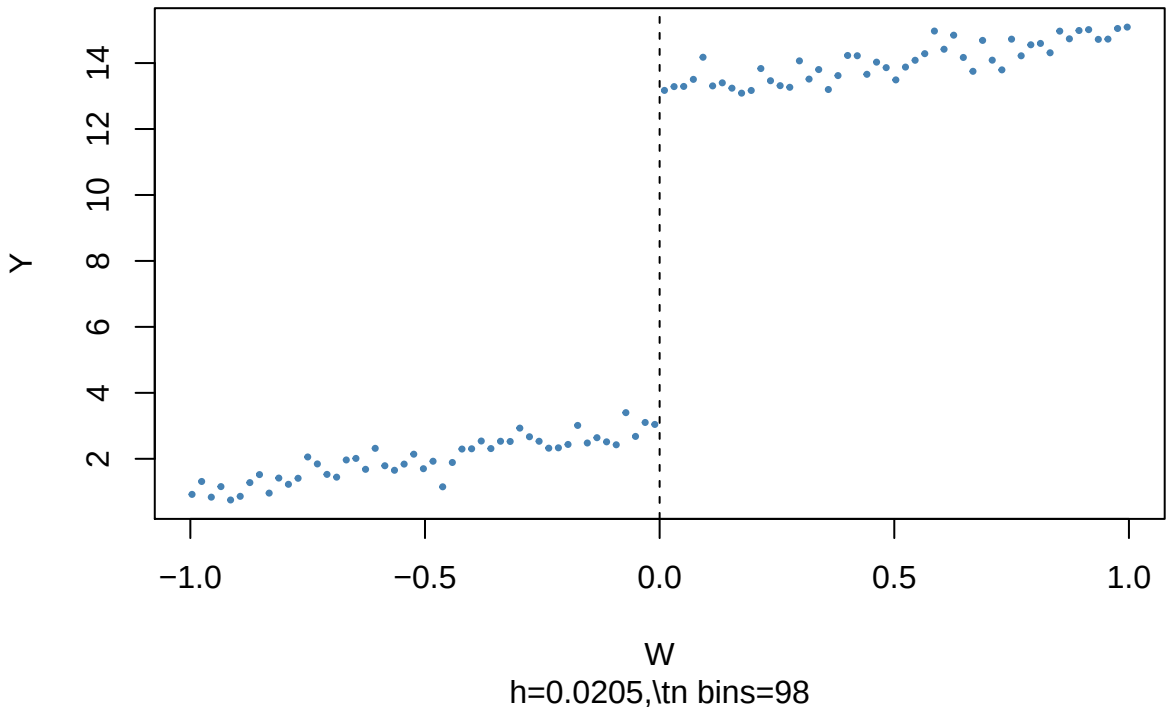
$$X_i = \begin{cases} 1, & W_i \geq c \\ 0, & W_i < c \end{cases}$$

so that the receipt of treatment, X_i, is determined by some threshold c of a continuous variable W_i, the so called running variable. The idea of *regression discontinuity design* is to use observations with a W_i close to c for estimation of β_1, the average treatment effect for individuals with $W_i = c$ which is assumed to be a good approximation to the treatment effect in the population. (14.14) is called a *sharp regression discontinuity design* because treatment assignment is deterministic and discontinuous at the cutoff: all observations with $W_i < c$ do not receive treatment and all observations where $W_i \geq c$ are treated.

The subsequent code chunks show how to estimate a linear SRDD using R and how to produce plots in the way of Figure 13.2 of the book.

```
# generate sample data
W <- runif(1000, -1, 1)
y <- 3 + 2 * W + 10 * (W>=0) + rnorm(1000)
```

```
# load the package 'rddtools'
library(rddtools)

# construct rdd_data
data <- rdd_data(y, W, cutpoint = 0)

# plot the sample data
plot(data,
     col = "steelblue",
     cex = 0.35,
     xlab = "W",
     ylab = "Y"
     )
```



The argument `nbins` sets the number of bins the running variable is devided in. The dots represent bin averages of the outcome variable.

We may use the function `rdd_reg_lm()` to estimate the treatment effect using model (14.14) for the artificial data generated above. By choosing `slope = "same"` we restrict the slopes of the estimated regression function to be the same on both sides of the jump.

```
# estimate the sharp RDD model
rdd_mod <- rdd_reg_lm(rdd_object = data,
                      slope = "same")
summary(rdd_mod)
```

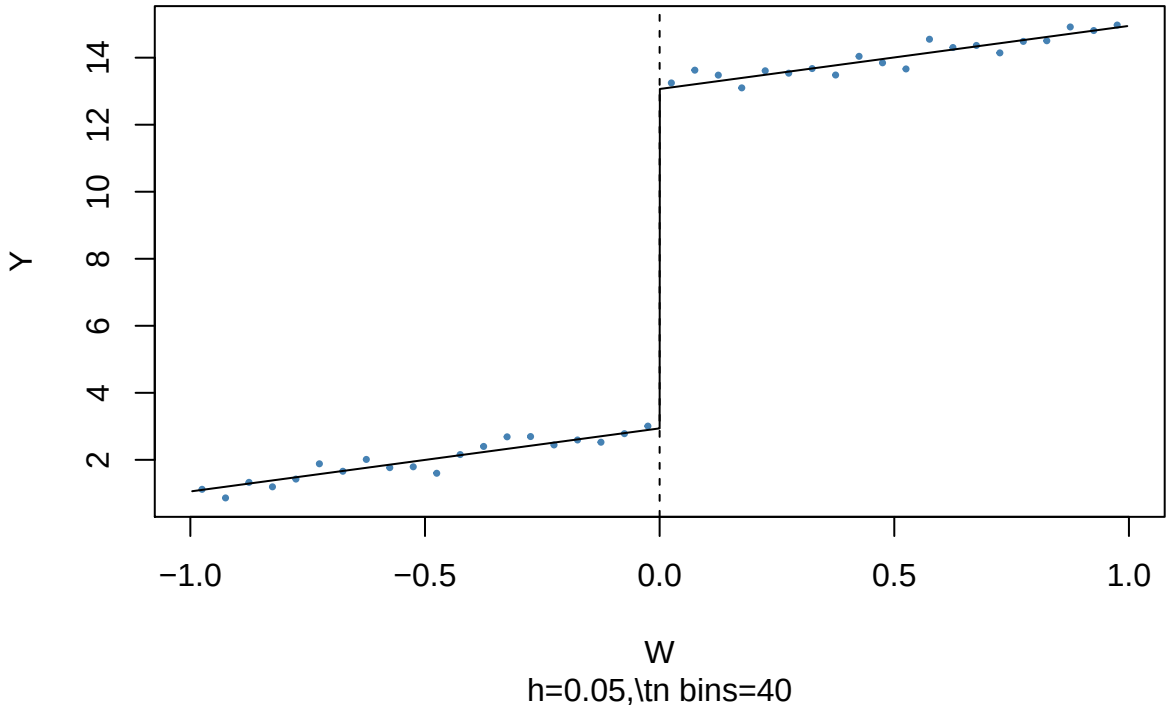
```
##
## Call:
## lm(formula = y ~ ., data = dat_step1, weights = weights)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.2361 -0.6779 -0.0039  0.7113  3.0096
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.93889    0.07082   41.50   <2e-16 ***
## D           10.12692    0.12631   80.18   <2e-16 ***
## x            1.88249    0.11074   17.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 997 degrees of freedom
## Multiple R-squared:  0.972,  Adjusted R-squared:  0.972
## F-statistic: 1.732e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

The coefficient estimate of interest is labeled D. Note that the estimate is very close to the treatment effect chosen in the DGP above.

It is easy to visualize the result: simply call `plot()` on the estimated model object.

```
plot(rdd_mod,
     cex = 0.35,
     col = "steelblue",
     xlab = "W",
     ylab = "Y")
```



Note that, as above, the dots represent averages of binned observations.

So far we have assumed that crossing of the threshold determines receipt of treatment so that the jump of the population regression functions at the threshold can be regarded as the causal effect of the treatment.

When crossing of the threshold c is not the only cause for receipt of the treatment and instead other, treatment is not a deterministic function of W_i. Instead, it is useful to think of the c as a threshold where the *probability of receiving* the treatment jumps.

This jump may be due to unobservable variables that have impact on the *probability* of being treated. Thus, X_i in (14.14) will be correlated with the error u_i and it becomes more difficult to consistently estimate the treatment effect. In this setting, using a *fuzzy regression discontinuity design* which is based an IV approach may be a remedy: take the binary variable Z_i as an indicator for crossing of the threshold,

$$Z_i = \begin{cases} 1, & W_i \geq c \\ 0, & W_i < c, \end{cases}$$

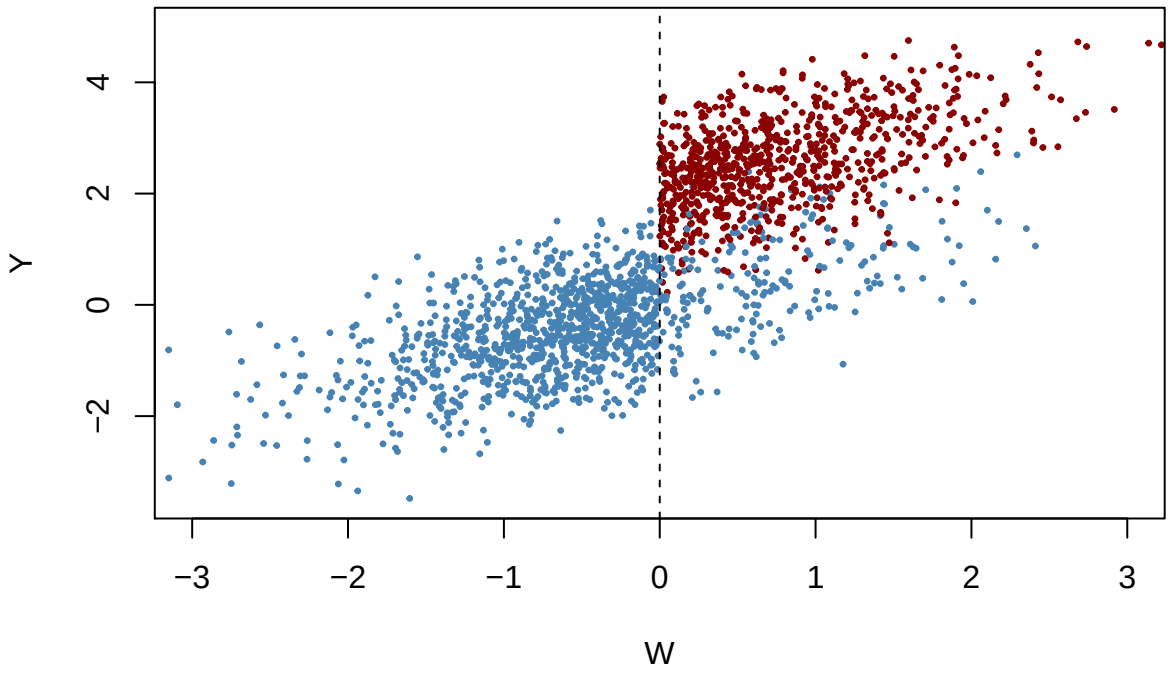
and that Z_i relates to Y_i only through the treatment indicator X_i. Then Z_i and u_i are uncorrelated but Z_i influences receipt of treatment so it is correlated with X_i. Thus, Z_i is a valid instrument for X_i and (14.14) can be estimated using TSLS.

The following code chunk generates sample data where observations with a value of the running variable W_i below the cutoff $c = 0$ do not receive treatment and observations with $W_i \geq 0$ do receive treatment with a probability of 80% so that treatment status is only partially determined by the running variable and the cutoff. Treatment leads to an increase in Y by 2 units. Observations with $W_i \geq 0$ that do not receive treatment are called *no-shows*: think of an individual that was assigned to receive the treatment but somehow managed to avoid it.

```
library(MASS)

# generate sample data
mu <- c(0,0)
sigma <- matrix(c(1, 0.7, 0.7, 1), ncol = 2)

set.seed(1234)
d <- as.data.frame(mvrnorm(2000, mu, sigma))
colnames(d) <- c("W", "Y")

# Introduce fuzziness
d$treatProb <- ifelse(d$W < 0, 0, 0.8)

fuzz <- sapply(d$treatProb, function(x) rbinom(1, 1, prob = x))

# Treatment effect
d$Y <- d$Y + fuzz * 2
```

We plot all observations and use blue color to mark individuals that did not receive the treatment and use red for those that received the treatment.

```
# Colored plot of tretment and control group
plot(d$W, d$Y,
     col = c("steelblue","darkred")[factor(fuzz)],
     pch= 20,
     cex = 0.5,
     xlim = c(-3,3),
     ylim = c(-3.5,5),
     xlab = "W",
     ylab = "Y"
)

# add dashed vertical line at cutoff
abline(v = 0, lty = 2)
```



Obviously, receipt of treatment is no longer a deterministic function of the running variable W. Some observations with $W \geq 0$ *did not* receive the treatment. We may estimate a FRDD by additionally setting `treatProb` as the assignment variable `z` in `rdd_data()`. Then `rdd_reg_lm()` applies the folling TSLS procedure: treatment is predicted using W_i and the cutoff dummy Z_i, the instrumental variable, in the first stage regression. The fitted values from the first stage regression are used to obtain an unbiased estimate of the treatment effect using the second stage where the outcome Y is regressed on the fitted values and the running variable W_i.

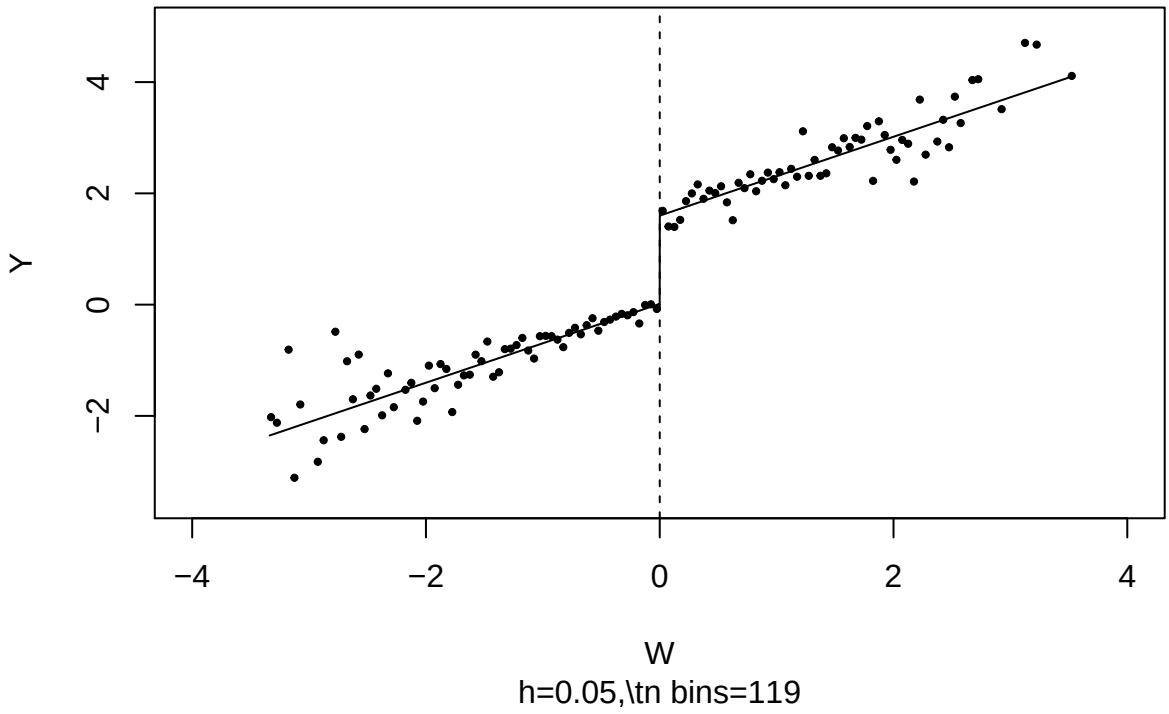
```
# Estimate the Fuzzy RDD
data <- rdd_data(d$Y, d$W,
                 cutpoint = 0,
                 z = d$treatProb)

frdd_mod <- rdd_reg_lm(rdd_object = data,
                       slope = "same")
frdd_mod
```

```
## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  same
##  Number of obs: 2000 (left: 999, right: 1001)
##
##  Coefficient:
##   Estimate Std. Error t value  Pr(>|t|)
## D 1.981297   0.084696  23.393 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimate is close to 2, the population treatment effect. We may call `plot()` on the model object to obtain a graph consisting of binned data and the estimated regression function.

```
plot(frdd_mod,
     cex = 0.5,
     lwd = 0.4,
```

```
     xlim = c(-4,4),
     ylim = c(-3.5,5),
     xlab = "W",
     ylab = "Y"
     )
```



What if we would use a SRDD instead, therby ignoring the fact that treatment is not perfectly determined by the cutoff in W?

```
# Estimate Sharp RDD
data <- rdd_data(d$Y, d$W,
                 cutpoint = 0
)
srdd_mod <- rdd_reg_lm(rdd_object = data,
                       slope = "same")
srdd_mod
```

```
## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  same
##  Number of obs: 2000 (left: 999, right: 1001)
##
##  Coefficient:
##   Estimate Std. Error t value  Pr(>|t|)
## D 1.585038   0.067756  23.393 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimate obtained using a SRDD shows a substantial downward bias. In fact this procedure is inconsistent for the true causal effect so enlarging the sample does not alleviate the bias.

Hereinafter the book continues with a discussion of potential problems with quasi-experiments. As for all empirical studies, these potential problems relate to internal and external validity. This part is followed by

a thechnical discussion of treatment effect estimation when the causal effect of treatment is heterogeneous in the population. We encourage you to work on these sections on your own.

Summary

In this chapter we have introduced the concept of causal effect in randomized controlled experiments and quasi-experiments where variations in circumstances or accidents of nature are treated as sources of "as if" random assignment to treatment. We have also discussed methods that allow for unbiased and consistent estimation of these effects in both settings. This included the *differences estimator*, the *differences-in-differences estimator* as well as *sharp* and *fuzzy regression discontinuity design* estimators. It was shown how to apply these estimation techniques in R.

In an empirical application we have shown how to replicate the results of the analysis of the STAR data presented in Chapter 13.3 of the book using R. This study uses a randomized controlled experiment to assess whether smaller classes improve students' performance on standardized tests. We have stressed that, being related to a randomized controlled experiment, the data produced by this study are fundamentally different than those used in the cross-section studies Chapters 4 to 8 are concerned with and thus motivated usage of a *differences estimator*.

Chapter 13.4 demonstrated how estimates of treatment effects can be obtained when the design of the study is a quasi-experiment that allows for *differences-in-differences* or *regression discontinuity design* estimators. In particular, we have introduced core functions of the package `rddtools` that are convenient for estimation as well as graphical analysis of the research design before and after estimating a discontinuity design.

Chapter 15

Introduction to Time Series Regression and Forecasting

```
library(dynlm)
library(stargazer)
library(scales)
library(readxl)
```

Time series data is data that is collected for a single entitity over time. This is fundamentally different from cross-section data which is data on *multiple* entities at the same point in time. Time series data allows estimation of the effect on Y of a change in X *over time*. This is what econometricians call a *dynamic causal effect*. Let us go back to the application to cigarette consumption of Chapter 12 where we where interest in estimating the effect on cigarette demand of a price increase caused by a raise of the general sales tax. One might use time series data to assess the effect of causal effect of a tax hike on smoking both, initially and in subsequent periods.

Another application of time series data is forecasting. For example, weather services use time series models to predict tomorrow's average temperatur using todays average termperature and average termperatures of the past, or, to motivate an economic example, central banks are interested in forecasting next month's unemployment rates.

The remainder of the book deals with the econometric techniques for the analysis of time series data and application of the latter to problems of forecasting and estimation of dynamic causal effects. This section covers the basic concepts presented in Chapter 14 of the book, explains how to visualize time series data and demonstrates how to estimate simple autoregressive models where the regressors are past values of the dependent variable or other variables. In this context we will also discuss the concept of stationarity, an important property which has far-reaching consequences since it determines whether the past of a series has any power in explaining the series' future.

Most empirical applications in this chapter are concerned with forecasting and use data on U.S. macroeconomic indicators or financial time series like Gross Domestic Product (GDP), the unemployment rate or excess stock returns.

15.1 Using Regression Models for Forecasting

What is the difference between estimating models for assessment of causal effects and forecasting? Consider again the simple example of estimating the casual effect on test scores of the student-teacher ratio from Chapter 4.

```
library(AER)
data(CASchools)
CASchools$STR <- CASchools$students/CASchools$teachers
CASchools$score <- (CASchools$read + CASchools$math)/2

mod <- lm(score ~ STR, data = CASchools)
mod
```

```
##
## Call:
## lm(formula = score ~ STR, data = CASchools)
##
## Coefficients:
## (Intercept)          STR
##      698.93        -2.28
```

As has been stressed in Chapter 6, the estimate of the coefficient on the student-teacher ratio does not have causal interpretation due to omitted variable bias. However, in terms deciding which schooling to send her child to, it might nevertheless be appealig for a parent to use `mod` for forecasting test scores in schooling districts where no public data about on scores are available.

As an example, assume that the average class in a district has 25 students. There is no such thing like a perfect forecast but the following one-liner might be helpful for the parent to decide.

```
predict(mod, newdata = data.frame("STR" = 25))
```

```
##        1
## 641.9377
```

In a time series context, the parent could use data on present and past years test scores to predict to forecast next years test scores — a typical application for an autoregressive model.

15.2 Time Series Data and Serial Correlation

GDP measures the productivity of an economy. It is commonly defined as the value of goods services produced over a given time period. The data set us_macro_quarterly.xlsx is provided by the authors and can be downloaded here. It provides data on quarterly data on US real (i.e. inflation adjusted) GDP from years 1947 to 2004.

As before, a good point to start with in pre-estimation is plotting the data. The package `quantmod` provides some very convenient functions for plotting and computing with time series data. We also load the package `readxl` to read the data into R.

```
# attach the packages
library(quantmod)
```

We begin by importing the data set.

```
# load US macroeconomic data
USMacroSWQ <- read_xlsx("Data/us_macro_quarterly.xlsx",
                        sheet = 1,
                        col_types = c("text", rep("numeric", 9))
                       )

# format date column
USMacroSWQ$X__1 <- as.yearqtr(USMacroSWQ$X__1, format = "%Y:0%q")
```

```
# adjust column names
colnames(USMacroSWQ) <- c("Date", "GDPC96", "JAPAN_IP", "PCECTPI", "GS10", "GS1", "TB3MS", "UNRATE", "E
```

When dealing with time series data in R it is preferable to work with time-series objects that keep track of the frequency of the data and are extensible. In what follows we will use xts objects, see `?xts`. Since the data In `USMacroSWQ` are in quarterly frequence we convert the first column to `yearqtr` format before generating the xts object `GDP`.

The function `Delt()` from the package `Quantmod` computes growth rates. Note that, since the data is quarterly, we anualize the quarterly changes to obtain anual growth rates as

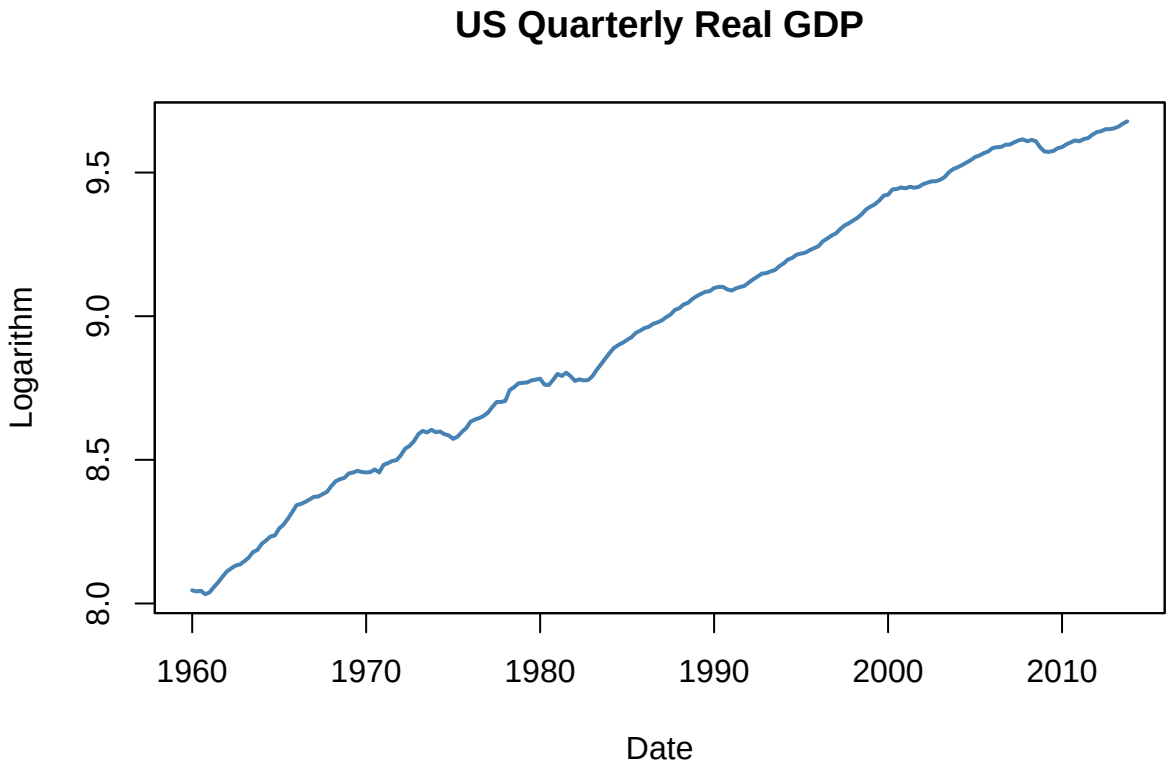
$$Rate_{annual} = (1 + Rate_{quarterly})^4 - 1$$

.

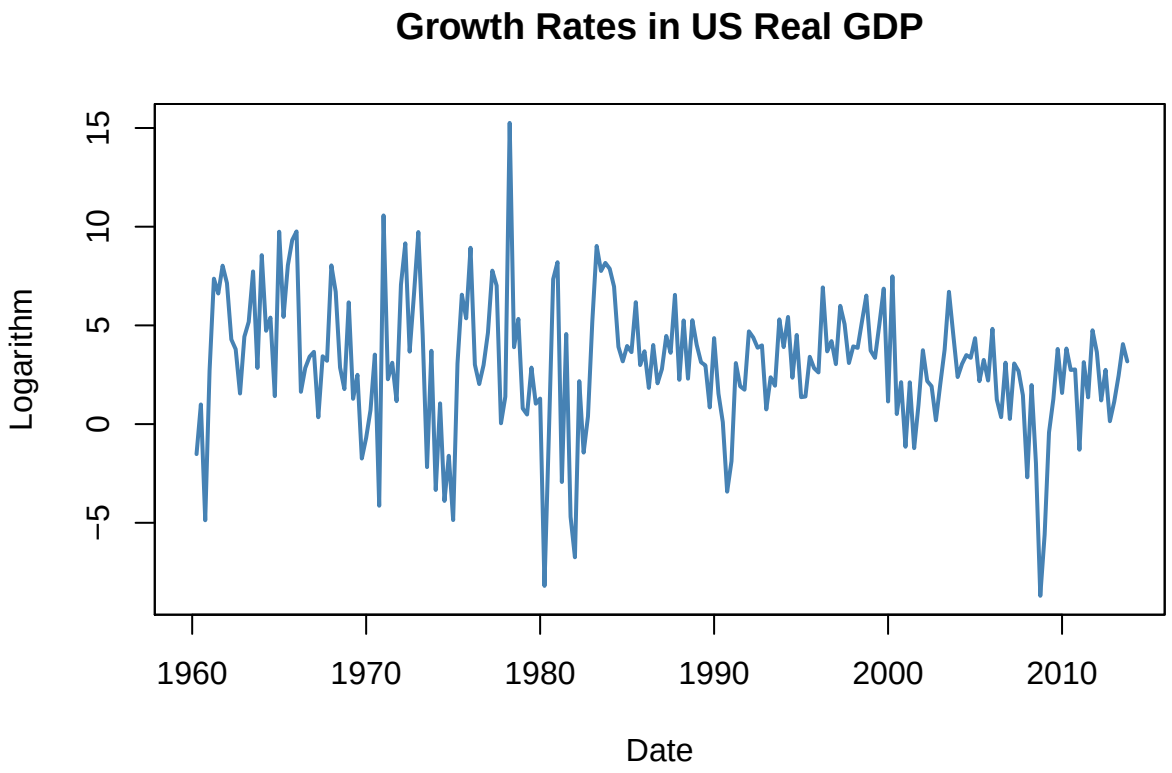
```
# GDP series as xts object
GDP <- xts(USMacroSWQ$GDPC96, USMacroSWQ$Date)["1960::2013"]

# GDP growth series as xts object
GDPGrowth <- xts(400 * log(GDP/lag(GDP)))
```

We use `chart.TimeSeries()` from the package `PerformanceAnalytics` to generate plots of `GDP` and `GDPGrowth`. The following code chunks reproduce figure 14.1 of the book.

```
# reproduce figure 14.1 (a) of the book
plot(log(as.zoo(GDP)),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "US Quarterly Real GDP"
     )
```



```
# reproduce figure 14.1 (b) of the book
plot(as.zoo(GDPGrowth),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "Growth Rates in US Real GDP"
     )
```



Notation, Lags, Differences, Logarithms and Growth Rates

If observations of a variable Y are recorded over time, we denote Y_t the value observed at time t. The period between two sequential observations Y_t and Y_{t-1} is a unit of time: hours, days, weeks, months, quarters, years and so on. Key Concept 14.1 introduces the essential terminology and notation for time series data we will use.

Key Concept 14.1

Lags, First Differences, Logarithms and Growth Rates

- Previous values of a time series are called *lags*. The first lag of Y_t is Y_{t-1}. The j^th lag of Y_t is Y_{t-j}. In R, lags of univariate or multivariate time series are conveniently computed by `lag()`, see `?lag`
- Sometimes we need to work with a differenced series. The first difference of a series is $\Delta Y_t = Y_t - Y_{t-1}$ — the difference between periods t and $t-1$. If `Y` is a time series, the first lag is computed as `Y-lag(Y)`.
- It may be convenient to work with the first difference in logarithms of a series. We denote this by $\Delta \log(Y_t) = \log(Y_t) - \log(Y_{t-1})$. For a time series `Y`, this obtained using `log(Y/lag(Y))`.
- $100\Delta Y_t$ is an approximation for the percentage change between Y_t and Y_{t-1}.

The definitions made in Key Concept 14.1 are useful because of two properties that are common to many economic time series:

- Exponential growth: some economic series grow approximately exponentially such that taking logarithm of the series makes them approximately linear.
- The standard deviation of many economic time series is approximately proportional to their level. Therefore, the standard deviation of the logarithm of such a series is approximately constant.

Furthermore, it is common to report rates of growth in macroeconomic series which is why log-differences are often used.

Table 14.1 of the book present quaterly U.S. GDP, its logarithm, the anualized growth rate and the first lag of the anualized growth rate series for the period 2012:Q1 - 2013:Q1. The following simple function can be used to compute these quantities for time series `series`.

```
# compute logarithms, annual growth rates and 1st lag of growth rates
quants <- function(series) {
  s <- series
  return(
    data.frame("Lead" = s,
               "Logarithm" = log(s),
               "AnnualGrowthRate" = 400 * log(s/lag(s)),
               "1stLagAnnualGrowthRate" = lag(400 * log(s/lag(s)))
      )
  )
}
```

Notice that the annual growth rate is computed using the approximation

$$AnnualGrowthY_t = 400 \cdot [\log(Y_t) - \log(Y_{t-1})]$$

discussed in Key Concept 14.1.

We call `quants()` on observations for the period 2011 Q3 - 2013 Q1.

```
quants(GDP["2011-07::2013-01"])
```

```
##             Lead Logarithm AnnualGrowthRate X1stLagAnnualGrowthRate
## 2011 Q3 15062.14  9.619940               NA                      NA
## 2011 Q4 15242.14  9.631819        4.7518062                      NA
## 2012 Q1 15381.56  9.640925        3.6422231               4.7518062
## 2012 Q2 15427.67  9.643918        1.1972004               3.6422231
## 2012 Q3 15533.99  9.650785        2.7470216               1.1972004
## 2012 Q4 15539.63  9.651149        0.1452808               2.7470216
## 2013 Q1 15583.95  9.653997        1.1392015               0.1452808
```

Autocorrelation

Observations of a time series are typically correlated. This type correlation is called *autocorrelation* or *serial correlation*. Key Concept 14.2 summarizes the concepts of population autocovariance and population autocorrelation and shows how to compute their sample equivalents.

Key Concept 14.2

Autocorrelation and Autocovariance

The covariance between Y_t and its j^{th} lag, Y_{t-j}, is called the j^{th} *autocovariance* of the series Y_t. The j^{th} *autocorrelation coefficient*, also called the *serial correlation coefficient*, measures the correlation between Y_t and Y_{t-j}.

We thus have

$$j^{th}\text{autocovariance} = cov(Y_t, Y_{t-j})$$
$$j^{th}\text{autocorrelation} = \rho_j = corr(Y_t, Y_{t-j}) = \frac{cov(Y_t, Y_{t-j})}{\sqrt{var(Y_t) var Y_{t-j}}}.$$

Population Autocovariance and population autocorrelation can be estimated by $\widehat{cov(Y_t, Y_{t-j})}$, the sample autocovariance, and $\widehat{\rho}_j$, the sample autocorrelation.

$$\widehat{cov(Y_t, Y_{t-j})} = \frac{1}{T}\sum_{t=j+1}^{T}(Y_t - \overline{Y}_{j+1:T})(Y_{t-j} - \overline{Y}_{1:T-j})$$
$$\widehat{\rho}_j = \frac{\widehat{cov(Y_t, Y_{t-j})}}{\widehat{var(Y_t)}}.$$

In R the function `acf()` from the package stats computes the sample autocovariance or the sample autocorrelation function.

Using `acf()` it is straightforward to compute the first four sample autocorrelations of the series `GDPGrowth`.

```
acf(na.omit(GDPGrowth), lag.max = 4, plot = F)
```

```
##
## Autocorrelations of series 'na.omit(GDPGrowth)', by lag
##
##  0.00  0.25  0.50  0.75  1.00
## 1.000 0.352 0.273 0.114 0.106
```

This is evidence that there is mild positive autocorrelation in the growth of GDP: if GDP grows faster than average in one period, there is a tendency that it grows faster than average in the following period.

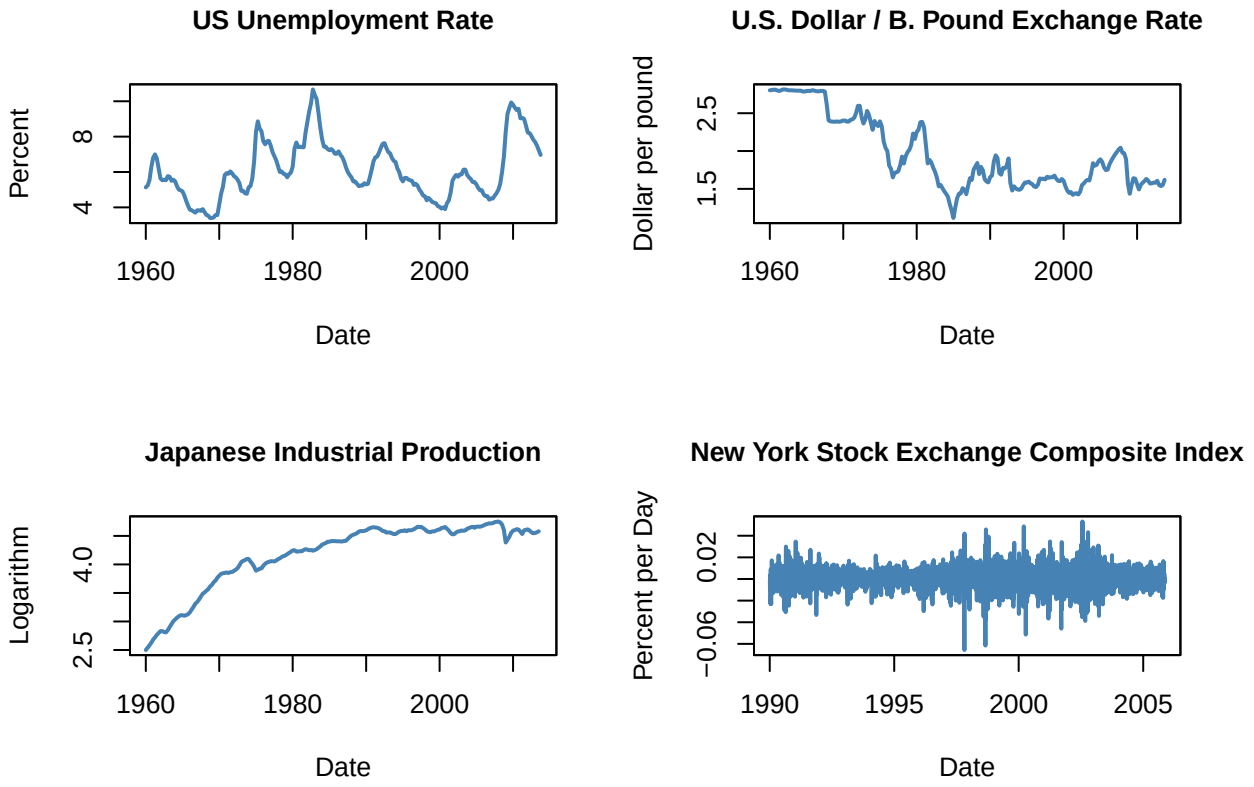
Other Examples of Economic Time Series

Figure 14.2 of the book presents four plots. The U.S. unemployment rate, the U.S. Dollar / British Pound exchange rate, The logarithm of the Janapese industrial production index as well as daily changes in the Whilshire 5000 stock price index, a financial time series. The next code chunck reproduces the plots of the three macroenomic series and adds percentage changes in the daily values of the New York Stock Exchange Composite index as a fourth one (the data set, `NYSESW` comes with the `AER` package).

```
# define series as xts objects
USUnemp <- xts(USMacroSWQ$UNRATE, USMacroSWQ$Date)["1960::2013"]

DollarPoundFX <- xts(USMacroSWQ$EXUSUK, USMacroSWQ$Date)["1960::2013"]

JPIndProd <- xts(log(USMacroSWQ$JAPAN_IP), USMacroSWQ$Date)["1960::2013"]

data("NYSESW")
NYSESW <- xts(Delt(NYSESW))
```

```
par(mfrow = c(2, 2))

plot(as.zoo(USUnemp),
```

```
     col = "steelblue",
     lwd = 2,
     ylab = "Percent",
     xlab = "Date",
     main = "US Unemployment Rate",
     cex.main = 1
)


plot(as.zoo(DollarPoundFX),
     col = "steelblue",
     lwd = 2,
     ylab = "Dollar per pound",
     xlab = "Date",
     main = "U.S. Dollar / B. Pound Exchange Rate",
     cex.main = 1
)

plot(as.zoo(JPIndProd),
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     xlab = "Date",
     main = "Japanese Industrial Production",
     cex.main = 1
)

plot(as.zoo(NYSESW),
     col = "steelblue",
     lwd = 2,
     ylab = "Percent per Day",
     xlab = "Date",
     main = "New York Stock Exchange Composite Index",
     cex.main = 1
)
```



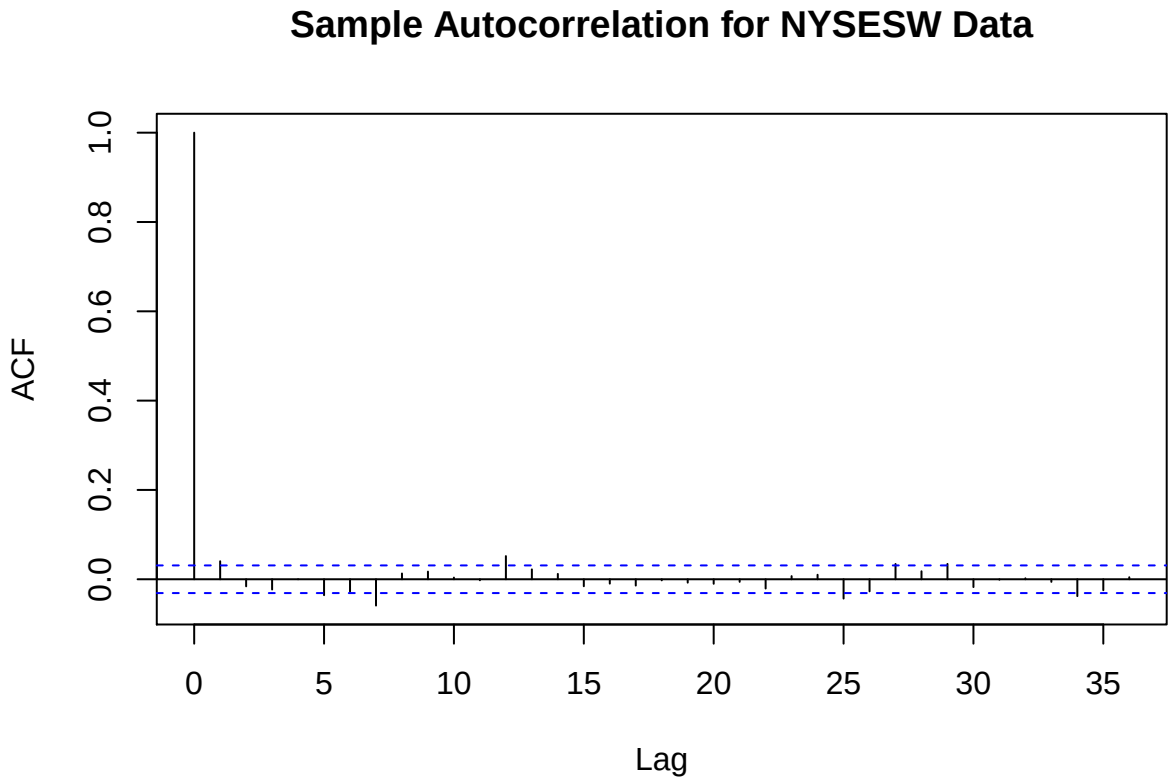
Note that the series show quite different characteristics. The unemployment rate increases during recessions and declines in times of economic recoveries and growth. The Dollar/Pund exchange rates shows a deterministic patern until the end of the Bretton Woods system. Japans industrial production exhibits an upward trend and decreasing growth. Daily changes in the New York Stock Exchange composite index seems to be random around the zero line. Sample autocorrelations confirm this conjecture.

```
acf(na.omit(NYSESW), plot = F, lag.max = 10)
```

```
##
## Autocorrelations of series 'na.omit(NYSESW)', by lag
##
##      0      1      2      3      4      5      6      7      8      9
##  1.000  0.040 -0.016 -0.023  0.000 -0.036 -0.027 -0.059  0.013  0.017
##     10
##  0.004
```

The first 10 sample autocorrelation coefficients are very close to zero. Further evidence can be found by looking at the plot generated by `acf()` by default.

```
acf(na.omit(NYSESW), main = "Sample Autocorrelation for NYSESW Data")
```



The blue dashed lines represent an approximate 95% confidence interval if there was no serial correlation. If a sample autocorrelation lies beyond these bands, it is statistically different from zero. For most lags we see that the sample autocorrelation does not exceed the confidence bands and there are only a few that lie marginally beyond the limits.

Furthermore, note that NYSWSW series shows a pattern which econometricians call *volatility clustering*: there are periods of high and periods of low variance. This is common for many financial time series.

15.3 Autoregressions

Growth forecasts are important for many economic entitites. For example, the production industry relies forecasts of GDP growth published by the central bank when deciding on future budgets and production plans. Autoregressive models are heavily used in economic forecasting. An autoregressive model relates a time series variable to its past values. Autoregressive models are heavily used in forecasting. This chapter discusses the basic ideas of autoregressions models, shows how they are estimated and discusses an application to forecasting of GDP growth using R.

15.3.0.1 The First-Order Autoregressive Model

It is intuitive that the immediate past of a time series should have power to predict its near future. The simplest autoregressive model uses only the most recent outcome of the time series observed to predict future value. Such a model is called a first-order autoregressive model which is often abbriviated AR(1) where the 1 indicates that the order is one.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t$$

is the AR(1) population model of a time series Y_t.

For the GDP growth series, an autoregressive model of order one uses only the information on GDP growth observed in the last quarter to predict a future growth rate. The first-order autoregression model of the GDP growth rate can be estimated by computing OLS estimates in the regression of $GDPGR_t$ on $GDPGR_{t-1}$,

$$\widehat{GDPGR}_t = \beta_0 + \beta_1 GDPGR_{t-1} + u_t. \tag{15.1}$$

Following the book we use data from 1962 to 2012 to estimate (15.1). This is easily done with the function `ar.ols()` from package stats.

```
# Subset data
GDPGRSub <- GDPGrowth["1962::2012"]

# estimate the model
ar.ols(GDPGRSub,
       order.max = 1,
       demean = F,
       intercept = T)
```

```
##
## Call:
## ar.ols(x = GDPGRSub, order.max = 1, demean = F, intercept = T)
##
## Coefficients:
##      1
## 0.3384
##
## Intercept: 1.995 (0.2993)
##
## Order selected 1  sigma^2 estimated as  9.886
```

We can check that the computations done by `ar.ols()` are the same as done by `lm()`.

```
# length of data set
N <-length(GDPGRSub)

GDPGR_leads <- as.numeric(GDPGRSub[-1])
GDPGR_lags <- as.numeric(GDPGRSub[-N])

# estimate the model
armod <- lm(GDPGR_leads ~ GDPGR_lags)
armod
```

```
##
## Call:
## lm(formula = GDPGR_leads ~ GDPGR_lags)
##
## Coefficients:
## (Intercept)   GDPGR_lags
##      1.9950       0.3384
```

As usual, we may use `coeftest()` to obtain a robust summary on the estimated regression coefficients.

```
# robust summary
coeftest(armod, vcov. = vcovHC(armod, type = "HC0"))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 1.994986   0.349539  5.7075 4.070e-08 ***
## GDPGR_lags  0.338436   0.075812  4.4641 1.339e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus the estimated model is

$$\widehat{GDPGR}_t = \underset{(0.350)}{1.995} + \underset{(0.076)}{0.338} GDPGR_{t-1}. \tag{15.2}$$

Notice that we omit the first observation for $GDPGR_{1962\ Q1}$ from the dependent variable vector since $GDPGR_{1962\ Q1-1} = GDPGR_{1961\ Q4}$, is not included in the sample. Similarly, the last observation, $GDPGR_{2012\ Q4}$, is excluded from the predictor vector since the data set does not include $GDPGR_{2012\ Q4+1} = GDPGR_{2013\ Q1}$. Put differently, when estimation the model, one observation is lost because of the time series structure of the data.

Forecasts and Forecast Errors

Suppose a random variable Y_t follows an AR(1) model with an intercept and that you have an OLS estimate of the model on the basis of observations for T periods. Then you may use the AR(1) model to obtain $Y_{T+1|T}$, a forecast for Y_{T+1} where

$$\widehat{Y}_{T+1|T} = \widehat{\beta}_0 + \widehat{\beta}_1 Y_T.$$

The forecast error is

$$\text{Forecast error} = Y_{T+1} - \widehat{Y}_{T+1|T}.$$

Forecasts versus predicted values

Note that *forecasted values* of Y_t are *not* what we refer to as *OLS predicted values* of Y_t. Also, the forecast error is *not* an OLS residual. Forecasts and forecast errors are obtained using *out-of-sample* values while predicted values and residuals are computed for actually *in-sample* values that were actually observed and used in estimation of the model.

The root mean squared forecast error (RMSFE) measures the typical size of the forecast error and is defined as

$$RMSFE = \sqrt{E\left[\left(Y_{T+1} - \widehat{Y}_{T+1|T}\right)^2\right]}.$$

Application to GDP Growth

Using (15.2), the estimated AR(1) model of GDP growth, we may perform the forecast for GDP growth in 2013:Q1 (remember that the model was estimated using data for periods 1962:Q1 - 2012:Q4 so 2013:Q1 is an out of sample period). This is done by plugging $GDPGR_{2012:Q4} \approx 0.15$ in (15.2),

$$\widehat{GDPGR}_{2013:Q1} = 1.995 + 0.348 \cdot 0.15 = 2.047.$$

`forecast()` from the forecast package has some useful features for forecasting of time series data.

```
library(forecast)

# GDP growth rate 2012:Q4
new <- data.frame("GDPGR_lags" = GDPGR_leads[N-1])

# predict GDP growth rate 2013:Q1
forecast(armod, newdata = new)
```

```
##   Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 1       2.044155 -2.036225 6.124534 -4.213414 8.301723
```

Using `forecast()` we obtain the same point forecast of about 2.0, along with 80% and 95% forecast intervals. We conclude that our AR(1) model forecasts GDP growth to be 2% in 2013:Q1.

How accurate is this forecast? First, notice that the forecast error is pretty big: $GDPGR_{2013:Q1} \approx 1.1\%$ while our forecast is 2%. Second, by calling `summary()` on `armod` we find that the model explains only little of the variation in the growth rate of GDP and the SER is about 3.16. Leaving aside forecast uncertainty due to estimation of β_0 and β_1, the $RMSFE$ must be at least 3.16% which is pretty inaccuarate.

```
# compute the forecast error
forecast(armod, newdata = new)$mean - GDPGrowth["2013"][1]
```

```
##                 x
## 2013 Q1 0.9049532
```

```
# R^2
summary(armod)$r.squared
```

```
## [1] 0.1149576
```

```
# SER
summary(armod)$sigma
```

```
## [1] 3.15979
```

The p^{th}-Order Autoregressive Model

For forecasting GPD growth in period t, the AR(1) model (15.2) disregards any information in the past of the series that is more distant than one period. An AR(p) model incorporates the information of p lags of the series. The idea is explained in Key Concept 14.3.

Key Concept 14.3

Autoregressions

An AR(p) model assumes that a time series Y_t can be represented by a linear function of p of its lagged values. We say that

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + u_t.$$

is an autoregressive model of order p where $E(u_t|Y_{t-1}, Y_{t-2}, \dots) = 0$.

Following the book, we estimate an AR(2) model of the GDP growth series from 1962:Q1 to 2012:Q4.

```
# estimate AR2 model
GDPGR_AR2 <- dynlm(ts(GDPGR_leads) ~ L(ts(GDPGR_leads)) + L(ts(GDPGR_leads), 2))

coeftest(GDPGR_AR2, vcov. = sandwich)
```

```
##
## t test of coefficients:
##
##                       Estimate Std. Error t value  Pr(>|t|)
## (Intercept)           1.631747   0.402023  4.0588 7.096e-05 ***
## L(ts(GDPGR_leads))    0.277787   0.079250  3.5052 0.0005643 ***
## L(ts(GDPGR_leads), 2) 0.179269   0.079951  2.2422 0.0260560 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimation yields

$$\widehat{GDPGR}_t = \underset{(0.40)}{1.63} + \underset{(0.08)}{0.28} GDPGR_{t-1} + \underset{(0.08)}{0.18} GDPGR_{t-1}. \tag{15.3}$$

We see that the coefficient on the second lag is significantly different from zero. Note that the fit improves slightly: $\overline{R^2}$ grows from 0.11 for the AR(1) model to about 0.14 and the SER reduces to 3.13.

```
# R^2
summary(GDPGR_AR2)$r.squared
```

```
## [1] 0.1425484
```

```
# SER
summary(GDPGR_AR2)$sigma
```

```
## [1] 3.132122
```

We may use the AR(2) model to obtain a forecast for GDP growth in 2013:Q1 in the same manner as for the AR(1) model.

```
# AR(2) forecast of GDP growth in 2013:Q1
"Forecast" <- c("2013:Q1" = coef(GDPGR_AR2) %*% c(1, GDPGR_leads[N-1], GDPGR_leads[N-2]))
```

This leads to a forecast error of roughly -1%.

```
# compute AR(2) forecast error
GDPGrowth["2013"][1] - Forecast
```

```
##                 x
## 2013 Q1 -1.025358
```

15.4 Can You Beat the Market? (Part I)

The thoery of efficient capital markets states stock prices embody all currently available information. If this hypothesis holds, it should not be possible to estimate a useful model for forecasting future stock returns using publicly available information on past returns (this is also refered to as the weak-form efficiency hypothesis): if it was possible to forecast the market, traders would be able to make arbitrage, e.g. by relying on an AR(2) model, they would use information that is not already priced-in which would be not consistent with the theory.

This idea is presented in the Box *Can You Beat the Market? (Part I)* on page 582 of the book. This section reproduces the estimation results.

We start by importing monthly data from 1931:1 to 2002:12 on excess returns of a broad-based index of stock prices, the CRSP value-weighted index. The data are provided by the authors of the book as an excel sheet which can be downloaded here.

```
# read in data on stock returns
SReturns <- read_xlsx("Data/Stock_Returns_1931_2002.xlsx",
                      sheet = 1,
                      col_types = "numeric"
                      )
```

We continue by converting the data to an object of class ts.

```
# convert to ts object
StockReturns <- ts(SReturns[, 3:4],
        start = c(1931, 1),
        end = c(2002, 12),
        frequency = 12)
```

Next, we estimate AR(1), AR(2) and AR(4) models of excess returns from 1960:1 to 2002:12.

```
# AR(1)
SR_AR1 <- dynlm(ExReturn ~ L(ExReturn),
      data = StockReturns, start = c(1960,1), end = c(2002,12))

# AR(2)
SR_AR2 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2),
      data = StockReturns, start = c(1960,1), end = c(2002,12))

# AR(4)
SR_AR4 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2) + L(ExReturn, 3) + L(ExReturn, 4),
      data = StockReturns, start = c(1960,1), end = c(2002,12))
```

After computing robust standard errors, we gather the results in a stargazer table.

```
rob_se <- list(
  sqrt(diag(sandwich(SR_AR1))),
  sqrt(diag(sandwich(SR_AR2))),
  sqrt(diag(sandwich(SR_AR4)))
  )
```

```
stargazer(SR_AR1, SR_AR2, SR_AR4,
  title = "Autoregressive Models of Monthly Excess Stock Returns",
  header = FALSE,
  model.numbers = F,
  omit.table.layout = "n",
  digits = 2,
```

```
  column.labels = c("AR(1)", "AR(2)", "AR(4)"),
  dep.var.caption  = "Excess returns on the CSRP value-weighted index",
  dep.var.labels.include = FALSE,
  covariate.labels = c("$excess return_{t-1}$", "$excess return_{t-2}$",
                       "$excess return_{t-3}$", "$excess return_{t-4}$",
                       "Intercept"),
  se = rob_se,
  omit.stat = c("rsq")
  )
```

Autoregressive Models of Monthly Excess Stock Returns

Excess returns on the CSRP value-weighted index

AR(1)

AR(2)

AR(4)

excess returnt-1

0.05

0.05

0.05

(0.05)

(0.05)

(0.05)

excess returnt-2

-0.05

-0.05

(0.05)

(0.05)

excess returnt-3

0.01

(0.05)

excess returnt-4

-0.02

(0.05)

Intercept

0.31

0.33*

0.33

(0.20)

(0.20)

(0.20)

Observations

516

516

516

Adjusted R2

0.001

0.001

-0.002

Residual Std. Error

4.33 (df = 514)

4.33 (df = 513)

4.34 (df = 511)

F Statistic

1.31 (df = 1; 514)

1.37 (df = 2; 513)

0.72 (df = 4; 511)

The results presented above are consistent with the hypothesis of efficient financial markets: there are no statistically significant coefficients in any of estimated models and the the hypotheses that the respective set lags has no power in explaining today's returns cannot be rejected. Notice also that $\overline{R^2}$ is almost zero in all models and even negative for the AR(4) model. This suggests that none of the models is useful for forecasting stock returns.

15.5 Additional Predictors and The ADL Model

Instead of only using the dependent variable's lags as predictors, an autoregressive distributed lag (ADL) model uses also other lags of other variables for forecasting. The general ADL model is summarized in Key Concept 14.4

Key Concept 14.4

The Autoregressive Distributed Lag Model

An ADL(p,q) model assumes that a time series Y_t can be represented by a linear function of p of its lagged values and q lags of X_t, another time series. We say that

$$\begin{aligned} Y_t = &\beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} \\ &+ \delta_1 X_{t-1} + \delta_2 X_{t-2} + \cdots + \delta_q X_{t-q} X_{t-q} + u_t. \end{aligned}$$

is an *autoregressive distributed lag model* with p lags of Y_t and q lags of X_t where $E(u_t|Y_{t-1}, Y_{t-2}, \ldots, X_{t-1}, X_{t-2}, \ldots) = 0$.

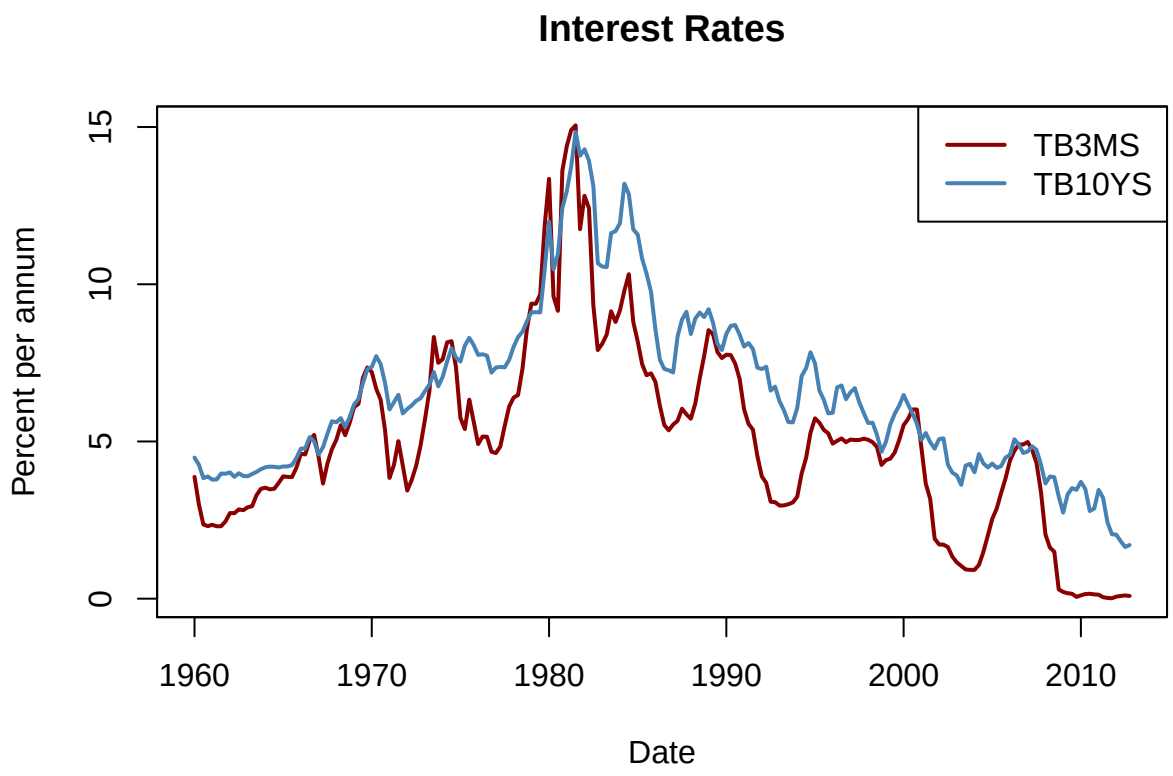
Forecasting GDP Growth Using the Term Spread

Interest on long-term and short term treasury bonds are closly linked to the macroeconomic development. While interest rates on both types of bonds have the same long-run tendencies, they behave quite differently in the short run. The difference in interest rates of two bonds with distinct matuarity is called the *term spread*.

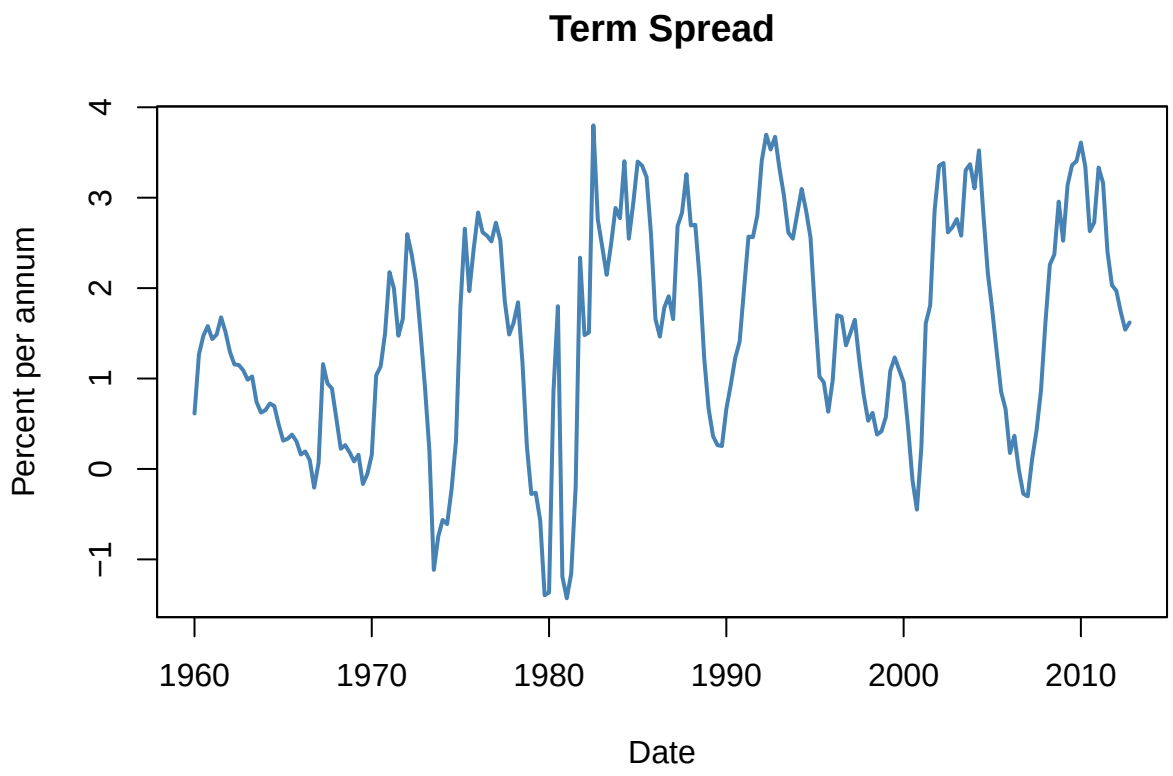
The following code chuncks reproduce figure 14.3 of the book which displays interest rates of 10-year U.S. Treasury bonds and 3 month U.S. Treasury bills from 1960 to 2012.

```
# 3 months Treasury bills interest rate
TB3MS <- xts(USMacroSWQ$TB3MS, USMacroSWQ$Date)["1960::2012"]

# 10 years Treasury bonds interest rate
TB10YS <- xts(USMacroSWQ$GS10, USMacroSWQ$Date)["1960::2012"]

# term spread
TSpread <- TB10YS - TB3MS
```

```
# reproduce figure 14.2 (a) of the book
plot(merge(as.zoo(TB3MS),as.zoo(TB10YS)),
     plot.type = "single",
     col = c("darkred", "steelblue"),
     lwd = 2,
     xlab = "Date",
     ylab = "Percent per annum",
     main = "Interest Rates"
)

legend("topright",
       legend = c("TB3MS","TB10YS"),
       col = c("darkred", "steelblue"),
       lwd = c(2,2)
       )
```




```
# reproduce figure 14.2 (b) of the book
plot(as.zoo(TSpread),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Percent per annum",
     main = "Term Spread"
)
```



Notice that before recessions, the gap between interests on long-term bonds and short term bills narrows and consequently the term spread declines drastically towards zero or even falls below zero in times of economic stress. This information might be used to improve forecasts of future GDP growth.

We check this by estimating an ADL(2,1) model and an ADL(2,2) model of the GDP growth rate using lags of GDP growth and lags of the term spread as regressors and use both models for forecasting the GDP growth in 2013:Q1.

```
# convert series to ts objects
GDPGrowth_ts <- ts(GDPGrowth,
                   start = c(1960, 1),
                   end = c(2013, 4),
                   frequency = 4)

TSpread_ts <- ts(TSpread,
                 start = c(1960, 1),
                 end = c(2012, 4),
                 frequency = 4)

# join both series
ADLdata <- ts.union(GDPGrowth_ts, TSpread_ts)
```

```
# estimate the ADL(2,1) model of GDP growth
GDPGR_ADL21 <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) + L(TSpread_ts),
      start = c(1962, 1), end = c(2012, 4))

coeftest(GDPGR_ADL21, vcov. = sandwich)
```

```
##
## t test of coefficients:
##
```

```
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.954990   0.486976  1.9611 0.051260 .
## L(GDPGrowth_ts)     0.267729   0.082562  3.2428 0.001387 **
## L(GDPGrowth_ts, 2) 0.192370   0.077683  2.4763 0.014104 *
## L(TSpread_ts)       0.444047   0.182637  2.4313 0.015925 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated equation of the ADL(2, 1) model is

$$\widehat{GDPGR}_t = \underset{(0.49)}{0.96} + \underset{(0.08)}{0.26}\,GDPGR_{t-1} + \underset{(0.08)}{0.19}\,GDPGR_{t-2} + \underset{(0.18)}{0.44}\,TSpread_{t-1} \tag{15.4}$$

Notice that all coefficients are significant at the level of 5%.

```
# 2012:Q3 / 2012:Q4 data on GDP growth and term spread
t <- window(ADLdata, c(2012, 3), c(2012, 4))

# ADL(2,1) GDP growth forecast for 2013:Q1
ADL21_forecast <- coef(GDPGR_ADL21) %*% c(1, t[2,1], t[1,1], t[2,2])
ADL21_forecast
```

```
##          [,1]
## [1,] 2.241689
```

```
# Forecast error
window(GDPGrowth_ts, c(2013,1), c(2013,1)) - ADL21_forecast
```

```
##           Qtr1
## 2013 -1.102487
```

Model (15.4) predicts the GDP growth in 2013:Q1 to be 2.24% which leads to a forecast error of −1.10%.

We estimate the ADL(2,2) specification to see whether adding additional information on past term spread improves the forecast.

```
# estimate the ADL(2,2) model of GDP growth
GDPGR_ADL22 <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) + L(TSpread_ts) + L(TSpread_ts

coeftest(GDPGR_ADL22, vcov. = sandwich)
```

```
##
## t test of coefficients:
##
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.967967   0.472470  2.0487 0.041800 *
## L(GDPGrowth_ts)      0.243175   0.077836  3.1242 0.002049 **
## L(GDPGrowth_ts, 2)  0.177070   0.077027  2.2988 0.022555 *
## L(TSpread_ts)       -0.139554   0.422162 -0.3306 0.741317
## L(TSpread_ts, 2)     0.656347   0.429802  1.5271 0.128326
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the ADL(2,2) model we obtain

$$\widehat{GDPGR}_t = \underset{(0.47)}{0.98} + \underset{(0.08)}{0.24} GDPGR_{t-1} \tag{15.5}$$

$$+ \underset{(0.08)}{0.18} GDPGR_{t-2} + \underset{(0.42)}{-0.14} TSpread_{t-1} + \underset{(0.43)}{0.66} TSpread_{t-2}. \tag{15.6}$$

The coefficients on both lags of the term spread are not significant at the 10% level.

```
# ADL(2,2) GDP growth forecast for 2013:Q1
ADL22_forecast <- coef(GDPGR_ADL22) %*% c(1, t[2,1], t[1,1], t[2,2], t[1,2])
ADL22_forecast
```

```
##          [,1]
## [1,] 2.274407
```

```
# Forecast error
window(GDPGrowth_ts, c(2013,1), c(2013,1)) - ADL22_forecast
```

```
##           Qtr1
## 2013 -1.135206
```

The ADL(2,2) forecast of GDP growth in 2013:Q1 is 2.27% which imples a forecast error of 1.14%.

Do the ADL models (15.4) and (15.6) improve upon the simple AR(2) model (15.3) in terms of forecasting GPD growth in 2013:Q1? The answer is yes: while SER and $\overline{R}^2$ improve only slightly, an F-test on the term spread coefficients in (15.6) provides evidence that the model does better in explaining GPD growth than the AR(2) model as the hypothesis that both coefficients are zero cannot be rejected at the significance level of 5%.

```
# compare adj. R2
c(
  "Adj.R2 AR(2)" = summary(GDPGR_AR2)$r.squared,
  "Adj.R2 ADL(2,1)" = summary(GDPGR_ADL21)$r.squared,
  "Adj.R2 ADL(2,2)" = summary(GDPGR_ADL22)$r.squared
)
```

```
##    Adj.R2 AR(2) Adj.R2 ADL(2,1) Adj.R2 ADL(2,2)
##       0.1425484       0.1743996       0.1855245
```

```
# compare SER
c(
  "SER AR(2)" = summary(GDPGR_AR2)$sigma,
  "SER ADL(2,1)" = summary(GDPGR_ADL21)$sigma,
  "SER ADL(2,2)" = summary(GDPGR_ADL22)$sigma
)
```

```
##    SER AR(2) SER ADL(2,1) SER ADL(2,2)
##     3.132122     3.070760     3.057655
```

```
# F-test on coefficients of term spread
linearHypothesis(GDPGR_ADL22,
                 c("L(TSpread_ts)=0","L(TSpread_ts, 2)=0"),
                 vcov. = sandwich
                 )
```

```
## Linear hypothesis test
##
## Hypothesis:
## L(TSpread_ts) = 0
```

```
## L(TSpread_ts, 2) = 0
##
## Model 1: restricted model
## Model 2: GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) + L(TSpread_ts) +
##     L(TSpread_ts, 2)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F  Pr(>F)
## 1    201
## 2    199  2 4.4344 0.01306 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Stationarity

In general, forecasts an be improved by using mutliple predictors — just as in cross-sectional regression. For time series regressions to yield reliable models, the assumption of *stationarity* must be fulfilled. Key Concept 14.5 explains what stationarity is.

Key Concept 14.5

Stationarity

We say that a time series Y_t is stationary if its probability distribution is time independent, that is the joint distribution $Y_{s+1}, Y_{s+2}, \ldots, Y_{s+T}$ does not change as s varies, regardless of T.

Similarly, we say that two time series X_t and Y_t are *jointly stationary* if the joint distribution of $(X_{s+1}, Y_{s+1}, X_{s+2}, Y_{s+2} \ldots, X_{s+T} Y_{s+T})$ does not depend on s, regardless of T.

In a probabilistic sense, stationarity means that information about how a time series evolves in the future is inherent to its past. If this is not the case, we cannot use the past of a series as a reliable guideline for its future.

Time Series Regression with Multiple Predictors

The concept of stationarity is a key assumption in the general time series regression model with multiple predictors. Key Concept 14.6 elaborates the model and its assumptions.

Key Concept 14.6

Time Series Regression with Multiple Predictors

The general time series regression model extends the ADL such that multiple regressors and their lags are included. It uses p lags of the dependent variable and q_l lags of l additional predictors where $l = 1, \ldots, k$:

$$\begin{aligned} Y_t = &\beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} \\ &+ \delta_{11} X_{1,t-1} + \delta_{12} X_{1,t-2} + \cdots + \delta_{1q} X_{1,t-q} \\ &+ \ldots \\ &+ \delta_{k1} X_{k,t-1} + \delta_{k2} X_{k,t-2} + \cdots + \delta_{kq} X_{k,t-q} \\ &+ u_t \end{aligned} \tag{15.7}$$

For estimation we make the following assumptions:

1. The error term u_t has conditional mean zero given all regressors and their lags:

$$E(u_t|Y_{t-1}, Y_{t-2}, \dots, X_{1,t-1}, X_{1,t-2} \dots, X_{k,t-1}, X_{k,t-2}, \dots)$$

   This assumption is an extension of the conditional mean zero assumption used for AR and ADL models and guarantees that the gernal time series regression model stated above gives the best forecast of Y_t given its lags, the additional regressors $X_{1,t}, \dots, X_{k,t}$ and their lags.

2. The *i.i.d* assumption for cross-sectional data is not (entirely) meaningful for time series data. We replace it by the following one with two parts:

   (a) The $(Y_t, X_{1,t}, \dots, X_{k,t})$ have a stationary distribution (the "identically distributed" part of the i.i.d. assumption for cross-setional data). If this does not hold, forecasts may be biased and infernce can be strongly misleading.

   (b) $(Y_t, X_{1,t}, \dots, X_{k,t})$ and $(Y_{t-j}, X_{1,t-j}, \dots, X_{k,t-j})$ become independent as j gets large (the "idependly" distributed part of the i.i.d. assumption for cross-sectional data). This assumption is also called *weak dependence*. It ensures that, in large samples, the WLLN and the central limit theorem hold.

3. Large outliers are unlikely: $E(X_{1,t}^4), E(X_{2,t}^4), \dots, E(X_{k,t}^4)$ and $E(Y_t^4)$ have nonzero, finite fourth moments.

4. No perfect multicollinearity.

Since many economic time series appear to be nonstationary, assumption 2 of Key Concept 14.6 is a crucial one in applied macroeconomics and finance which is why statistical test for stationarity / nonstationarity have been developed. Chapters 14.6 and 14.7 are devoted to this topic.

Statistical inference and the Granger causality test

If a X is a useful predictor for Y, in a regression of Y_t on lags of its own and lags of X_t, not all of the coefficients on the lags on X_t are zero. This concept is called "Granger causality" and is an interesting hypothesis to test. Key Concept 14.7 summarizes the idea.

Key Concept 14.7

Granger Causality Tests

The Granger causality test (Granger, 1969) is an F-test of the null hypothesis that *all* lags of a variable X included in a time series regression model do not have predictive power for Y_t. The Granger causality test does not test whether X actually *causes* Y but whether the included lags have are informative in terms of predicting Y.

Notice that we have already performed a Granger causality test on the coefficients of term spread in (15.6), the ADL(2,2) model of GDP growth and concluded that at least on of the first two lags of term spread has predictive power for GDP growth.

Forecast Uncertainty and Forecast Intervals

In general, it is a good practice to report a measure of the uncertainty inherent the estimation. Uncertainty is particulary of interest when forecasting a time series. For example, consider a simple ADL$(1,1)$ model

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \delta_1 X_{t-1} + u_t$$

where u_t is a homoskedastic error term. Then, the forecast error is

$$Y_{T+1} - \widehat{Y}_{T+1|T} = u_{T+1} - \left[(\widehat{\beta}_0 - \beta_0) + (\widehat{\beta}_1 - \beta_1)Y_T + (\widehat{\delta}_1 - \delta_1)X_t\right]$$

The mean squared forecast error (MSFE) and the RMFSE are

$$\begin{aligned} MFSE &= E\left[(Y_{T+1} - \widehat{Y}_{T+1|T})^2\right] \\ &= \sigma_u^2 + var\left[(\widehat{\beta}_0 - \beta_0) + (\widehat{\beta}_1 - \beta_1)Y_T + (\widehat{\delta}_1 - \delta_1)X_t\right], \\ RMFSE &= \sqrt{\sigma_u^2 + var\left[(\widehat{\beta}_0 - \beta_0) + (\widehat{\beta}_1 - \beta_1)Y_T + (\widehat{\delta}_1 - \delta_1)X_t\right]}. \end{aligned}$$

A 95% forecast interval is an interval that covers the true value of Y_{T+1} in 95% of repeated applications. Note that there is a major difference in computing a confidence interval and a forecast interval: when computing a confidence interval of a point estimate we use large sample approximations that are justified by the central limit theorem and thus are valid for a large range of error term distributions. For computation of a forecast interval of Y_{T+1}, however, we must make an additional assumption about the distribution of u_{T+1}, the error term in period $T+1$. Assuming that u_{T+1} is normally distributed on can construct a 95% *forecast interval* for Y_{T+1} using $SE(Y_{T+1} - \widehat{Y}_{T+1|T})$, an estimate of the RMSFE:

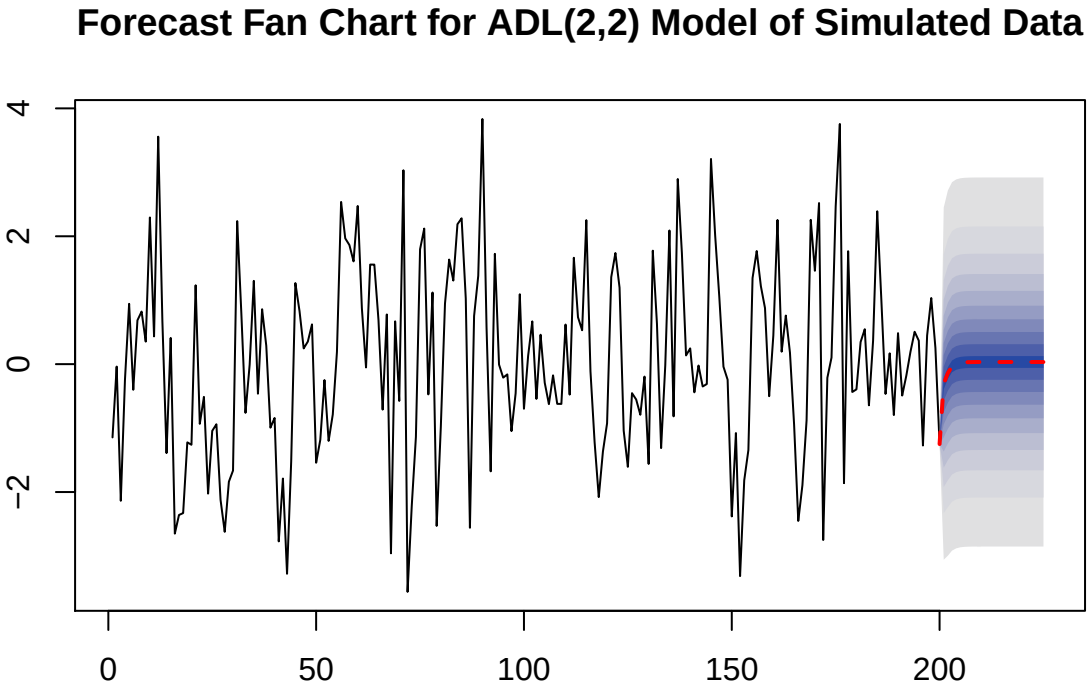
$$\widehat{Y}_{T+1|T} \pm 1.96 \cdot SE(Y_{T+1} - \widehat{Y}_{T+1|T})$$

Of course the computation gets more complicated when the error term is heteroskedastic or if we are interested in computing a forecast interval for $T+s, s>1$.

In some applications it is useful to report multiple forecast intervals for each step in range of subsequents periods, see the Box *The River of Blood* on p. 592 of the book. Such forecast ranges can be visualized in a so-called fan chart. We will not replicate the fan chart presented in Figure 14.2 of book because the underlying model is by far more compley than the simple AR and ADL models treated here. Instead, in the example below we use simulated time series data and estimate an ADL(1,1) model which is than used for forecasting the subsequent 25 future outcomes of the series. We choose `level = seq(5,99,10)` in the call of `forecast()` such that forecast intervals with levels $5\%, 15\%, \ldots, 95\%$ are computed for each point forecast of the series.

```
# simulate time series
set.seed(1234)

Z <- arima.sim(list(order = c(1, 0, 0), ar = 0.5),  n = 200)
X <- arima.sim(list(order = c(1, 0, 0), ar = 0.2),  n = 200)

Y <- 0.7 * Z + 0.7 * X + rnorm(100)

# estimate an ADL(1,1) model using arima, see ?arima
model <- arima(Y, order = c(2, 0, 0))

# compute points forecasts and prediction intervals for next 25 periods
fc <- forecast(model, h = 25, level = seq(5, 99, 10))

# plot fan chart
plot(fc,
     main = "Forecast Fan Chart for ADL(2,2) Model of Simulated Data",
```

```
    showgap = F,
    fcol = "red",
    flty = 2)
```



The dashed red line shows point forecasts of the series for the next 25 periods based on an $ADL(1,1)$ model and the shaded areas represent the prediction intervals. The degree of shading indicates the level of the prediction interval. The darkest of the blue bands displays the 5% forecast intervals and the color fades towards grey as the level of the intervals increases.

15.6 Lag Length Selection Using Information Criteria

The selection of lag lengths in AR and ADL models can be partially governed by economic theory. However, there are statistical methods that are helpful to determine how many lags should be included as regressors. In general, to many lags inflate the standard errors of coefficient estimates and thus imply an increase in the forecast error while omitting lags that should be included in the model may result in estimation bias.

The order of an AR model can be determined using two approaches

1. **F-test approach**:

   Estimate an AR(p) model and test significance of the largest lag(s). If the test rejects, drop the respective lag(s) from the model. This approach has the tendency to produce models where the order is too large: in a significance test we always face the risk of rejecting a true null hypothesis!

2. **Relying on an information criterion**:

   The circumvent the issue of producing too large models, one may choose the lag order that minimizes one of the following two information criteria:

   - The *Bayes information criterion* (BIC):

$$BIC(p) = \log\left(\frac{SSR(p)}{T}\right) + (p+1)\frac{\log(T)}{T}$$

- The *Akaike information criterion* (AIC):

$$AIC(p) = \log\left(\frac{SSR(p)}{T}\right) + (p+1)\frac{2}{T}$$

Both criteria are estimators of the optimal lag length p. The lag order $\widehat{p}$ that minimizes the respective criterion is called the *BIC estimate* or the *AIC estimate* of the optimal model order. The basic idea of both criteria is that the SSR decreases as additional lags are added to the model such that the first addend decreases wheres the second addend increases as the lag order grows. One can show that the the BIC is a consistent estimator of the true lag order while the AIC is not which is due to the differing factors in the second addend. Nevertheless, both estimators are used in practice where the AIC is sometimes used as an alternative when the BIC yields a model with too few lags.

`dynlm()` does not compute informatin criteria by default. We will therefore write a short function that reports the BIC (and also the chosen lag order p and R^2) for objects of class `dynlm`.

```
# compute BIC for AR models
BIC <- function(model) {

  ssr <- sum(model$residuals^2)
  t <- length(model$residuals)
  npar <- length(model$coef)

  return(
    round(
      c(
      "p" = npar - 1,
      "BIC" = log(ssr/t) + npar * log(t)/t,
      "R2" = summary(model)$r.squared
      ), 4
    )
  )

}
```

Table 14.3 of the book presents a breakdown of how the BIC is computed for AR(p) models of GDP growth with order $p = 1, \dots, 6$. The final result can be easily reproduced using `sapply()` and the function `BIC()` defined above.

```
# Apply BIC() to an intercept only model of GDP growth
BIC(
  dynlm(ts(GDPGR_leads) ~ 1)
)
```

```
##      p    BIC     R2
## 0.0000 2.4394 0.0000
```

```
# loop BIC over models of different orders
order <- 1:6

BICs <- sapply(order,
       function(x)
        "AR" = BIC(
          dynlm(ts(GDPGR_leads) ~ L(ts(GDPGR_leads), 1:x))
          )
       )
```

```
BICs
```

```
##       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## p   1.0000 2.0000 3.0000 4.0000 5.0000 6.0000
## BIC 2.3486 2.3475 2.3774 2.4034 2.4188 2.4429
## R2  0.1143 0.1425 0.1434 0.1478 0.1604 0.1591
```

Note that increasing the lag order increases R^2 becaused SSR decreases as additional lags are added to the model but according to the BIC, we should decide for the AR(2) model instead of the AR(6) model. It helps to decide wether the decrease in SSR is enough to justify adding another regressor.

If we would have to compare a bigger set of models, a convenient way to select the model with the lowest BIC is using the function `which.min()`

```
# select the AR model with the smallest BIC
BICs[, which.min(BICs[2, ])]
```

```
##      p    BIC     R2
## 2.0000 2.3475 0.1425
```

The BIC may also be used to select lag lengths in time series regression models with multiple predictors. In a model with K coefficients, including the intercept, we have

$$BIC(K) = \log\left(\frac{SSR(K)}{T}\right) + K\frac{\log(T)}{T}.$$

Notice that choosing the optimal model according to the BIC can be computationally demaning because there may be many different combinations of lag lengths when there are more multiple predictors.

To motivate an example, we estimate ADL(p,q) models of GDP growth where, as above, the additional variable is the term spread between short-term and long-term bonds. We impose the restriction that $p = q_1 = \cdots = q_k$ so that only p_{max} models ($p = 1, \ldots, p_{max}$) need to be estimated. In the example below we choose $p_{max} = 12$.

```
# loop BIC over ADL models
order <- 1:12

BICs <- sapply(order,
       function(x)
         BIC(
          dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts, 1:x) + L(TSpread_ts, 1:x),
          start = c(1962, 1), end = c(2012, 4))
          )
       )

BICs
```

```
##       [,1]   [,2]   [,3]   [,4]    [,5]    [,6]    [,7]    [,8]    [,9]
## p   2.0000 4.0000 6.0000 8.0000 10.0000 12.0000 14.0000 16.0000 18.0000
## BIC 2.3411 2.3408 2.3813 2.4181  2.4568  2.5048  2.5539  2.6029  2.6182
## R2  0.1417 0.1855 0.1950 0.2072  0.2178  0.2211  0.2234  0.2253  0.2581
##       [,10]   [,11]   [,12]
## p   20.0000 22.0000 24.0000
## BIC  2.6646  2.7205  2.7664
## R2   0.2678  0.2702  0.2803
```

Notice that from the definition of `BIC()`, for ADL models with $p = q$ it follows that p reports the number of estimated coefficients *excluding* the intercept. Tus the lag order is obtained by deviding p by 2.

```
# select the ADL model with the smallest BIC
BICs[, which.min(BICs[2, ])]
```

```
##      p    BIC     R2
## 4.0000 2.3408 0.1855
```

The BIC is in favour of the ADL(2,2) model (15.6) we have estimated before.

15.7 Nonstationarity I: Trends

If a series is nonstationary, conventional hypothesis tests, confidence intervals and forecasts can be strongly misleading. The assumption of stationarity is violated if a series exhibits trends or breaks and the resulting complications in an econometric analysis depend on the specific type of nonstationarity. This section focuses on time series that exhibit trends.

A series is said to exhibit a trend if it fluctuates around a persistent long-term movement. One distinguishes between *deterministic* and *stochastic* trends.

- We say that a trend is *deterministic* if it is a nonrandom function of time.
- A trend is said to be *stochastic* if it is a random function of time.

A careful look at the figures we have produced in Chapter 14.2 reveals that many economic time series show a trending behaviour that is probably best modeled by stochastic trends. This is why the book focuses on the treatment of stochastic trends.

The Random Walk Model of a Trend

The simplest way to model a time series Y_t that has stochastic trend is the *random walk*

$$Y_t = Y_{t-1} + u_t \tag{15.8}$$

where the u_t are i.i.d. errors with $E(u_t|Y_{t-1}, Y_{t-2}, \dots) = 0$. Note that

$$\begin{aligned} E(Y_t|Y_{t-1}, Y_{t-2} \dots) &= E(Y_{t-1}|Y_{t-1}, Y_{t-2} \dots) + E(u_t|Y_{t-1}, Y_{t-2} \dots) \\ &= Y_{t-1} \end{aligned}$$

so the best forecast for Y_t, todays value of Y, is Y_{t-1}, the observation made yesterday so the difference between Y_t and Y_{t-1} is unpredictable. One can shows that the path followed by Y_t consists of random steps u_t, hence it is called a random walk.

Assume that Y_0, the starting of a random walk is 0. Another way to write out (15.8) is

$$
\begin{aligned}
Y_0 &= 0 \\
Y_1 &= 0 + u_1 \\
Y_2 &= 0 + u_1 + u_2 \\
&\vdots \\
Y_t &= \sum_{i=1}^{t} u_i.
\end{aligned}
$$

Therefore we have that

$$
\begin{aligned}
var(Y_t) &= var(u_1 + u_2 + \cdots + u_t) \\
&= t\sigma_u^2.
\end{aligned}
$$

Thus the variance of a random walk depends on time which violates the assumption presented in Key Concept 14.5: a random walk is nonstationary.

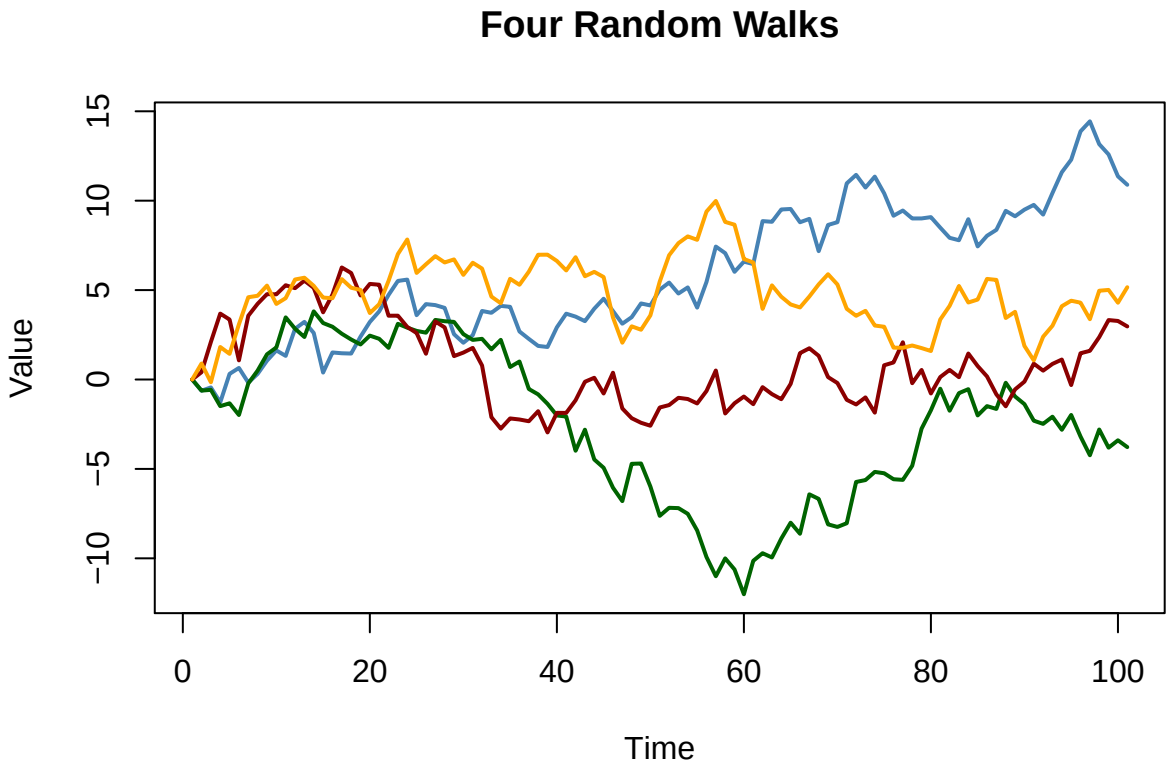
Obviously, (15.8) is a special case of an AR(1) model where the $\beta_1 = 1$. One can show that a time series that follows an AR(1) model is stationary if $|\beta_1| < 1$. In a general AR(p) model, stationarity is linked to the roots of the polynomial

$$1 - \beta_1 z - \beta_2 z^2 - \beta_3 z^3 - \cdots - \beta_p z^p.$$

If all roots are greater than 1 in absolute value, the AR(p) series is stationary. If at least one root equals 1, the AR(p) is said to have a *unit root* and thus has a stochastic trend.

It is straightforward to simulate random walks in R using `arima.sim()`. The function `matplot()` is covenient for simple plots of the columns of a matrix.

```
# simulate and plot random walks starting at 0
set.seed(1)

RWs <- ts(
  replicate(n = 4,
            arima.sim(model = list(order = c(0, 1 ,0)), n = 100)
            )
  )

matplot(RWs,
        type ="l",
        col = c("steelblue", "darkgreen", "darkred", "orange"),
        lty = 1,
        lwd = 2,
        main = "Four Random Walks",
        xlab = "Time",
        ylab = "Value"
        )
```

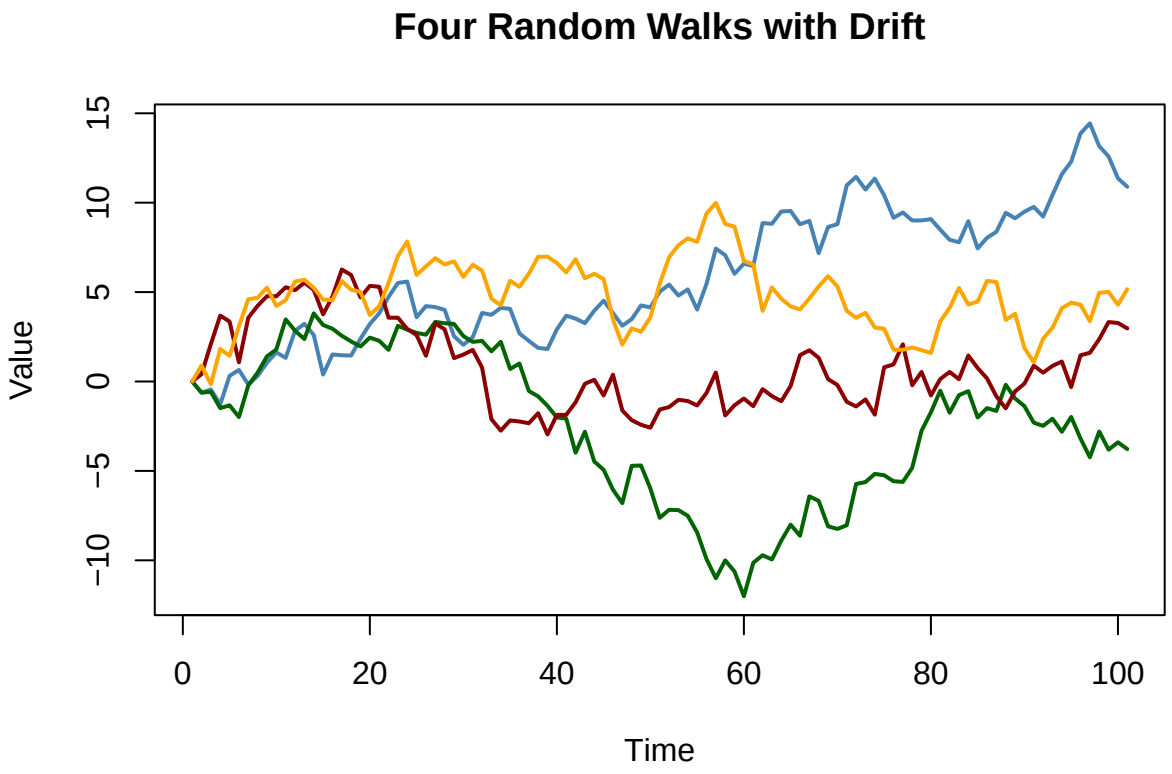


Adding a constant to (15.8) yields

$$Y_t = \beta_0 + Y_{t-1} + u_t, \tag{15.9}$$

a *random walk with drift*. This allows to model the tendency of a series to move in one direction or the other. If β_0 is positive, the series drifts upwards and it follows a downward trend if β_0 is negative.

```
# simulate and plot random walks with drift starting at 0
set.seed(1)

RWsd <- ts(
  replicate(n = 4,
            arima.sim(model = list(order = c(0,1,0)), n = 100)
            )
  )

matplot(RWsd,
        type="l",
        col = c("steelblue","darkgreen","darkred","orange"),
        lty = 1,
        lwd = 2,
        main = "Four Random Walks with Drift",
        xlab = "Time",
        ylab = "Value"
        )
```



Problems Caused by Stochastic Trends

OLS estimation of the coefficients on regressors that have a stochastic trend is problematic because the distribution of the estimator and its t-statistic is nonnormal, even asymptotically. This has various consequences:

- Downward bias of autoregressive coefficients

  If Y_t is a random walk, the coefficient β_1 can be consistently estimated by OLS but the estimator is biased toward zero. This bias is roughly $E(\widehat{\beta}_1) = 1 - 5.3/T$ which is substantial for sample sizes encountered in macroeconomics. This estimation bias causes forecasts of Y_t to perform worse than a pure random walk model.

- Nonnormally distributed t-statistics

  The nonnormal distributions of OLS estimates of the coefficient on a stochastic regressors translates to a nonnormal distributions of its t-statistic so that normal critical values are invalid and therefore usual confidence intervals and hypothesis tests are invalid, too, and the true distribution of the t-statistic cannot be readily determined.

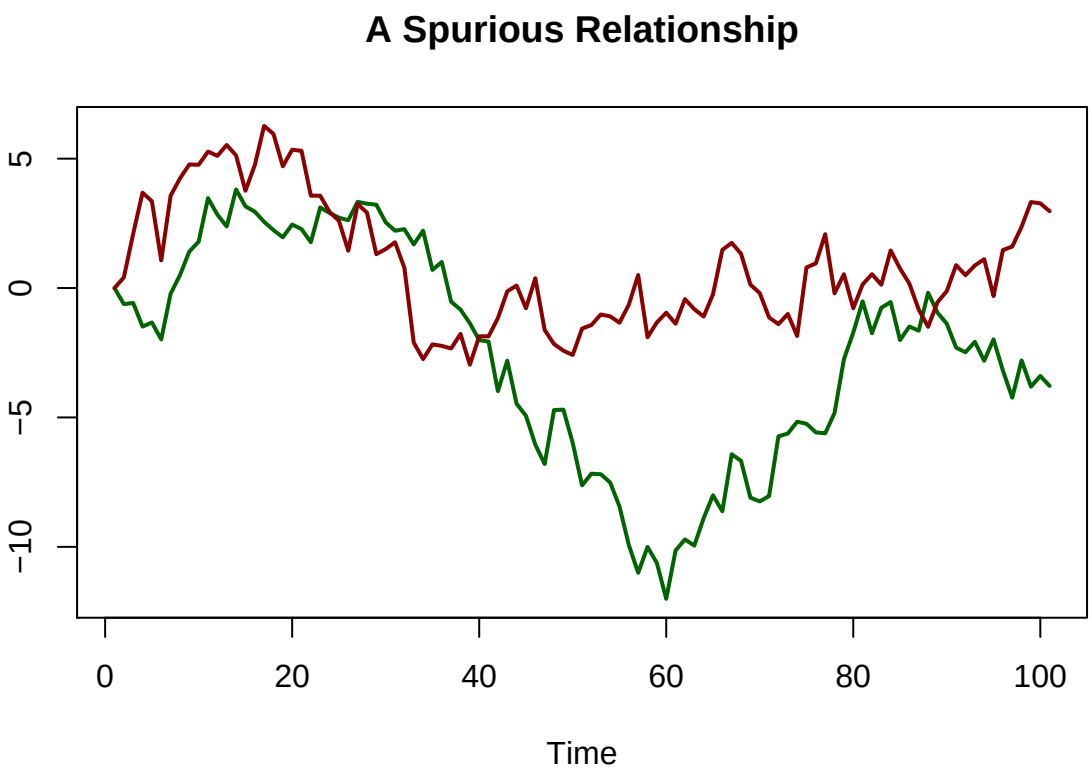
- Spurious Regression

  When a time series that exhibits a stochastic trend is regressed on another time series that does have a stochastic trend too, the estimated relationship may appear highly significant. This is what econometricians call a *spurious* relationship.

As an example for spurious regression, consider again the green and the red random walks that we have simulated above. We know that there is no relationship between both series: they are purely random and independent of each other.

```
# spurious relationship
matplot(RWs[,c(2,3)],
        lty = 1,
```

```
        lwd = 2,
        type = "l",
        col = c("darkgreen","darkred"),
        xlab = "Time",
        ylab = "",
        main = "A Spurious Relationship"
        )
```



Imagine we did not have this information and instead conject that the green series is useful for predicting the red series and thus end up estimating the ADL(0,1) model

$$Red_t = \beta_0 + \beta_1 Green_{t-1} + u_t.$$

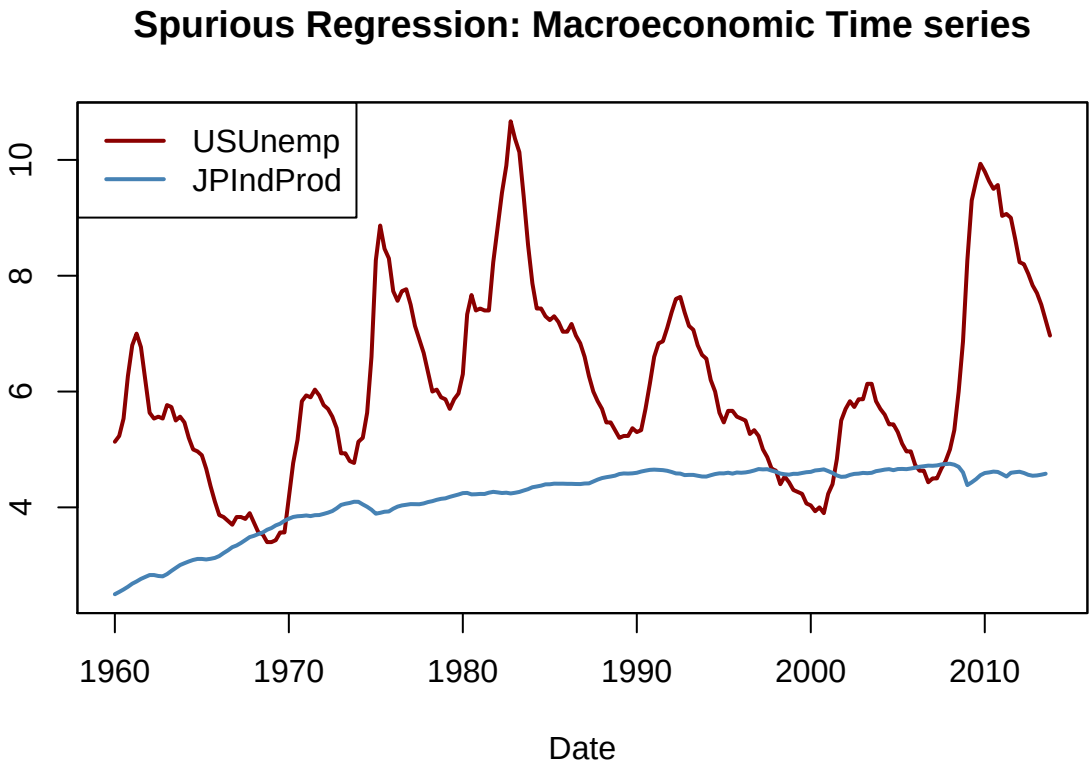
```
# estimate spurious AR model
summary(
  dynlm(RWs[,2] ~ L(RWs[,3]))
  )$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -3.459488  0.3635104 -9.516889 1.354156e-15
## L(RWs[, 3])  1.047195  0.1450874  7.217687 1.135828e-10
```

The result is obviously spurious: the coefficient on $Green_{t-1}$ is estimated to be about 1 and the p-value of $1.14 \cdot 10^{-10}$ of the corresponding t-test indicates that the coefficient is highly significant while its true value is in fact zero.

As an empirical example, consider the U.S. unemployment rate and the Japanese industrial production. Both series show an upward trending behaviour from the mod-1960s through the early 1980s.

```
# Plot U.S. unemployment rate & Japanese industrial production
plot(merge(as.zoo(USUnemp), as.zoo(JPIndProd)),
     plot.type = "single",
     col = c("darkred","steelblue"),
     lwd = 2,
     xlab = "Date",
     ylab = "",
     main = "Spurious Regression: Macroeconomic Time series"
)
legend("topleft",
       legend = c("USUnemp","JPIndProd"),
       col = c("darkred", "steelblue"),
       lwd = c(2, 2)
       )
```




```
# Estimate regression using data from 1962 to 1985
SR_Unemp1 <- dynlm(ts(USUnemp["1962::1985"]) ~ ts(JPIndProd["1962::1985"]))
coeftest(SR_Unemp1, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##                              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                  -2.37452    1.12041 -2.1193    0.0367 *
## ts(JPIndProd["1962::1985"])  2.22057    0.29233  7.5961 2.227e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A simple regression of the U.S. unemployment rate on Japanese industrial production using data from 1962 to 1985 yields

$$\widehat{U.S.UR}_t = \underset{(1.12)}{-2.37} + \underset{(0.29)}{2.22} \log(JapaneseIP)_t. \tag{15.10}$$

This appears to be a significant relationship: the t-statistic of the coefficient on $\log(JapaneseIP)_t$ is bigger than 7.

```
# Estimate regression using data from 1986 to 2012
SR_Unemp2 <- dynlm(ts(USUnemp["1986::2012"]) ~ ts(JPIndProd["1986::2012"]))
coeftest(SR_Unemp2, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##                              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                   41.7763     5.4066  7.7270 6.596e-12 ***
## ts(JPIndProd["1986::2012"])  -7.7771     1.1714 -6.6391 1.386e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

By estimating the same model, this time with data from 1986 to 2012, we obtain

$$\widehat{U.S.UR}_t = \underset{(5.41)}{41.78} + \underset{(1.17)}{-7.78} \log(JapaneseIP)_t \tag{15.11}$$

which is suprisingly quite different from (15.10) which indicates a moderate postive relationship, in contrast to the strong negative (15.11). This phenomenon can be attributed to stochastic trends in the series. Since, there is no economic reasoning that relates both trends, both regressions are spurious.

Testing for a Unit AR Root

A formal test for a stochastic trend has been proposed by Dickey and Fuller (1979) and is therefore termend the *Dickey-Fuller test*. As discussed above, a time series that follows an AR(1) model with $\beta_1 = 1$ has a stochastic trend. Thus, the testing problem is

$$H_0 : \beta_1 = 1 \quad \text{vs.} \quad H_1 : \beta_1 < 1.$$

so the null hypothesis is that the AR(1) has a unit root and the alternative hypothesis is that it is stationary. One often rewrites the AR(1) by substracting Y_{t-1} on both sides:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t \quad \Leftrightarrow \quad \Delta Y_t = \beta_0 + \delta_1 Y_{t-1} +$$

where $\delta_1 = \beta_1 - 1$. The testing problem then becomes

$$H_0 : \delta_1 = 0 \quad \text{vs.} \quad H_1 : \beta_1 < 0$$

which is convenient since the corresponding test statistic by default reported by relevant R functions.[1]

[1]The t-statistic of the Dickey-Fuller test is computed using homoskedasticity-only standard errors since under the null hypothesis, the usual t-statistic is robust to heteroskedasticity.

The Dickey-Fuller test can also be applied in an AR(p) model. The *Augmented Dickey-Fuller (ADF) test* is summarized in Key Concept 14.8.

Key Concept 14.8

The ADF Test for a Unit Root

Consider the regression

$$\Delta Y_t = \beta_0 + \delta Y_{t-1} + \gamma_1 \Delta_1 Y_{t-1} + \gamma_2 \Delta Y_{t-2} + \cdots + \gamma_p \Delta Y_{t-p} + u_t. \tag{15.12}$$

The ADF test for a unit autoregressive root test the hypothesis $H_0 : \delta = 0$ (stochastic trend) against the one-sided alternative $H_1 : \delta < 0$ (stationarity) using the usual OLS t-statistic.

If it is assumed that Y_t is stationary around a deterministic linear time trend, the model is augmented by the regressor t, that is Y_t becomes

$$\Delta Y_t = \beta_0 + at + \delta Y_{t-1} + \gamma_1 \Delta_1 Y_{t-1} + \gamma_2 \Delta Y_{t-2} + \cdots + \gamma_p \Delta Y_{t-p} + u_t \tag{15.13}$$

where again $H_0 : \delta = 0$ is tested against $H_1 : \delta < 0$.

The optimal lag length p can be estimated using information criteria. Notice that in the regression (15.12), $p = 0$ (that is no lags of ΔY_t are used as regressors) corresponds to a simple AR(1).

Under the null hypothesis, the t-statistic for $H_0 : \delta = 0$ does not have a normal distribution. The cricital values can only be obtained from simulation and differ for regressions (15.12) and (15.13) since the distribution of the ADF test statistic is sensitive to whether a deterministic time trend is included or not.

Critical Values for the ADF Statistic

Key Concept 14.8 states that the critical values for the ADF test in the regressions (15.12) and (15.13) can only be determined using simulation. The idea of the simulation study is to simulate a large number of ADF test test statistics and use them to estimate quantiles of their distribution. This sections shows how this is feasible whithin R.

First, consider an AR(1) model with drift. The procedure is as follows:

- Simulate N random walks with n observations using the data generating process

  $$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t,$$

  $t = 1, \ldots, n$ where N and n are large numbers.
- For each random walk, estimate the regression

  $$\Delta Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t,$$

  compute ADF test statistic. Save all N test statistics in a vector.
- Estimate quantiles of the distribution of the ADF test statistics using the N test statistics obtained from the simulation.

For the case with drift and linear time trend we replace the data generating process by

$$Y_t = \beta_0 + \alpha t + \beta_1 Y_{t-1} + u_t$$

and estimate

$$\Delta Y_t = \beta_0 + \alpha t + \beta_1 Y_{t-1} + u_t,$$

Loosely speaking, the precision of the estimated quantiles depends on two factors: n, the length of the underlying series and N, the number of test statistics used. Since we are interested in estimating quantiles of the *asymptotic* distribution (the Dickey-Fuller distribution) of the ADF test statistic so both using many observations and large number of simulated test statistics will increase the precision of the estimated quantiles. We choose $n = N = 1000$ as the computational burden grows quickly with n and N.

```
# repititions
N <- 1000

# observations
n <- 1000

# define drift an trend
drift <- 0.5
trend <- 1:n

# simulate N random walks with drift
RWD <- ts(replicate(n = N,
            drift + arima.sim(model = list(order = c(0,1,0)),
                              n = n - 1)
            )
          )

# compute ADF test statistics and store them in 'ADFD'
ADFD <- numeric(N)

for(i in 1:ncol(RWD)) {
  ADFD[i] <- summary(
    dynlm(diff(RWD[,i],1) ~ L(RWD[,i],1))
    )$coef[2,3]
}

# simulate N random walks with drift + trend
RWDT <- ts(replicate(n = N,
                     trend + drift + arima.sim(model = list(order = c(0,1,0)),
                                               n = n - 1)
                     )
           )

# compute ADF test statistics and store them in 'ADFDT'
ADFDT <- numeric(N)

for(i in 1:ncol(RWDT)) {
  ADFDT[i] <- summary(
```

```
    dynlm(diff(RWDT[,i],1) ~ L(RWDT[,i],1) + trend(RWDT[,i], scale = F))
  )$coef[2,3]
}
```

```
# estimate quantiles for ADF regression with drift
round(quantile(ADFD, c(0.1,0.05,0.01)),2)
```

```
##   10%    5%    1%
## -2.62 -2.83 -3.39
```

```
# estimate quantiles for ADF regression with drift + trend
round(quantile(ADFDT, c(0.1,0.05,0.01)),2)
```

```
##   10%    5%    1%
## -3.11 -3.43 -3.97
```

The estimated quantiles are close to the large sample critical values of the ADF test statistic reported in Table 14.4 of the book.

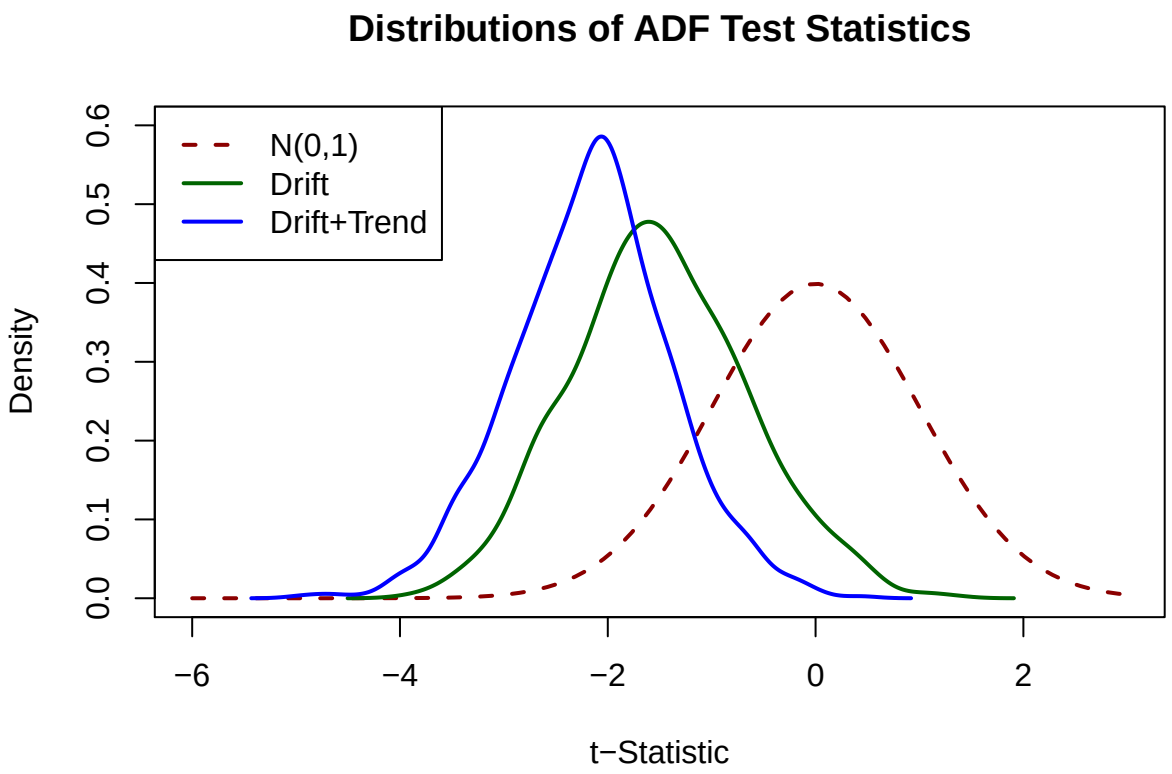
Table 15.1: Large Sample Critical Values of ADF Test

Deterministic Regressors	10%	5%	1%
Intercept only	-2.57	-2.86	-3.43
Intercept and time trend	-3.12	-3.41	-3.96

The results show that using standard normal critical values might be fatal: the 5% critical value of the standard normal distribution is -1.64 but for the Dickey-Fuller distributions the estimated critical values are -2.87 (drift) and -3.43 (drift and linear time trend). This implies that a true null hypothesis (the series has a stochastic trend) would be rejected far to often if the inappropriate normal critical values were used.

We may use the simulated test statistics for a graphical comparison of the standard normal density and (estimates of) both Dickey-Fuller densities.

```
# plot standard normal density
curve(dnorm(x),
      from = -6, to = 3,
      ylim = c(0,0.6),
      lty = 2,
      ylab = "Density",
      xlab = "t-Statistic",
      main = "Distributions of ADF Test Statistics",
      col = "darkred",
      lwd = 2)

# plot density estimates of both Dickey-Fuller distributions
lines(density(ADFD), lwd = 2, col = "darkgreen")
lines(density(ADFDT), lwd = 2, col = "blue")

# add a legend
legend("topleft",
       c("N(0,1)","Drift","Drift+Trend"),
       col = c("darkred","darkgreen","blue"),
       lty = c(2,1,1),
       lwd = 2
       )
```



The deviations from the standard normal distribution are significant: both Dickey-Fuller distributions are skewed to the left and have a havier left tail.

Does U.S. GDP Have a Unit Root?

As an empirical example, we use the ADF test to assess whether there is a stochastic trend in U.S. GDP using the regression

$$\Delta \log(GDP_t) = \beta_0 + \alpha t + \beta_1 \log(GDP_{t-1}) + \beta_2 \Delta \log(GDP_{t-1}) + \beta_3 \Delta \log(GDP_{t-2}) + u_t.$$

```
# log GDP series
LogGDP <- ts(log(GDP["1962::2012"]))

# estimate model
coeftest(
  dynlm(diff(LogGDP) ~ trend(LogGDP, scale = F) + L(LogGDP) + diff(L(LogGDP)) + diff(L(LogGDP),2))
  )
```

```
##
## t test of coefficients:
##
##                             Estimate  Std. Error t value Pr(>|t|)
## (Intercept)               0.27877045  0.11793233  2.3638 0.019066 *
## trend(LogGDP, scale = F)  0.00023818  0.00011090  2.1476 0.032970 *
## L(LogGDP)                -0.03332452  0.01441436 -2.3119 0.021822 *
## diff(L(LogGDP))           0.08317976  0.11295542  0.7364 0.462371
## diff(L(LogGDP), 2)        0.18763384  0.07055574  2.6594 0.008476 **
## ---
```

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

The estimation yields

$$\begin{aligned}\Delta \log(GDP_t) = \underset{(0.118)}{0.28} + \underset{(0.0001)}{0.0002}t + \underset{(0.014)}{-0.033}\log(GDP_{t-1}) \\ + \underset{(0.113)}{0.083}\Delta\log(GDP_{t-1}) + \underset{(0.071)}{0.188}\Delta\log(GDP_{t-2}) + u_t,\end{aligned}$$

so the ADF test statistic is $t = -0.033/0.014 = -2.35$. The corresponding 5% critical value from table 15.1 is -3.41 so we cannot reject the null hypothesis that $\log(GDP$ =has a stochastic trend in favour of the alternative that it is stationary around a deterministic linear time trend.

15.8 Nonstationarity II: Breaks

When there are discrete (at a distinct date) or gradual changes (over time) in the population regression ceofficients, the series is nonstationary. These changes are called *breaks*. There is a variety of reasons why breaks can occur in macroeconomic time series but most often they are related to changes in economic policy or major changes in the structure of the economy. See Chapter 14.7 for a discussion of examples.

If breaks are not accounted for in the regression model, OLS estimates will reflect the average relationship. Since these estimates might be strongly misleading and result in poor forecast quality, we are interested in testing for breaks. One distinguishes between testing for a break when the date is known and testing for a break with an unknown break data.

Let τ denote a known break date and let $D_t(\tau)$ be a binary variable indicating time periods before and after the break. Incorporating the break in an ADL(1,1) regression model yields

$$\begin{aligned}Y_t =& \beta_0 + \beta_1 Y_{t-1} + \delta_1 X_{t-1} + \gamma_0 D_t(\tau) \left[D_t(\tau) Y_{t-1}\right] \\ &+ \gamma_2 \left[D_t(\tau) X_{t-1}\right] + u_t\end{aligned}$$

where we allows for discrete changes in β_0, β_1 and β_2 at the break date τ. The null hypothesis of no break,

$$H_0 : \gamma_0 = \gamma_1 = \gamma_2 = 0,$$

can be tested against the alternative that at least one of the γ's is not zero using an F-Test. This idea is called a Chow test after Gregory Chow (1960).

When the break date is unknown the *Quandt likelihood ratio* (QLR) test (Quandt, 1960) may be used. This is a modified version of the Chow test which uses the largest of all F-statistic obtained for application of the Chow test for all possible break dates in a predetermined range $[\tau_0, \tau_1]$. The QLR test is summarized in Key Concept 14.9.

Key Concept 14.9

The QLR Test for Coefficient Stability

The QLR test can be used to test for a break in the population regression function if the date of the break is unknown. The QLR test statistic is the largest (Chow) $F(\tau)$-statistic computed over a range of eligible break dates $\tau_0 \leq \tau \leq \tau_1$:

$$QLR = \max\left[F(\tau_0), F(\tau_0 + 1), \ldots, F(\tau_1)\right] \tag{15.14}$$

The most important properties are:

- The QLR test also be applied to test whether a subset of the coefficients in the population regression function breaks but the test also rejects if there is a slow evolution of the regression function.
- When there is a single discrete break in the population regression function, the QLR test statistic is $F(\widehat{\tau})$ and $\widehat{\tau}/T$ is a consistent estimator of the true break date.
- The large-sample distribution of QLR depends on q, the number of restrictions beeing tested and both ratios of end points to the sample size, $\tau_0/T, \tau_1/T$.
- Similar as the for the ADF test, the large-sample distribution of QLR is nonstandard. Cricital values are presented in Table 14.5 of the book.

Has the Predictive Power of the term spread been stable?

Using the QLR statistic we may test whether there is a break in the coefficients of terms spread in (15.6), the ADL(2,2) regression model of GDP growth. Following Key Concept 14.9 we modify the specification of (15.6) by adding a break dummy $D(\tau)$ and its interactions with both lags of term spread and choose the range of break points to be tested as 1970:Q1 - 2005:Q2 (these periods are the center 70% of the sample data from 1962:Q2 - 2012:Q4). Thus, the model becomes

$$\begin{aligned} GDPGR_t = &\beta_0 + \beta_1 GDPGR_{t-1} + \beta_2 GDPGR_{t-2} \\ &+ \beta_3 TSpread_{t-1} + \beta_4 TSpread_{t-2} \\ &+ \gamma_1 D(\tau) + \gamma_2 (D(\tau) \cdot TSpread_{t-1}) \\ &+ \gamma_3 (D(\tau) \cdot TSpread_{t-2}) \\ &+ u_t. \end{aligned}$$

Next, we estimate the model for each break point and compute the F-statistic corresponding to the null hypothesis $H_0 : \gamma_1 = \gamma_2 = \gamma_3 = 0$. The QLR-statistic is the largest of the F-statistics obtained in this manner.

```
# set up a range of possible break dates
tau <- seq(1970, 2005, 0.25)

# initialize vector of F-statistics
Fstats <- numeric(length(tau))

# estimation loop over break dates
for(i in 1:length(tau)) {

  # set up dummy variable
  D <- time(GDPGrowth_ts) > tau[i]

  # estimate ADL(2,2) model with intercations
  test <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) +
                  D*L(TSpread_ts) + D*L(TSpread_ts, 2),
                start = c(1962, 1),
                end = c(2012, 4))

  # compute and save F-statistic
  Fstats[i] <- linearHypothesis(test,
                                c("DTRUE=0", "DTRUE:L(TSpread_ts)", "DTRUE:L(TSpread_ts, 2)"),
                                vcov. = sandwich)$F[2]
```

```
}
```

We determine the QLR statistic using `max()`.

```
# identify QLR statistic
QLR <- max(Fstats)
QLR
```

```
## [1] 6.651156
```

It is straightforward to check that the QLR-statistic is the F-statistic obtained for the regression where 1980:Q4 is chosen as the break date.

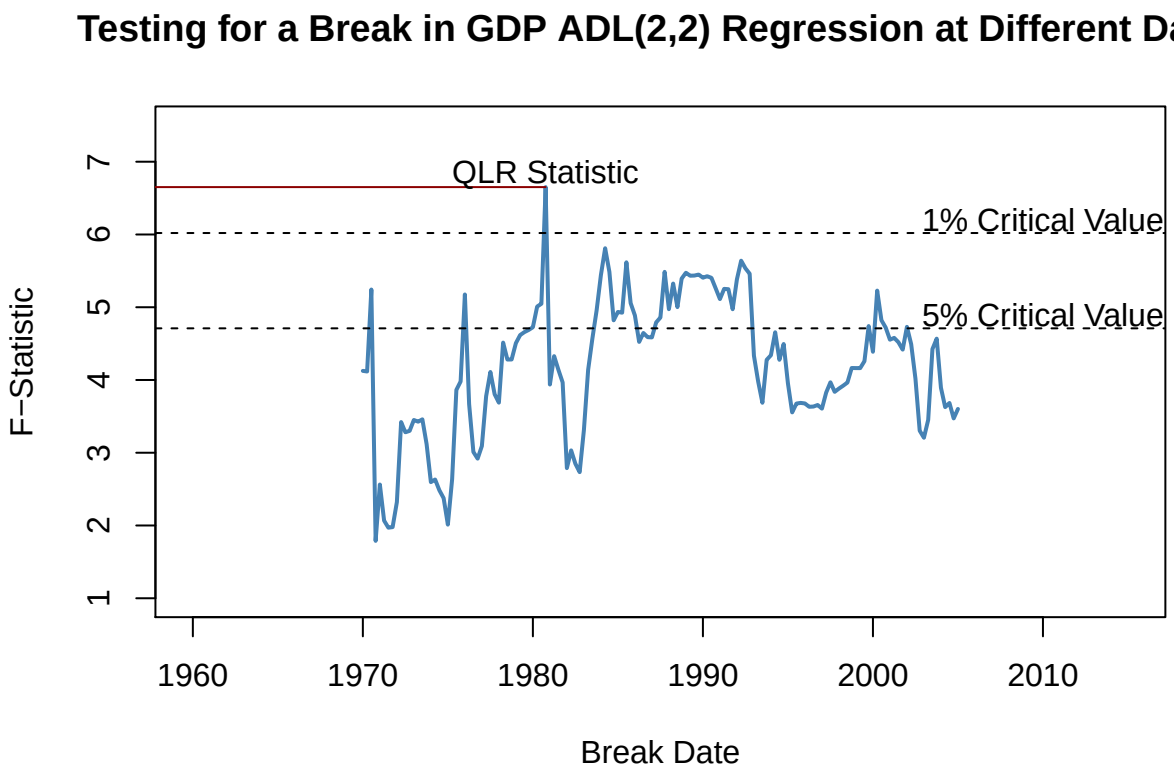
```
# identify time period where QLR-statistic is observed
as.yearqtr(
  tau[which.max(Fstats)]
)
```

```
## [1] "1980 Q4"
```

Since $q = 3$ hypotheses are tested and the central 70% of the sample are considered to contain breaks, the corresponding 1% critical value of the QLR test is 6.02. We reject the null hypothesis that all coefficients (the coefficients on both lags of term spread and the intercept) are stable since the computed QLR-statistic exceeds this threshold. Thus evidence from the QLR test suggests that there is a break in the ADL(2,2) model of GDP growth in the early 1980s.

To reproduce Figure 14.5 of the book, we convert the vector of sequential break-point F-statistics into a time series object and then generate a simple plot with some annotations.

```
# series of F-statistics
Fstatsseries <- ts(Fstats,
                   start = tau[1],
                   end = tau[length(tau)],
                   frequency = 4)

# plot the F-statistics
plot(Fstatsseries,
     xlim = c(1960, 2015),
     ylim = c(1, 7.5),
     lwd = 2,
     col = "steelblue",
     ylab = "F-Statistic",
     xlab = "Break Date",
     main = "Testing for a Break in GDP ADL(2,2) Regression at Different Dates"
     )

# dashed horizontal lines for critical values and QLR statistic
abline(h = 4.71, lty = 2)
abline(h = 6.02, lty = 2)
segments(0, QLR, 1980.75, QLR, col = "darkred")
text(2010, 6.2, "1% Critical Value")
text(2010, 4.9, "5% Critical Value")
text(1980.75, QLR+0.2, "QLR Statistic")
```



Pseudo Out-of-Sample Forecasting

Pseudo out-of-sample forecasts are used to simulate the out-of-sample performance (the real time forecast performance) of a time series regression model. In particular, pseudo out-of-sample forecast allows estimation of the $RMSFE$ of the model and enable researchers to compare different model specifications with respect to their reliability. Key Concept 14.10 summarizes the idea.

Key Concept 14.10

Pseudo Out-of-Sample Forecasting

1. Divide the sample data into $s = T - P$ and P subsequent overservations. The P observations are used as pseudo-out-of-sample observations.
2. Estimate the model using the first s observations.
3. Compute the pseudo-forecast $\widetilde{Y}_{s+1|s}$.
4. Compute the pseudo-forecast-error $\widetilde{u}_{s+1} = Y_{s+1} - \widetilde{Y}_{s+1|s}$.
5. Repeat stepts 2 trough 4 for all reamaining pseudo-out-of-sample dates.

Did the Predictive Power of the Term Spread Change During the 2000s?

The insight gained in the previous section gives reason to presume that the pseudo-out-of-sample performance of ADL(2,2) models estimated using data after the break in the early 1980s should not deteriorate: provided that the coefficients of the population regression function are stable after the potential break in 1980:Q4, these models should have good predictive power. We check this by computing pseudo-out-of-sample forecasts for the period 2003:Q1 - 2012:Q4, a range covering 40 periods, where the forecast for 2003:Q1 is done using data from 1981:Q1 - 2002:Q4, the forecast for 2003:Q2 is based on data frome 1981:Q1 - 2003:Q1 and so on.

Similarly as for the QLR-test we use a `for()` loop for estimation of all 40 models and gather their SERs and the obtained forecasts in a vector which is then used to compute pseudo-out-of-sample forecast errors.

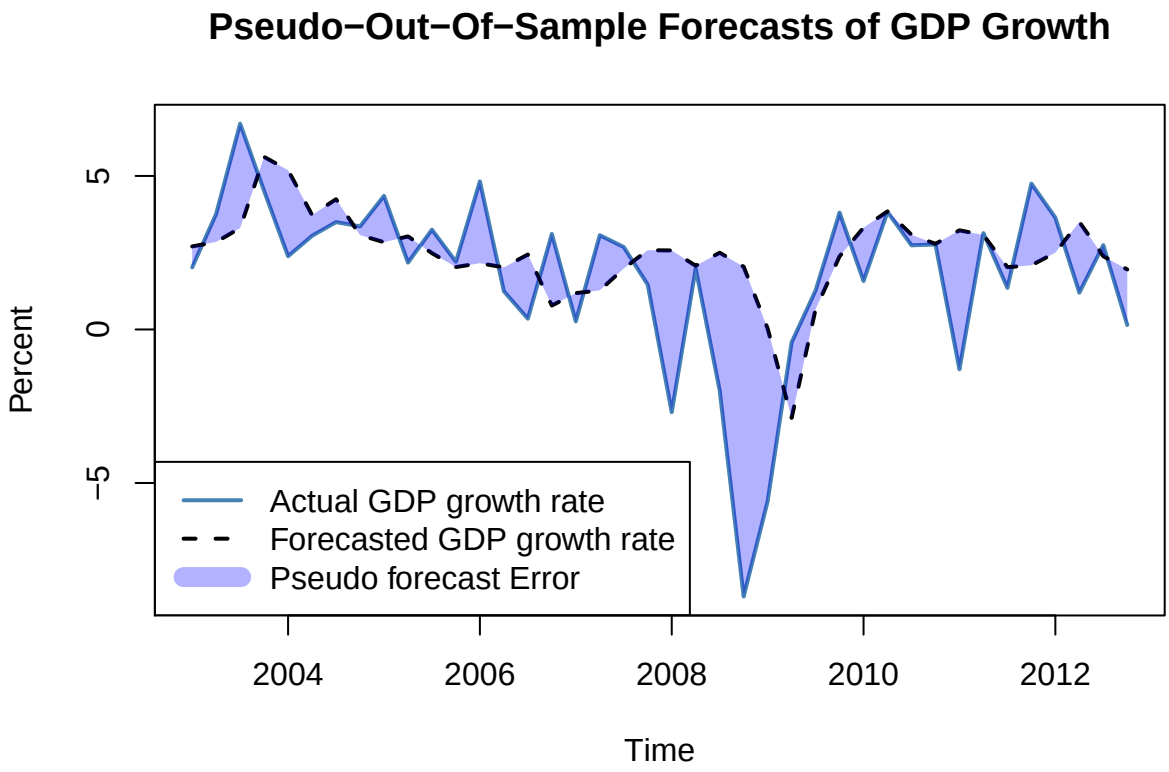
```
# end of sample dates
EndOfSample <- seq(2002.75, 2012.5, 0.25)

forecasts <- numeric(length(EndOfSample))
SER <- forecasts

# estimation loop over end of sample dates
for(i in 1:length(EndOfSample)) {

  # estimate ADL(2,2) model
  m <- dynlm(GDPGrowth_ts ~ L(GDPGrowth_ts) + L(GDPGrowth_ts, 2) + L(TSpread_ts) + L(TSpread_ts, 2),
             start = c(1981, 1),
             end = EndOfSample[i])

  SER[i] <- summary(m)$sigma

  # sample data for one-period ahead forecast
  t <- window(ADLdata, EndOfSample[i]-0.25, EndOfSample[i])

  # compute forecast
  forecasts[i] <- coef(m) %*% c(1, t[1,1], t[2,1], t[1,2], t[2,2])
}
```

```
# compute psuedo-out-of-sample forecast errors
POOSFCE <- window(GDPGrowth_ts, c(2003,1), c(2012,4)) - forecasts
```

We translate the pseudo-out-of-sample forecasts into an object of class ts and plot the real GDP growth rate against the forecasted series. Using

```
# series of pseudo-out-of-sample forecasts
PSOSSFc <- ts(forecasts,
              start = 2003,
              end = 2012.75,
              frequency = 4)

# plot GDP growth time series
plot(window(GDPGrowth_ts, c(2003,1), c(2012,4)),
     col = "steelblue",
     lwd = 2,
     ylab = "Percent",
     main = "Pseudo-Out-Of-Sample Forecasts of GDP Growth"
     )

# add series of pseudo-out-of-sample forecasts
lines(PSOSSFc,
      lwd = 2,
      lty = 2
      )

# shade area between curves (the pseudo forecast error)
polygon(c(time(PSOSSFc), rev(time(PSOSSFc))),
        c(window(GDPGrowth_ts, c(2003,1), c(2012,4)), rev(PSOSSFc)),
        col = alpha("blue", alpha = 0.3),
```

```
        border = NA
        )

# add a legend
legend("bottomleft",
       lty = c(1, 2, 1),
       lwd = c(2,2,10),
       col = c("steelblue","black",alpha("blue", alpha = 0.3)),
       legend = c("Actual GDP growth rate",
         "Forecasted GDP growth rate",
         "Pseudo forecast Error")
       )
```



Apparently, the pseudo forecasts track the actual GDP growth rate quite well, except for the kink in 2009 which can be attributed to the recent financial crisis.

The SER of the first model (estimated using data from 1981:Q1 to 2002:Q4) is 2.39 so based on the in sample fit we would expect out of sample forecast errors to have mean zero and a mean squared forecast error of about 2.39.

```
# SER of ADL(2,2) mode using data from 1981:Q1 - 2002:Q4
SER[1]
```

```
## [1] 2.389773
```

The root mean squared forecast error of the pseudo-out-of-sample forecasts is somewhat larger.

```
# compute root mean squared forecast error
sd(POOSFCE)
```

```
## [1] 2.667612
```

An interesting hypothesis is whether the mean forecast error, is zero that is the ADL(2,2) forecasts are right,

on average. This hypothesis is easily tested using the function `t.test()`.

```
# test if mean forecast error is zero
t.test(POOSFCE)
```

```
##
##  One Sample t-test
##
## data:  POOSFCE
## t = -1.5523, df = 39, p-value = 0.1287
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -1.5078876  0.1984001
## sample estimates:
##  mean of x
## -0.6547438
```

The hypothesis cannot be rejected at the 10% significance level. Alltogther the analysis suggests that the ADL(2,2) model coefficients have been stable since the presumed break in the early 1980s.

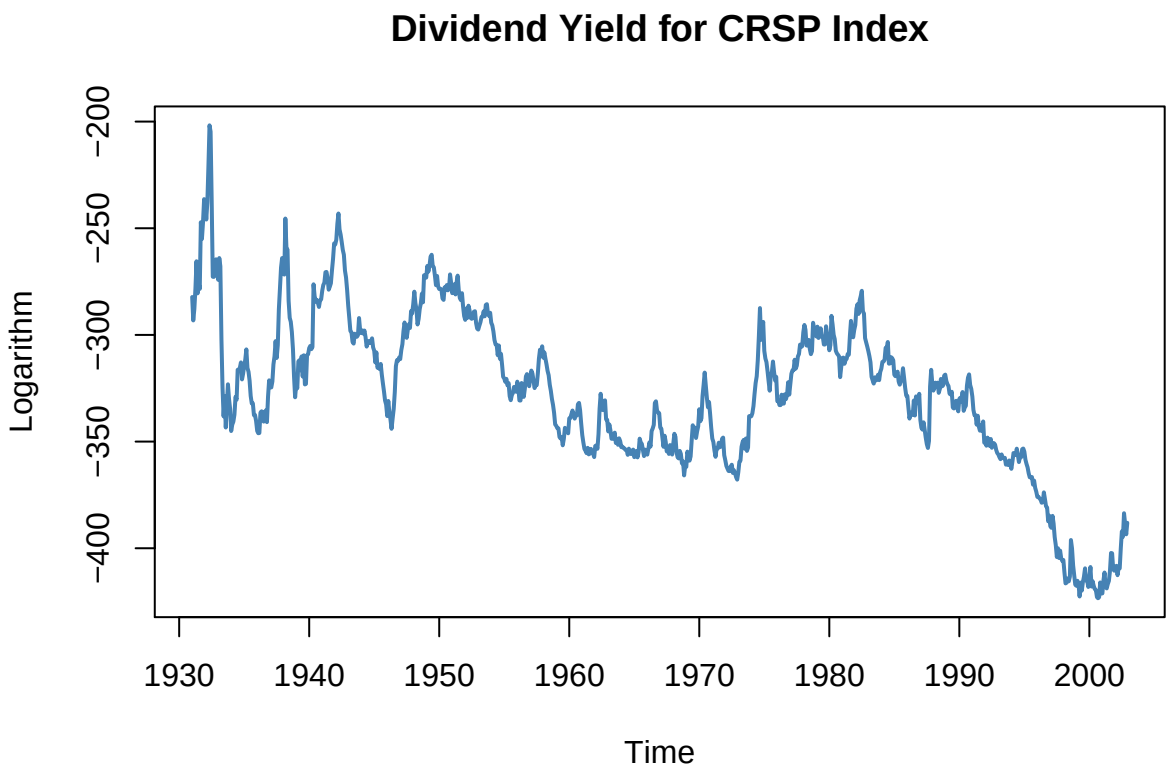
15.9 Can You Beat the Market? Part II

The dividend yield — the ratio of current dividens to the stock price — can be considered as an indicator of future dividends: if a stock has a high current dividend yield, it can be considered undervalued and it can be presumed that the price of the stock goes up in the future, meaning that future excess returns go up.

This presumption can be examined using ADL models of exess returns where lags of the logarithm of the stock's dividend yield serve as additional regressors.

Unfortunatly, a graphical inspection of the time series of the logarithm of the dividend yield casts doubt on the assumption that the series is stationary which, as has been discussed in Chapter 14.7, is necessary to obtain meaningful results in a regression analysis.

```
# plot logarithm of dividend yield series
plot(StockReturns[,2],
     col = "steelblue",
     lwd = 2,
     ylab = "Logarithm",
     main = "Dividend Yield for CRSP Index")
```



The Dickey-Fuller test statistic for an autoregressive unit root in an AR(1) model with drift provides further evidence that the series might be nonstationary.

```
# DF-Regression for log dividend yield
LDivYield <- dynlm(d(ln_DivYield) ~ L(ln_DivYield), data = StockReturns,
      start = c(1960,1), end = c(2002,12)
      )

summary(LDivYield)$coef
```

```
##                    Estimate  Std. Error   t value  Pr(>|t|)
## (Intercept)    -2.728786496 2.083152348 -1.309931 0.1908042
## L(ln_DivYield) -0.007656929 0.005998017 -1.276577 0.2023282
```

Since the t-value for the coefficient on the logarithm of the dividend yield is -1.27 so the hypothesis that the true coefficient is zero cannot be rejected, even at the 10% significance level.

However, it is possible to examine whether the dividend yield has predictive power for excess returns by using its differences in an ADL(1,1) and an ADL(2,2) model (differencing a series with a unit root yield a stationary series) though these model specifications do not correspond to the ecnonomic reasoning mentioned above. Thus we also estimate a ADL(1,1) regression using the level of the logarithm of the dividend yield.

Thus we estimate three model specifications:

$$excess\,returns_t = \beta_0 + \beta_1 excess\,returns_{t-1} + \beta_3 \Delta \log(dividendyield_t - 1) + u_t$$

$$excess\,returns_t = \beta_0 + \beta_1 excess\,returns_{t-1} + \beta_2 excess\,returns_{t-2} + \beta_3 \Delta \log(dividendyield_t - 1) + \beta_4 \Delta \log(dividendyield_t - 2) + u_t$$

$$excess\,returns_t = \beta_0 + \beta_1 excess\,returns_{t-1} + \beta_5 \log(dividendyield_t - 1) + u_t$$

```
# ADL(1,1) (1st difference of log dividend yield)
CRSP_ADL_1 <- dynlm(ExReturn ~ L(ExReturn) + d(L(ln_DivYield)), data = StockReturns,
      start = c(1960,1), end = c(2002,12)
      )

# ADL(2,2) (1st & 2nd differences of log dividend yield)
CRSP_ADL_2 <- dynlm(ExReturn ~ L(ExReturn) + L(ExReturn, 2) + d(L(ln_DivYield)) + d(L(ln_DivYield, 2)),
      start = c(1960,1), end = c(2002,12)
      )

# ADL(1,1) (levels of log dividend yield)
CRSP_ADL_3 <- dynlm(ExReturn ~ L(ExReturn) + L(ln_DivYield), data = StockReturns,
      start = c(1960,1), end = c(1992,12)
      )
```

```
rob_se_CRSP_ADL <- list(
  sqrt(diag(sandwich(CRSP_ADL_1))),
  sqrt(diag(sandwich(CRSP_ADL_2))),
  sqrt(diag(sandwich(CRSP_ADL_3)))
)
```

ADL Models of Monthly Excess Stock Returns

Excess returns on the CSRP value-weighted index

ADL(1,1)

ADL(2,2)

ADL(1,1)

(1)

(2)

(3)

excess returnt-1

0.059

0.042

0.078

(0.158)

(0.162)

(0.057)

excess returnt-2

-0.213

(0.193)

1st diff log(dividend yieldt-1)

0.009

-0.012

(0.157)

(0.163)

1st diff log(dividend yieldt-2)

-0.161

(0.185)

log(dividend yieldt-1)

0.026**

(0.012)

Constant

0.309

0.372*

8.987**

(0.199)

(0.208)

(3.912)

Observations

516

516

396

R2

0.003

0.007

0.018

Adjusted R2

-0.001

-0.001

0.013

Residual Std. Error

4.338 (df = 513)

4.337 (df = 511)

4.407 (df = 393)

F Statistic

0.653 (df = 2; 513)

0.897 (df = 4; 511)

3.683** (df = 2; 393)

Note:

p<0.1; ***p<0.05;*** p<0.01

For models (1) and (2), notice that none of the individual t-statistics suggests that the coefficients are different from zero. Also we cannot reject the hypothesis that none of the lags have predictive power for excess returns at any common level of significance (an F-test that the lags have predictive power is reject for both models).

Things are different for model (3). The coefficient on the level of the logarithm of the devidend yield is different from zero at the 5% level and the F-test does not reject at this level either. But we should be suspicious: the high degree of persistence in the devidend yield series probably renders this inference useless because t- and F-statistics may follow distributions that deviate considerably from their theoretical large-sample distributions such that the usual critical values cannot be applied.

If model (3) would be of any use for predicting excess returns, pseudo-out-of-sample forecasts based on (3) should outperform forecasts of an intercept-only model in terms of lower sample RMSFE. We can perform this type of comparison using R code in the fashion of the applications of Chapter 14.8.

```
# end of sample dates
EndOfSample <- as.numeric(window(time(StockReturns), c(1992,12), c(2002,11)))

# initialize matrix  forecasts
forecasts <- matrix(nrow = 2,
                    ncol = length(EndOfSample))

# estimation loop over end of sample dates
for(i in 1:length(EndOfSample)) {

  # estimate model (3)
  mod3 <- dynlm(ExReturn ~ L(ExReturn) + L(ln_DivYield), data = StockReturns,
                start = c(1960, 1),
                end = EndOfSample[i])

  # estimate intercept only model
  modconst <- dynlm(ExReturn ~ 1, data = StockReturns,
                start = c(1960, 1),
                end = EndOfSample[i])

  # sample data for one-period ahead forecast
  t <- window(StockReturns, EndOfSample[i], EndOfSample[i])

  # compute forecast
  forecasts[,i] <- c(
    coef(mod3) %*% c(1, t[1], t[2]),
    coef(modconst)
  )

}
```

```
# gather data
d <- cbind(
  "Excess Returns" = c(window(StockReturns[,1], c(1993, 1), c(2002, 12))),
  "Model (3)" = forecasts[1,],
  "Intercept Only" = forecasts[2,],
  "Always Zero" =  0)

# Compute RMSFEs
c(
  "ADL model (3)" = sd(d[,1] - d[,2]),
  "Intercept-only model" = sd(d[,1] - d[,3]),
  "Always zero" = sd(d[,1] - d[,4])
)
```

```
##        ADL model (3) Intercept-only model          Always zero
##             4.043757             4.000221            3.995428
```

The comparison indicates that model (3) has no predictive power since it is outperformed in terms of sample RMSFE by the intercept-only model. A model forecasting excess returns always by zero has an even lower sample RMSFE. This finding is consistent with the strong-form efficiency hypothesis which states that *all* publicly available information is accounted for in stock prices such that there is no way to predict future stock prices or excess returns, implying that the perceived significant relationship indicated by model (3) is wrong.

Summary

Chapter 14 dealt with introductory topics in time series regression analysis where variables are generally correlated from one observation to the next, a concept termed serial correlation. At the beginning, several ways of storing and plotting time series data using R have been presented and used for informal analysis of economic data.

We have introduced AR and ADL models and applied them in the context of forecasting of macroeconomic and financial time series using R. The discussion also included the topic of lag length selection. It was shown how to set up a simple function that computes the BIC for a model object supplied.

We have also seen how to write simple R code for performing and evaluating forecasts and demonstrated some more sophisticated approaches to conduct pseudo-out-of-sample forecasts for assessment of a model's predictive power for unobserved future outcomes of a series, to check model stability and to compare different models.

Furthermore, some more technical aspects like the concept of stationarity were adressed. This included applications to testing for an autoregressive unit root with the Dickey-Fuller test and the detection of a break in the population regression function using the QLR statistic. For both approaches, the distribution of the relevant test statistic is nonnormal, even in large samples. Concerning the Dickey-Fuller test we have used R's random number generation facilities to produce evidence for this issue by means of Monte-Carlo simulation and motivated usage of tabulated quantiles.

Also, empirical studies on the validity of the weak and the strong form efficiency hypothesis which are presented in the applications *Can You Beat the Market? Part I & II* in the book have been reproduced using R.

In all applications presented in this chapter, the focus was layed on forecasting future outcomes rather than estimation of causal relationships between time series variables. However, the framework of methods needed for the latter is very similar. Chapter 15 is devoted to estimation of so called *dynamic causal effects*.

Chapter 16

Estimation of Dynamic Causal Effects

For answering some questions in economic research it interesting to know how large the effect on Y now and in the future of a change in X is. This called the *dynamic causal effect* on Y of a change in X. In this section we will see how to estimate dynamic causal effects using several R applications where we investigate the dynamic effect of cold weather in Florida on the price of orange juice concentrate.

The discussion covers:

- Estimation of distributed lag models
- Heteroskedasticity- and autocorrelation-consistent (HAC) standard errors
- Generalized least squares (GLS) of ADL models

If you would like to reproduce code examples, install the R packages listed below and make sure that the subsequent code chunk executes without any errors.

```
library(AER)
library(quantmod)
library(dynlm)
library(orcutt)
library(nlme)
library(stargazer)
```

16.1 The Orange Juice Data

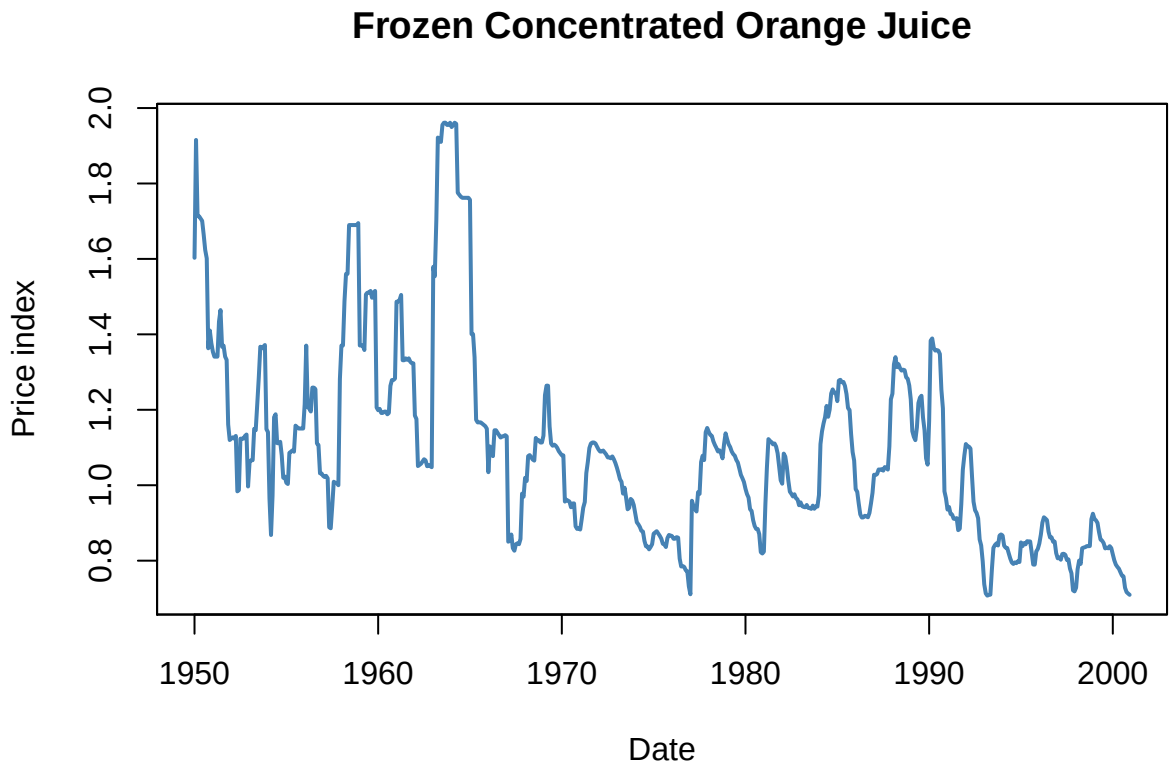
The largest cultivation region for oranges in the U.S. is located in state of Florida which usually has ideal climate for the fruit growth and thus makes Florida the source of almost all frozen juice contentrate produced in the country However, from time to time and depening on their severeness, cold snaps cause loss of harvests such that the supply of oranges decreases and consequently the price of frozen juice conzentrate rises. The timing of the price increases is complicated: a cut in today's supply of concentrate influences not today's price but also future prices because supply in future periods will be lower, too. Clearly, the magnitude of today's and future price increases due to a freeze is an empirical question that can be investigated using a distributed lag model — a model that relates price changes to weather conditions.

To begin with the analysis, we reproduce Figure 15.1 of the book which displays plots of the price index for frozen concentrated orange juice, percent changes in the price as well as monthly freezing degree days in Orlando, the center of Florida's orange-growing region.

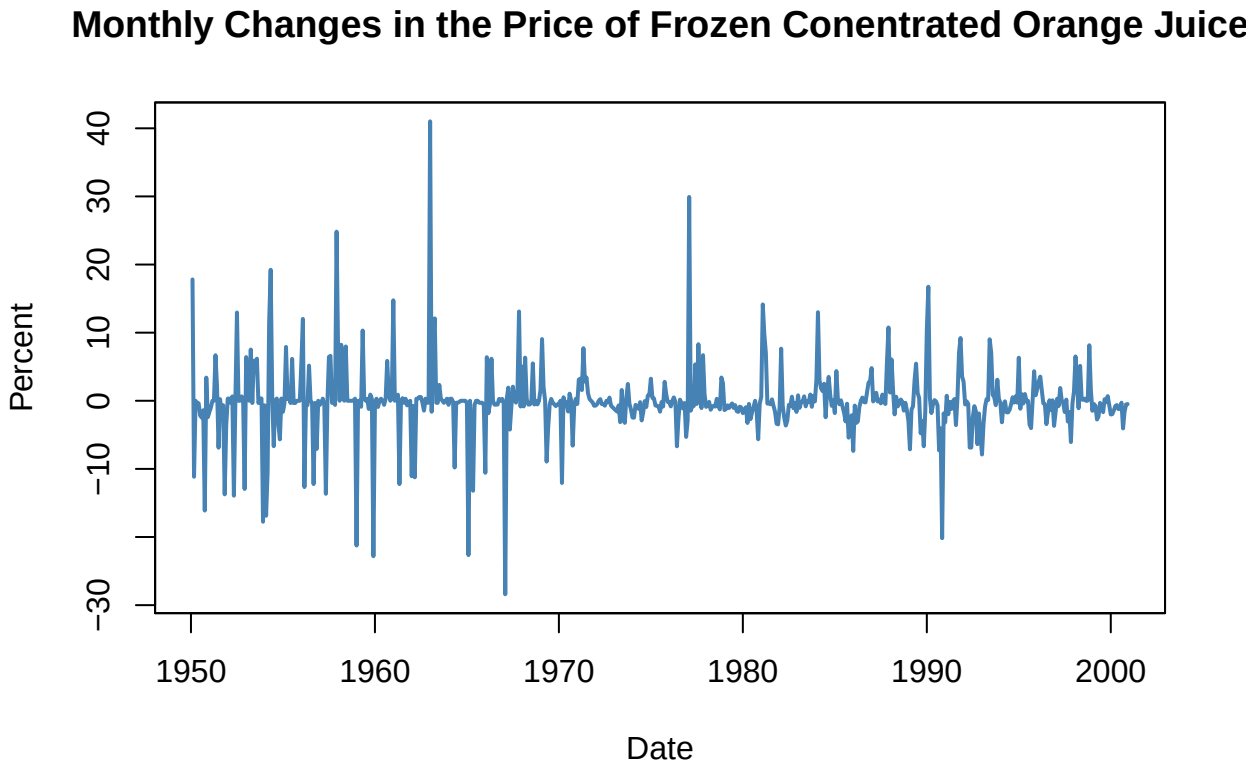
```
# Load the frozen orange juice data set
data("FrozenJuice")
```

```
# price index for frozen concentrated juice
FOJCPI <- FrozenJuice[,"price"]/FrozenJuice[,"ppi"]
FOJC_pctc <- 100 * diff(log(FOJCPI))
FDD <- FrozenJuice[, "fdd"]
```

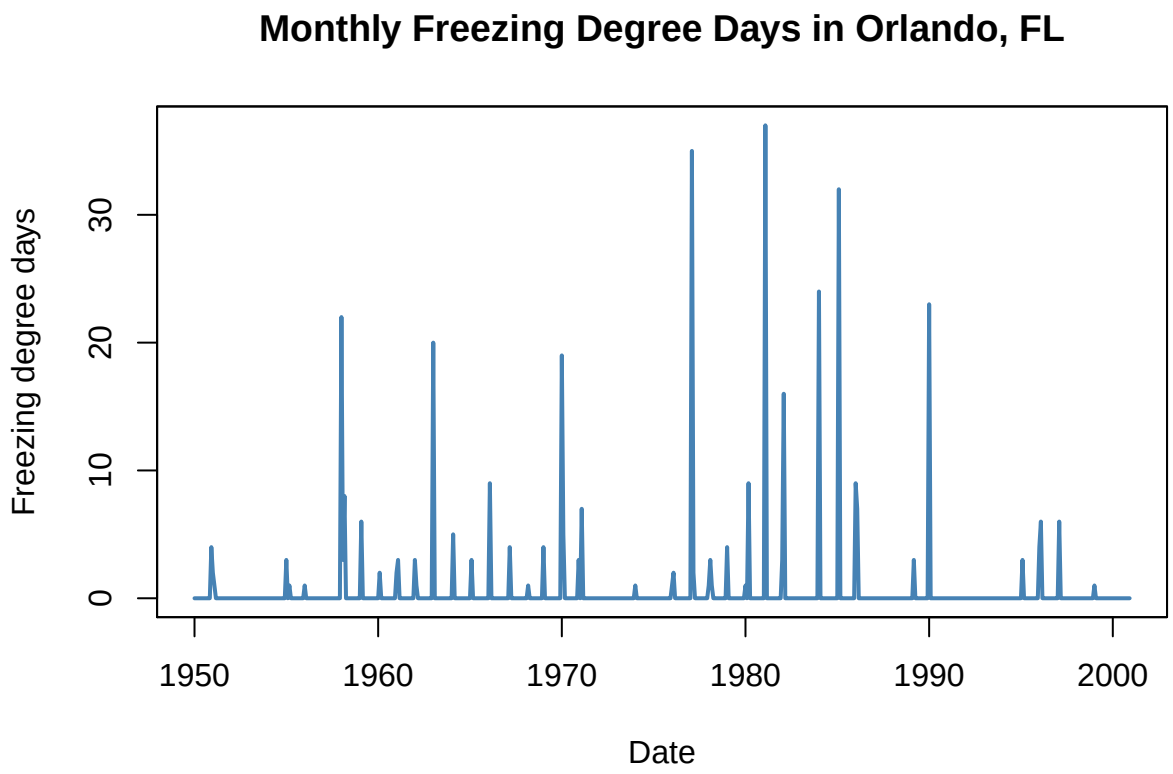
```
FOJCPI_xts <- as.xts(FOJCPI)
FDD_xts <- as.xts(FrozenJuice[,3])

# Plot orange juice price index
plot(as.zoo(FOJCPI),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Price index",
     main = "Frozen Concentrated Orange Juice"
)
```




```
# Plot percent changes in prices
plot(as.zoo(FOJC_pctc),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Percent",
     main = "Monthly Changes in the Price of Frozen Conentrated Orange Juice"
)
```




```
# plot freezing degree days
plot(as.zoo(FDD),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Freezing degree days",
     main = "Monthly Freezing Degree Days in Orlando, FL"
)
```



It is conspicious that periods with a high amount of freezing degree days are followed by large month-to-month price changes. These coindciding movements give rise to run a simple regression of price changes $\%ChgOJC_t$ on freezing degree days (FDD_t) to estimate the effect of an additional freezing degree day one the price in the current month. For this, as for all other regressions in this chapter, we use $T = 611$ observations (January 1950 to December 200).

```
# simple regression of percent changes on freezing degree days
orange_SR <- dynlm(FOJC_pctc ~ FDD)
coeftest(orange_SR, vcov. = vcovHAC)
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.42095    0.18683 -2.2531 0.0246064 *
## FDD          0.46724    0.13385  3.4906 0.0005167 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the standard errors are computed using a HAC estimator of the model variance-covariance matrix, a matter that will not be further commented at this point, see Chapter 14.5.

$$\widehat{\%ChgOJC_t} = \underset{(0.19)}{-0.42} + \underset{(0.13)}{0.47}\, FDD_t$$

The estimated coefficient on FDD_t has the following interpretation: an additional freezing degree day in month t leads to a price increase 0f 0.47% in the same month.

To consider effects of cold snaps on the orange juice price over following, we include lagged values of FDD_t in our model which leads to a so-called *distributed lag regression*. We estimate a specification of a contemporeneous and six lagged values of FDD_t.

```
# distributed lag model with 6 lags of freezing degree days
orange_DLM <- dynlm(FOJC_pctc ~ FDD + L(FDD, 1:6))
coeftest(orange_DLM, vcov. = vcovHAC)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -0.692961   0.212445 -3.2618 0.0011700 **
## FDD           0.471433   0.135195  3.4871 0.0005242 ***
## L(FDD, 1:6)1  0.145021   0.081557  1.7782 0.0758853 .
## L(FDD, 1:6)2  0.058364   0.058911  0.9907 0.3222318
## L(FDD, 1:6)3  0.074166   0.047143  1.5732 0.1162007
## L(FDD, 1:6)4  0.036304   0.029335  1.2376 0.2163670
## L(FDD, 1:6)5  0.048756   0.031370  1.5543 0.1206535
## L(FDD, 1:6)6  0.050246   0.045129  1.1134 0.2659919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As a result we obtain

$$\widehat{\%ChgOJC_t} = \underset{(0.21)}{-0.69} + \underset{(0.14)}{0.47}\,FDD_t + \underset{(0.08)}{0.15}\,FDD_{t-1} + \underset{(0.06)}{0.06}\,FDD_{t-2} + \underset{(0.05)}{0.07}\,FDD_{t-3} \tag{16.1}$$

$$+ \underset{(0.03)}{0.04}\,FDD_{t-4} + \underset{(0.03)}{0.05}\,FDD_{t-5} + \underset{(0.05)}{0.05}\,FDD_{t-6} \tag{16.2}$$

where the coefficient on FDD_{t-1} estimates the price increase in period t caused by an additional freezing degree day in the preceding month, the coefficient on FDD_{t-2} estimates the effect of an additional freezing degree day two month ago and so on. Consequently, the coefficients in (16.2) can be interpreted as price changes in current and future periods due to a unit increase in the current month' freezing degree days.

16.2 Dynamic Causal Effects

This section of the book describes the general idea of a dynamic causal effect and how the concept of a randomized controlled experiment which (as has been discussed in Chapter 13) constitutes and ideal study design can be translated to time series applications, using several examples. For brevity we will not go into the details but note once again that the distributed lag model mentionted above can often be used to estimate a dynamic causal relationship.

In general, for empirical attempts to measure a dynamic causal effect, the assumptions of stationarity (see Key Concept 14.5) and exogeneity must hold. In the time series appliaction up until here we have assumed that the model error term has conditional mean zero given current and past values of the regressors. For estimation of a dynamic causal effect using a distributed lag model, assuming a stronger form termed *strict exogeneity* may be useful. Strict exogeneity states that the error term has mean zero conditional on past, present *and future* values of the independent variables.

The two concepts of exogeneity and the distributive lag model are summarized in Key Concept 15.1

Key Concept 15.1

The Distributed Lag Model and Exogeneity

The general distributed lag model is

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \beta_3 X_{t-2} + \cdots + \beta_{r+1} X_{t-r} + u_t \tag{16.3}$$

where it is assumed that

1. X is an exogenous variable,
$$E(u_t | X_t, X_{t-1}, X_{t-2}, \dots) = 0.$$
2. (a) X_t, Y_t have a stationary distribution.
(b) (Y_t, X_t) and (Y_{t-j}, X_{t-j}) become independently distributed as j gets large.
3. Large outliers are unlikely. In particular, we need that the all variables have more than eight nonzero and finite moments — a stronger assumption than before (four finite nonzero moments) that is required for computation of the HAC covariance matrix estimator.
4. There is no perfect multicollinearity.

The distributed lag model may be extended to include contempoarneous and past values of additional regressors.

On the assumption of exogeneity

- There is another form of exogeneity termed *strict exogeneity* which assumes
$$E(u_t | \dots X_{t+2}, X_{t+1}, X_t, X_{t-1}, X_{t-2}, \dots) = 0,$$
that is the error term is has mean zero conditional on past, present and future values of X. Strict exogeneity implies exogeneity (as defined in 1.) but not the other way around. From this point we will therefore distinguish between exogeneity and strict exogeneity.
- Exogeneity as in 1. suffices for OLS estimators of the coefficient in distributed lag models to be consistent. However, if the the assumption of strict exogeneity can be made, more efficient estimators can be applied.

16.3 Dynamic Multipliers and Cumulative Dynamic Multipliers

The following terminoligy regarding the coefficients in the distributed lag model (16.3) are useful for upcoming applications:

- The dynamic causal effect is also termed the *dynamic multiplier*. We say that β_{h+1} in (16.3) is the h-period dynamic multiplier.
- The contemporaneous effect of X ond Y, β_1, is termed the *impact effect*.
- The h-period *cumulative dynamic multiplier* of a unit change in X and Y is defined as the cumulative sum of the dynamic multipliers. In particualr, β_1 is the zero-period cumulative dynamic multiplier, $\beta_1 + \beta_2$ is the one-period cumulative dynamic multiplier and so forth.

  Cumulative dynamic multipliers of (16.3) are the coefficients $\delta_1, \delta_2, \dots, \delta_r, \delta_{r+1}$ in the modified regression

$$Y_t = \delta_0 + \delta_1 \Delta X_t + \delta_2 \Delta X_{t-1} + \cdots + \delta_r \Delta X_{t-r+1} + \delta_{r+1} X_{t-r} + u_t. \tag{16.4}$$

  and thus can be directly estimated using OLS wich makes it convenient to compute their HAC standard errors. δ_{r+1} is called the *long-run dynamic multiplier*.

It is straightforward to compute the cumulative dynamic multipliers for (16.2), the estimated distributed lag regression of changes in orange juice concentrate prices on freezing degree days, using the corresponding model object `orange_DLM` and the function `cumsum()`.

```
# compute cumulative multipliers
cum_mult <-cumsum(orange_DLM$coefficients[-1])

# rename entries
names(cum_mult) <- paste(0:6, sep = "-", "period CDM")

cum_mult
```

```
## 0-period CDM 1-period CDM 2-period CDM 3-period CDM 4-period CDM
##    0.4714329    0.6164542    0.6748177    0.7489835    0.7852874
## 5-period CDM 6-period CDM
##    0.8340436    0.8842895
```

Translating the distributed lag model with six lags of FDD to (16.4), we see that the OLS coefficient estimates in this model coincide with the multipliers stored in `cum_mult`.

```
# estimate cumulative dynamic multipliers using the modified regression
cum_mult_reg <-dynlm(FOJC_pctc ~ d(FDD) + d(L(FDD,1:5)) + L(FDD,6))
coef(cum_mult_reg)[-1]
```

```
##          d(FDD) d(L(FDD, 1:5))1 d(L(FDD, 1:5))2 d(L(FDD, 1:5))3
##       0.4714329       0.6164542       0.6748177       0.7489835
## d(L(FDD, 1:5))4 d(L(FDD, 1:5))5       L(FDD, 6)
##       0.7852874       0.8340436       0.8842895
```

As noted above, this spcification of model allows to obtain standard errors for the estimated dynamic cumulative multipliers.

```
coeftest(cum_mult_reg, vcov. = vcovHAC)
```

```
##
## t test of coefficients:
##
##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)     -0.69296    0.23668 -2.9278 0.0035431 **
## d(FDD)           0.47143    0.13583  3.4709 0.0005562 ***
## d(L(FDD, 1:5))1  0.61645    0.13145  4.6896 3.395e-06 ***
## d(L(FDD, 1:5))2  0.67482    0.16009  4.2151 2.882e-05 ***
## d(L(FDD, 1:5))3  0.74898    0.17263  4.3387 1.682e-05 ***
## d(L(FDD, 1:5))4  0.78529    0.17351  4.5260 7.255e-06 ***
## d(L(FDD, 1:5))5  0.83404    0.18236  4.5737 5.827e-06 ***
## L(FDD, 6)        0.88429    0.19303  4.5810 5.634e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.4 HAC Standard Errors

The error term u_t in the distributed lag model (16.3) may be serially correlated due to serially correlated determinants of Y_t that are not included as regressors. When these factors are not correlated with the regressors included in the model, serially correlated errors do not violate the assumption of exogeneity such that the OLS estimator remains unbiased and consistent.

Instead, autocorrelated standard errors render the usual homoskedasticity-only *and* heteroskedasticity-robust standard errrors invalid and may lead to misleading statistical inference. HAC errors are a remedy for this issue. Key Concept 15.2

Key Concept 15.2

HAC Standard errors

Problem:

If the error term u_t in the distributed lag model (16.3) is serially correlated, satistical inference that rests on usual (heteroskedasticity-robust) standard errors can be strongly misleading.

Solution:

Heteroskedasticity- and autocorrelation-consistent (HAC) estimators of the variance-covariance matrix circumvent this issue. There are R functions like `vcovHAC()` from the package sandwich which are convinient for computation of such estimators.

The package sandwich also contains the function `NeweyWest()`, an implementation of the HAC variace-covariance estimator proposed by Newey and West (1987).

Consider the distributed lag regression model with no lags and the single regressor X_t

$$Y_t = \beta_0 + \beta_1 X_t + u_t.$$

with autocorrelated errors. A brief derivation of

$$\widetilde{\sigma}^2_{\widehat{\beta}_1} = \widehat{\sigma}^2_{\widehat{\beta}_1} \widehat{f}_t, \tag{16.5}$$

the so-called *Newey-West variance estimator* for the variance of the OLS estimator of β_1 is presented in Chapter 15.4 of the book. $\widehat{\sigma}^2_{\widehat{\beta}_1}$ in (16.5) is the heteroskedasticity-robust variance estimate of $\widehat{\beta}_1$ and

$$\widehat{f}_t = 1 + 2 \sum_{j=1}^{m-1} \left(\frac{m-j}{m} \right) \widetilde{\rho}_j \tag{16.6}$$

is a correction factor that adjusts for serially correlated errors and involves estimates of $m-1$ autocovariances $\widetilde{\rho}_j$, whereby m is a truncation parameter to be chosen. A rule of thumb for choosing m is

$$m = \left\lceil 0.75 \cdot T^{1/3} \right\rceil. \tag{16.7}$$

In the following we simulate a time series that, as stated above, follows a distributed lag model with autocorrelated errors and then show how to compute the Newey West HAC estimate of $SD(\widehat{\beta}_1)$ using R. This is done in two seperat but, as we will see, identical approches: at first we follow the derivation presented in the book step-by-step and compute the estimate manually. We then prove that the result is exactly the estimate obtained when using the function `NeweyWest()`.

```
# function that computes rho tilde
acf_c <- function(x,j) {
  return(
    t(x[-c(1:j)]) %*% na.omit(Lag(x, j)) / t(x) %*% x
```

```
  )
}

# simulate time series with serially correlated errors
set.seed(1)

eps <- arima.sim(n = 100, model = list(ma = 0.5))
X <- runif(100, 1, 10)
Y <- 0.5 * X + eps

# compute OLS residuals
res <- lm(Y ~ X)$res

# compute v
v <- (X - mean(X)) * res

# compute robust estimate of beta_1 variance
var_beta_hat <- 1/100 * (1/(100-2) * sum((X - mean(X))^2 * res^2) ) /
                        (1/100 * sum((X-mean(X))^2))^2

# rule of thumb truncation parameter
m <- floor(0.75 * 100^(1/3))

# compute correction factor
f_hat_T <- 1 + 2 * sum(
  (m - 1:(m-1))/m * sapply(1:(m-1), function(i) acf_c(x=v, j=i))
  )

# compute Newey West HAC estimate of the standard error
sqrt(var_beta_hat * f_hat_T)
```

```
## [1] 0.04036208
```

By choosing `lag = m-1` it is ensured that the maximum order of autocorrelations used is $m - 1$ — just as in equation (16.6). Notice that we further set the arguments `prewhite = F` and `adjust = T` to ensure that the formula (16.5) is used and finite sample adjustments are made.

```
# Using NeweyWest():
NW_VCOV <- NeweyWest(lm(Y ~ X),
               lag = m - 1, prewhite = F,
               adjust = T
               )

# compute standard error
sqrt(diag(NW_VCOV))[2]
```

```
##          X
## 0.04036208
```

We find that the computed standard errors coincide. Of course, a variance-covariance matrix estimate as computed by `NeweyWest()` can be supplied as the argument `vcov` in `coeftest()` such that HAC t-statistics and p-values are provided.

```
example_mod <- lm(Y ~ X)
coeftest(example_mod, vcov = NW_VCOV)
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.542310   0.235423  2.3036  0.02336 *
## X           0.423305   0.040362 10.4877  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.5 Estimation of Dynamic Causal Effects with Strictly Exogeneous Regressors

In general, the errors in a distributed lag model are correlated which necessitates usage of HAC standard errors for valid inference. If, however, the assumption of exogeneity (the frist assumption stated in Key Concept 15.1) is replaced by strict exogeneity, that is

$$E(u_t|\dots, X_{t+1}, X_t, X_{t-1}, \dots),$$

more efficient approaches than OLS estimation of the coefficients may be available. For a general distributed lag model with r lags and AR(p) errors, these approaches are summarized in Key Concept 15.4.

Key Concept 15.4

Estimation of Dynamic Multipliers Under Strict Exogeneity

Consider the general distributed lag model with r lags and assume that the errors follow an $AR(p)$ process,

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + \cdots + \beta_{r+1} X_{t-r} + u_t \tag{16.8}$$

$$u_t = \phi_1 u_{t-1} + \phi u_{t-2} + \cdots + \phi_p u_{t-p} + \widetilde{u}_t. \tag{16.9}$$

Assuming strict exogeneity of X_t, one may rewrite the above model in the ADL specification

$$\begin{aligned} Y_t = &\alpha_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \\ &+ \delta_0 X_t + \delta_1 X_{t-1} + \cdots + \delta_q X_{t-q} + \widetilde{u}_t \end{aligned}$$

where $q = r + p$ and compute estimates of the dynamic multipliers $\beta_1, \beta_2, \dots, \beta_{r+1}$ using OLS estimates of $\phi_1, \phi_2, \dots, \phi_p, \delta_0, \delta_1, \dots, \delta_q$.

An alternative is to estimate the dynamic multipliers using feasible GLS, that is to apply the OLS estimator to a quasi-differenced specification of (16.9). Under strict exogeneity, the feasible GLS approach is the BLUE estimator for the dynamic multipliers in large samples.

On the one han, as demonstrated in Chapter 15.5 of the book, the OLS estimation of the ADL representation can be beneficial for estimation of dynamic multipliers in large distributed lag model beacause it allows for a more parsimonious model that may be a good approximation to a large distributed lag model. On the other hand, the GLS approach is more efficient than the ADL estimator if the sample size is large.

In what follows we shortly review how different representations of a distribted lag model can be obtained when the causal effect in Y of a change in X lasts for only two periods and show how these specification can be estimated by OLS and GLS using R.

The model is

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_{t-1} + u_t \tag{16.10}$$

so a change in X has a contemporaneous effect on Y (β_1) and an effect in the next period (β_2). The error term u_t follows an AR(1) process,

$$u_t = \phi_1 u_{t-1} + \widetilde{u}_t$$

where $\widetilde{u}_t$ is serially uncorrelated.

One can show that the ADL representation of this model is

$$Y_t = \alpha_0 + \phi_1 Y_{t-1} + \delta_0 X_t + \delta_1 X_{t-1} + \delta_2 X_{t-2} + \widetilde{u}_t \tag{16.11}$$

with the restrictions

$$\begin{aligned}\beta_1 &= \delta_0,\\ \beta_2 &= \delta_1 + \phi_1 \delta_0.\end{aligned}$$

16.5.0.0.1 Quasi-Differences

Another way of writing this ADL(1,2) representation is the *quasi-difference* model

$$\widetilde{Y}_t = \alpha_0 + \beta_1 \widetilde{X}_t + \beta_2 \widetilde{X}_{t-1} + \widetilde{u}_t \tag{16.12}$$

where $\widetilde{Y}_t = Y_t - \phi_1 Y_{t-1}$ and $\widetilde{X}_t = X_t - \phi_1 X_{t-1}$. Notice that the error term $\widetilde{u}_t$ is uncorrelated in both models and, as shown in Chapter 15.5 of the book,

$$E(u_t | X_{t+1}, X_t, X_{t-1}, \dots) = 0$$

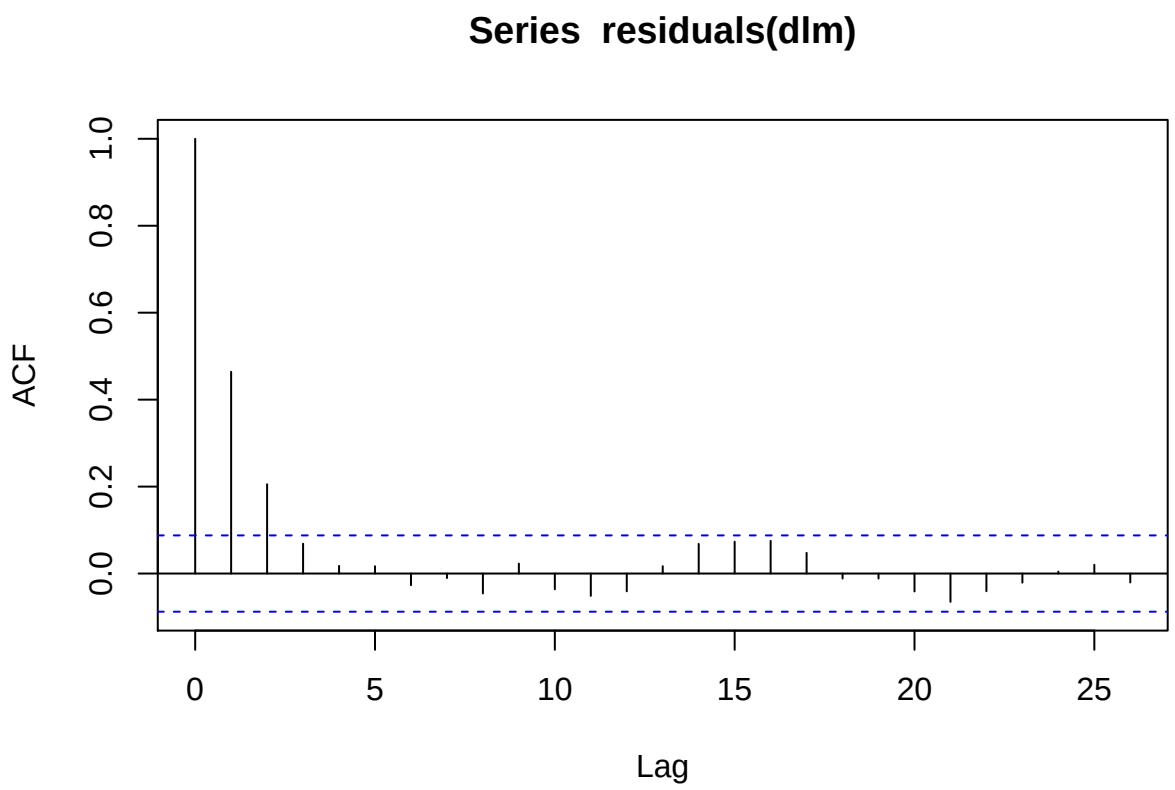
which is implied by the assumption of strict exogeneity.

We continue by simulation a time series of 500 observations using the model (16.10) where $\beta_1 = 0.1$, $\beta_2 = 0.25$, $\phi = 0.5$ and $\widetilde{u}_t \sim N(0,1)$ and start by estimating the distributed lag model.

```
# simulate time series with serially correlated errors
set.seed(1)

obs <- 501

eps <- arima.sim(n = obs-1 , model = list(ar = 0.5))
X <- arima.sim(n = obs, model = list(ar = 0.25))
Y <- 0.1 * X[-1] + 0.25 * X[-obs] + eps
X <- ts(X[-1])

# estimate the distributed lag model
dlm <- dynlm(Y ~ X + L(X))
```

It is straightforward to check whether the residuals of this model exhibit autocorrelation by using `acf()`.

```
# check that the residuals are serially correlated
acf(
  residuals(dlm)
)
```



The plot indicates that the residuals are autocorrelated. In particular, the pattern reveals that the residuals follow an autoregressive process, as the sample autocorrelation function decays quickly for the first few lags and is probably zero for higher lag orders. At any case, HAC standard errors should be used.

```
# coefficient summary using NeweyWest estimator
coeftest(dlm, vcov = NeweyWest(dlm, prewhite = F, adjust = T))
```

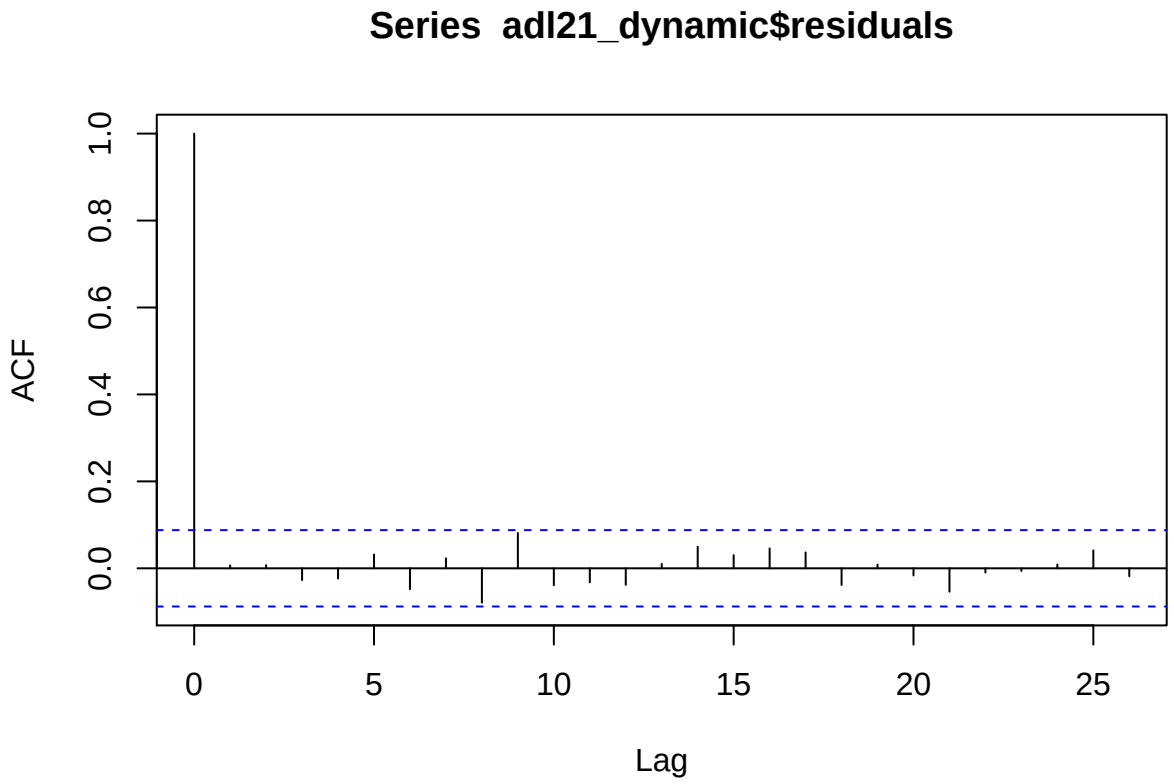
```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.038340   0.073411  0.5223  0.601717
## X           0.123661   0.046710  2.6474  0.008368 **
## L(X)        0.247406   0.046377  5.3347 1.458e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.5.0.0.2 OLS Estimation of the ADL Model

Next, we estimate the ADL(1,2) model (16.11) using OLS. Notice that the errors are uncorrelated in this representation of the model. This statement is supported by a plot of the sample autocorrelation function of the residual series.

```
# Estimate ADL(2, 1) representation
adl21_dynamic <- dynlm(Y ~ L(Y) + X + L(X, 1:2))
```

```
# plot sample autocorrelaltions of residuals
acf(adl21_dynamic$residuals)
```



The estimated coefficients in `adl21_dynamic$coefficients` *are not* the dynamic multiplers we are interested in but can be computed according to the restrictions in (16.11) where the true coefficients are replaced by their OLS estimates.

```
t <- adl21_dynamic$coefficients

# compute estimated dynamic effects using coefficient restrictions
# in the ADL(2,1) representation
c(
  "hat_beta_1" = t[3],
  "hat_beta_2" = t[4] + t[3] * t[2]
  )
```

```
##          hat_beta_1.X hat_beta_2.L(X, 1:2)1
##             0.1176425             0.2478484
```

16.5.0.0.3 GLS Estimation

Strict exogeneity allows OLS estimation of the quasi-difference model (16.12). The idea of applying the OLS estimator to a model where the variables are linearly transformed such that the model errors are uncorrelated and homoskedastic is called *generalized least squares* (GLS)

The OLS estimator in (16.12) is called the *infeasible GLS* estimator because we $\widetilde{Y}$ and $\widetilde{X}$ cannot be computed without knowledge of ϕ_1, the autoregressive coefficient in the error AR(1) model, which is generally unknown in practice.

Suppose $\phi = 0.5$ was known. The infeasible GLS estimates of the dynamic multipliers in (16.10) are obtained by applying OLS to the transformed data.

```
# GLS: estimate quasi-differenced specification by OLS
iGLS_dynamic <- dynlm(I(Y- 0.5 * L(Y)) ~ I(X - 0.5* L(X)) + I(L(X) - 0.5* L(X,2)) )

summary(iGLS_dynamic)
```

```
##
## Time series regression with "ts" data:
## Start = 3, End = 500
##
## Call:
## dynlm(formula = I(Y - 0.5 * L(Y)) ~ I(X - 0.5 * L(X)) + I(L(X) -
##     0.5 * L(X, 2)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0325 -0.6375 -0.0499  0.6658  3.7724
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.01620    0.04564   0.355  0.72273
## I(X - 0.5 * L(X))         0.12000    0.04237   2.832  0.00481 **
## I(L(X) - 0.5 * L(X, 2))   0.25266    0.04237   5.963 4.72e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 495 degrees of freedom
## Multiple R-squared:  0.07035,    Adjusted R-squared:  0.0666
## F-statistic: 18.73 on 2 and 495 DF,  p-value: 1.442e-08
```

The *feasible GLS* estimator uses preliminary estimation of the coefficients in the presumed error term model, computes the quasi-differenced data and then estimates the model using OLS. This idea was introduced by Cochrance and Orcutt (1949) and can be extended by continuing this process iteratively. Such a procedure is implemented in the function cochrane.orcutt() from the package orcutt.

```
X_t <- c(X[-1])
X_l1 <- c(X[-500])
Y_t <- c(Y[-1])

# iterated cochrance-orcutt procedure
summary(
  cochrane.orcutt(lm(Y_t ~ X_t + X_l1))
)
```

```
## Call:
## lm(formula = Y_t ~ X_t + X_l1)
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.032885   0.085163   0.386   0.69956
## X_t         0.120128   0.042534   2.824   0.00493 **
## X_l1        0.252406   0.042538   5.934 5.572e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.0165 on 495 degrees of freedom
## Multiple R-squared:  0.0704 ,  Adjusted R-squared:  0.0666
## F-statistic: 18.7 on 2 and 495 DF,  p-value: < 1.429e-08
##
## Durbin-Watson statistic
## (original):    1.06907 , p-value: 1.05e-25
## (transformed): 1.98192 , p-value: 4.246e-01
```

Some more sophisticated methods are provided with the package nlme. The function gls() can be used to fit linear models by maximum likelihood estimation algorithms and allows to specify a correlation structure for the error term.

```
# feasible GLS maximum likelihood estimation procedure
summary(
  gls(Y_t ~ X_t + X_l1, correlation = corAR1())
)
```

```
## Generalized least squares fit by REML
##   Model: Y_t ~ X_t + X_l1
##   Data: NULL
##        AIC     BIC    logLik
##   1451.847 1472.88 -720.9235
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##       Phi
## 0.4668343
##
## Coefficients:
##                   Value  Std.Error  t-value p-value
## (Intercept) 0.03929124 0.08530544 0.460595  0.6453
## X_t         0.11986994 0.04252270 2.818963  0.0050
## X_l1        0.25287471 0.04252497 5.946500  0.0000
##
##  Correlation:
##      (Intr) X_t
## X_t  0.039
## X_l1 0.037  0.230
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -3.00075518 -0.64255522 -0.05400347  0.69101814  3.28555793
##
## Residual standard error: 1.14952
## Degrees of freedom: 499 total; 496 residual
```

Feasible GLS is the BLUE of the dynamic multipliers when there is serial correlation and/or heteroskedasticity. Notice that in this example, the coefficient estimates produced by GLS are somewhat closer to their true values and that the standard errors are the smallest for the GLS estimator.

16.6 Orange Juice Prices and Cold Weather

In this section we investigate the following two questions using the time series regression methods discussed before:

- How persistent is the effect of a single freeze on prices?
- Has this effect been stable over the whole time span considered?

We start by estimating dynamic causal effects with a distributed lag model where $\%ChgOJC_t$ is regressed on FDD_t and 18 lags of it. A second model specification is a transformation of the the distributed lag model which allows to estimate the 19 cumulative dynamic multipliers using OLS. In a third model, we add 11 binary variables (one for each of the months from February to December) to adjust for a possible omitted variable bias arising from correllation of FDD_t and seasons by adding `season(FDD)` to the right hand side of the formula of the second model.

```
# estimate distributed lag models of frozen orange juice price changes
FOJC_mod_DM <- dynlm(FOJC_pctc ~ L(FDD, 0:18))
FOJC_mod_CM1 <- dynlm(FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18))
FOJC_mod_CM2 <- dynlm(FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD))
```

The models above include a large number of lags with default labels that correspond to the degree of differencing and the lag orders which makes it somewhat cumbersome to read the outcomes. The regressor labels of a model object may be altered by overriding the attribute `names` of the coefficient section using the function `attr()`. Thus, for better readability we use the lag orders as regressor labels.

Next, we compute HAC standard errors standard errors for each model using $NeweyWest()$ and gather the results in a list which is then supplied as the argument `se` to the function `stargazer()`, see below. Note that the sample consists of 612 observations.

```
length(FDD)
```

```
## [1] 612
```

According to (16.7), the rule of thumb for choosing the HAC standard error truncation parameter m, we choose

$$m = \left\lceil 0.75 \cdot 612^{1/3} \right\rceil = \lceil 6.37 \rceil = 7.$$

To check for sensitivity of the standard errors to different choices of the truncation parameter in the model that is used to estimate the cumulative multipliers, we also compute the Newey and West estimator for $m = 14$.

```
# gather hAC standard error errors in a list
SEs <- list(
  sqrt(
    diag(
      NeweyWest(FOJC_mod_DM, lag = 7, prewhite = F)
    )
  ),
  sqrt(
    diag(
      NeweyWest(FOJC_mod_CM1, lag = 7, prewhite = F)
    )
  ),
   sqrt(
    diag(
      NeweyWest(FOJC_mod_CM1, lag = 14, prewhite = F)
    )
```

```
  ),
  sqrt(
    diag(
      NeweyWest(FOJC_mod_CM2, lag = 7, prewhite = F)
    )
  )
)
```

The results are then used to reproduce the outcomes presented in Table 15.1 of the book.

```
stargazer(FOJC_mod_DM , FOJC_mod_CM1, FOJC_mod_CM1, FOJC_mod_CM2,
  title = "Dynamic Effects of a Freezing Degree Day on the Price of Orange Juice",
  header = FALSE,
  digits = 2,
  column.labels = c("Dynamic Multipliers", rep("Dynamic Cumulative Multipliers",3)),
  dep.var.caption  = "Monthly Percentage Change in Orange Juice Price",
  dep.var.labels.include = FALSE,
  covariate.labels = as.character(0:18),
  omit = "season",
  se = SEs,
  no.space = T,
  add.lines = list(
                  c("Monthly indicators?","no", "no", "no", "yes"),
                  c("HAC truncation","7", "7", "14", "7")
                  ),
  omit.stat = c("rsq", "f","ser")
  )
```

Dynamic Effects of a Freezing Degree Day on the Price of Orange Juice

Monthly Percentage Change in Orange Juice Price

Dynamic Multipliers

Dynamic Cumulative Multipliers

Dynamic Cumulative Multipliers

Dynamic Cumulative Multipliers

(1)

(2)

(3)

(4)

0

0.51***

0.51***

0.51***

0.52***

(0.14)

(0.14)

(0.14)

(0.14)
1
0.17**
0.68***
0.68***
0.72***
(0.09)
(0.13)
(0.13)
(0.14)
2
0.07
0.75***
0.75***
0.78***
(0.06)
(0.17)
(0.16)
(0.17)
3
0.07
0.82***
0.82***
0.86***
(0.04)
(0.18)
(0.18)
(0.19)
4
0.02
0.84***
0.84***
0.89***
(0.03)
(0.18)
(0.18)

(0.19)
5
0.03
0.87***
0.87***
0.90***
(0.03)
(0.19)
(0.19)
(0.20)
6
0.03
0.90***
0.90***
0.92***
(0.05)
(0.20)
(0.21)
(0.21)
7
0.02
0.91***
0.91***
0.94***
(0.02)
(0.20)
(0.21)
(0.21)
8
-0.04
0.87***
0.87***
0.90***
(0.03)
(0.21)
(0.22)

(0.22)
9
-0.01
0.86***
0.86***
0.88***
(0.05)
(0.24)
(0.24)
(0.24)
10
-0.12*
0.75***
0.75***
0.75***
(0.07)
(0.26)
(0.26)
(0.26)
11
-0.07
0.68**
0.68**
0.68**
(0.05)
(0.27)
(0.27)
(0.27)
12
-0.14*
0.54**
0.54**
0.55**
(0.08)
(0.27)
(0.27)

(0.27)
13
-0.08*
0.45*
0.45*
0.49*
(0.04)
(0.27)
(0.27)
(0.27)
14
-0.06
0.40
0.40
0.43
(0.03)
(0.27)
(0.28)
(0.28)
15
-0.03
0.37
0.37
0.41
(0.03)
(0.28)
(0.29)
(0.28)
16
-0.01
0.36
0.36
0.41
(0.05)
(0.28)
(0.29)

(0.29)
17
0.003
0.36
0.36
0.40
(0.02)
(0.29)
(0.29)
(0.29)
18
0.003
0.37
0.37
0.39
(0.02)
(0.29)
(0.30)
(0.29)
Constant
-0.34
-0.34
-0.34
-0.24
(0.27)
(0.27)
(0.26)
(0.93)
Monthly indicators?
no
no
no
yes
HAC truncation
7
7

14

7

Observations

594

594

594

594

Adjusted R2

0.11

0.11

0.11

0.10

Note:

p<0.1; ***p<0.05;*** p<0.01

Acording to coloumn (1), the contemporaneous effect of a freezing degree day is an increase of 0.5% in orange juice prices. The estimated effect is only 0.17% for the next month and close to zero for subsequent months. In fact, for all lags larger than 1, we cannot reject the individual null hypotheses that the respective coefficients are zero. Notice also that the model `FOJC_mod_DM` does only explain little variation in the dependent variable ($\overline{R}^2 = 0.11$).

Columns (2) and (3) present estimates of the dynamic cumulative multipliers of model `FOJC_mod_CM1`. Apparently, it does not matter whether we choose $m = 7$ or $m = 14$ when computing HAC standard errors so we stick with $m = 7$ and the standard errors reported in column (2).

If demand for orange juice is higher in winter, FDD_t would be correlated with the error term since since freezes occur rather in winter so we would face omitted variable bias. The third model estimate, `FOJC_mod_CM2`, accounts for this possible issue by using an additional set of 11 monthly dummies. For brevity, estimates of the dummy coefficients are excluded from the output produced by stargazer (this is achieved by setting `omit = "season"`). We may check that the dummy for January was omitted to prevent perfect multicollinearity.

```
# estimates on mothly dummies
FOJC_mod_CM2$coefficients[-c(1:20)]
```

```
## season(FDD)Feb season(FDD)Mar season(FDD)Apr season(FDD)May season(FDD)Jun
##     -0.9565759     -0.6358007      0.5006770     -1.0801764      0.3195624
## season(FDD)Jul season(FDD)Aug season(FDD)Sep season(FDD)Oct season(FDD)Nov
##      0.1951113      0.3644312     -0.4130969     -0.1566622      0.3116534
## season(FDD)Dec
##      0.1481589
```

A comparison of the estimates presented in columns (3) and (4) indicates that adding monthly dummies has a negligible effect. Further evidence for this comes from a test of the hypothesis that the 11 dummy coefficients are jointly zero. Instead of using `linearHypothesis()` which requires that all 11 restrictions are provided in a matrix, we use the function `waldtest()` and supply two model objects instead: `unres_model`, the unrestricted model object which is the same as `FOJC_mod_CM2` (except for the coefficient names since we have modified them above) and `res_model`, the model where the restriction that all dummy coefficients are zero is imposed. `res_model` is conveniently obtained using the function `update()`. It extracts the argument

formula of a model object, updates it as specified and then re-fits the model. By setting `formula = . ~ . - season(FDD)` we impose that the monthly dummies do not enter the model.

```
# test if coefficients on monthly dummies are zero

unres_model <- dynlm(FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD))

res_model <- update(unres_model, formula = . ~ . - season(FDD))

waldtest(unres_model,
         res_model,
         vcov = NeweyWest(unres_model, lag = 7, prewhite = F))
```

```
## Wald test
##
## Model 1: FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18) + season(FDD)
## Model 2: FOJC_pctc ~ L(d(FDD), 0:17) + L(FDD, 18)
##   Res.Df  Df      F Pr(>F)
## 1    563
## 2    574 -11 0.9683 0.4743
```

The p-value is 0.47 so we cannot reject the hypothesis that the coefficients on the monthly dummies are zero, even at the 10% level of significance. We conclude that the seasonal fluctuations in demand for orange juice do not pose a serious threat to internal validity of the model.

It is convenient to use plots of dynamic multipliers and cumulative dynamic multipliers. The following two code chunks reproduce Figures 15.2 (a) and 15.2 (b) of the book which display point estimates of dynamic and cumulative multipliers along with upper and lower bounds of their 95% confidence intervals computed using the HAC standard errors from above.

```
# 95% CI bounds
point_estimates <- FOJC_mod_DM$coefficients

CI_bounds <- cbind(
   "lower" = point_estimates - 1.96 * SEs[[1]],
   "upper" = point_estimates + 1.96 * SEs[[1]]
 )[-1,]


# plot estimated dynamic multipliers
plot(0:18, point_estimates[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     ylim = c(-0.4, 1),
     xlab = "Lag",
     ylab = "Dynamic multiplier",
     main = "Dynamic Effect of FDD on Orange Juice Price"
     )

# add dashed line at 0
abline(h = 0, lty = 2)

# add CI bounds
lines(0:18, CI_bounds[,1], col = "darkred")
lines(0:18, CI_bounds[,2], col = "darkred")
```

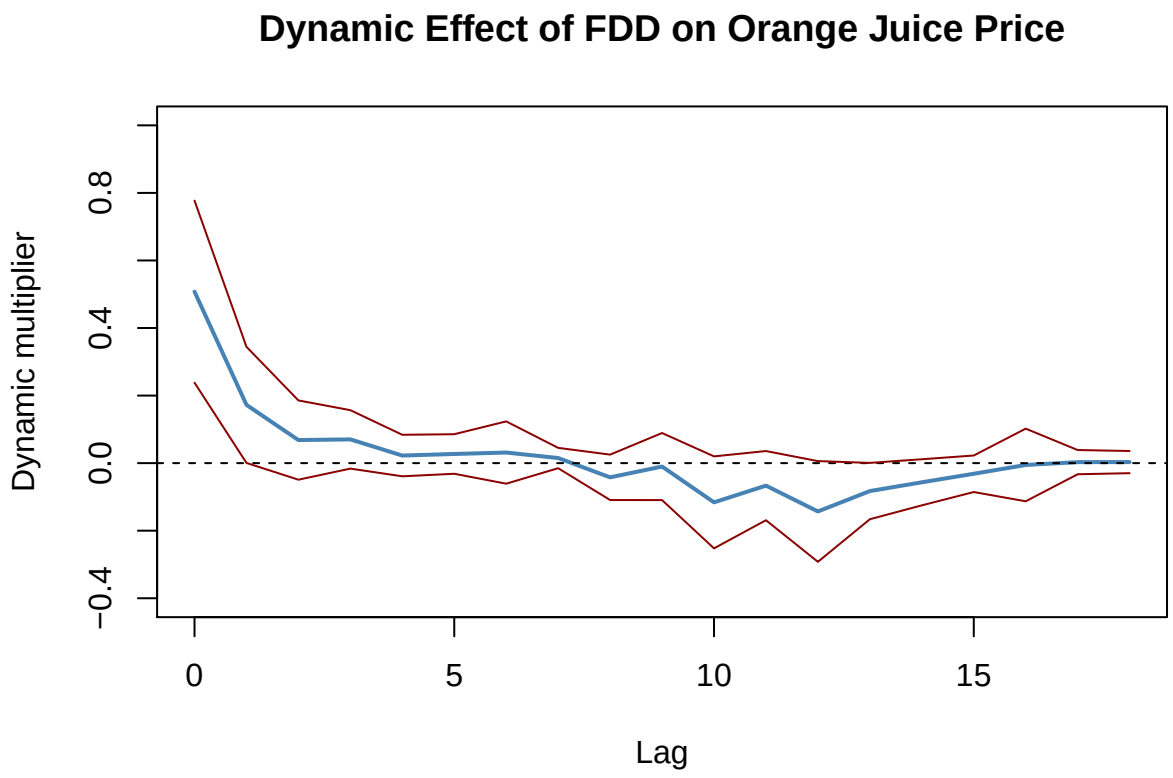


Figure 16.1: Dynamic Multipliers

Notice that the 95% confidence intervals plotted in Figure 16.1 indeed include zero for lags larger than 1 such that the null of a zero multiplier cannot be rejected for these lags.

```
# 95% CI bounds
point_estimates <- FOJC_mod_CM1$coefficients

CI_bounds <- cbind(
   "lower" = point_estimates - 1.96 * SEs[[2]],
   "upper" = point_estimates + 1.96 * SEs[[2]]
 )[-1,]


# plot estimated dynamic multipliers
plot(0:18, point_estimates[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     ylim = c(-0.4, 1.6),
     xlab = "Lag",
     ylab = "Cumulative dynamic multiplier",
    main = "Cumulative Dynamic Effect of FDD on Orange Juice Price"
     )

# add dashed line at 0
abline(h = 0, lty = 2)

# add CI bounds
```

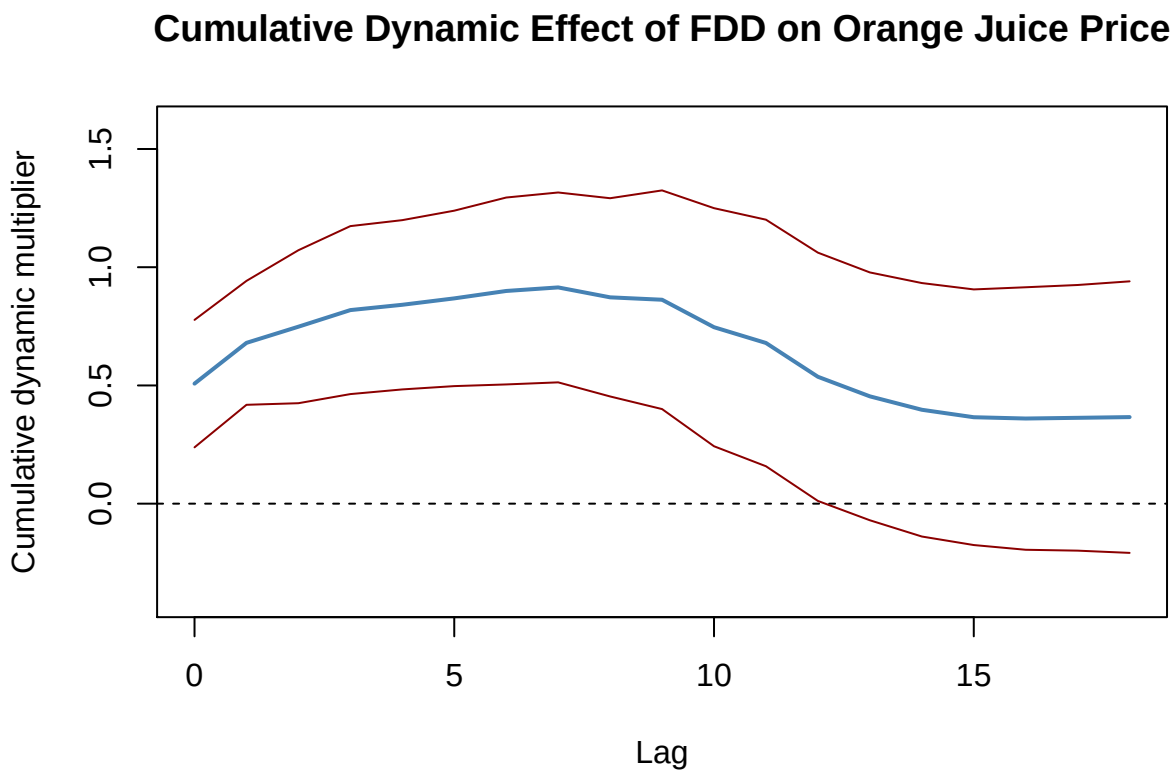


Figure 16.2: Dynamic Cumulative Multipliers

```
lines(0:18, CI_bounds[, 1], col = "darkred")
lines(0:18, CI_bounds[, 2], col = "darkred")
```

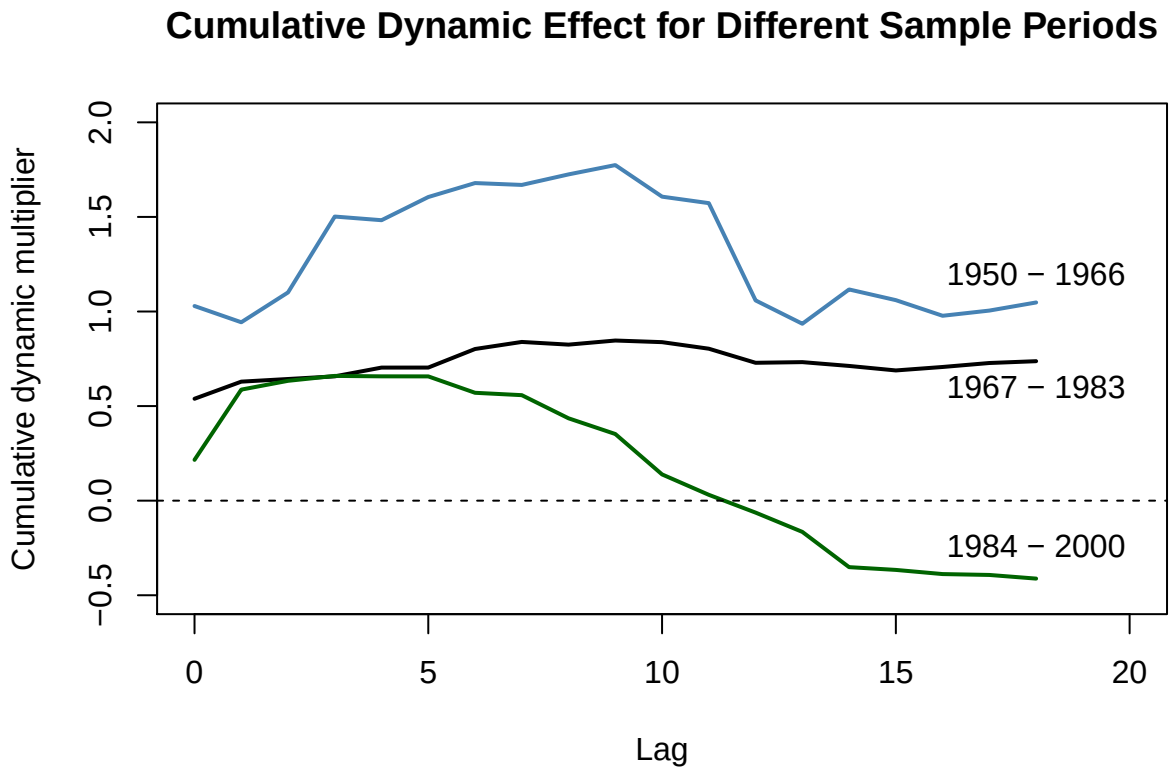
As can be seen from Figure 16.2, the estimated dynamic cumulative multipliers grow through the seventh month up to a price increase of about 0.91% and then decrease slighly to the estimated long-run cumulative multiplier of 0.37% which, however, is not significantly different from zero at the 5% level.

Have the dynamic multipliers been stable over time? One way to see this is the estimate them for different subperiods of the sample. For example, consider periods 1950 - 1966, 1967 - 1983 and 1984 - 2000. If the multipliers are the same for all three periods the estimates should be close and thus the estimated cumulative multipliers should be similar, too. We investigate this by re-estimating `FOJC_mod_CM1` for the three different time spans and then plot the estimated cumulative dynamic multipliers for the comparison.

```
# estimate cumulative multiplieres using different sample periods
FOJC_mod_CM1950 <- update(FOJC_mod_CM1, start = c(1950,1), end = c(1966,12))

FOJC_mod_CM1967 <- update(FOJC_mod_CM1, start = c(1967,1), end = c(1983,12))

FOJC_mod_CM1984 <- update(FOJC_mod_CM1, start = c(1984,1), end = c(2000,12))
```

```
# plot estimated dynamic cumulative multipliers (1950-1966)
plot(0:18, FOJC_mod_CM1950$coefficients[-1],
     type = "l",
     lwd = 2,
     col = "steelblue",
     xlim = c(0,20),
     ylim = c(-0.5, 2),
```

```
     xlab = "Lag",
     ylab = "Cumulative dynamic multiplier",
     main = "Cumulative Dynamic Effect for Different Sample Periods"
)

# plot estimated dynamic multipliers (1967-1983)
lines(0:18, FOJC_mod_CM1967$coefficients[-1], lwd = 2)

# plot estimated dynamic multipliers (1984-2000)
lines(0:18, FOJC_mod_CM1984$coefficients[-1], lwd = 2, col = "darkgreen")

# add dashed line at 0
abline(h = 0, lty = 2)

# add annotations
text(18, -0.24, "1984 - 2000")
text(18, 0.6, "1967 - 1983")
text(18, 1.2, "1950 - 1966")
```



Clearly, the cumulative dynamic multipliers have changed considerarbly over time. Notice that the effect of a freeze was stronger an more persistent in the 1950s and 1960s. For the 1970s the magnitude of the effect was lower but still highly persistent. We observe an even lower magnitude for the final third of the sample (1984 - 2000) where the long-run effect is much less persist and essentialy zero after a year.

Evidence of a QLR test for a break in the population distributed lag regression of column (1) with 15% trimming and HAC variance-covariance matrix estimator supports the conjection that the population regression coefficients have changed over time.

```
# set up a range of possible break dates
tau <- c(
```

```
  window(
    time(FDD),
    time(FDD)[round(612/100*15)],
    time(FDD)[round(612/100*85)]
    )
  )

# initialize vector of F-statistics
Fstats <- numeric(length(tau))

# restricted model
res_model <- update(unres_model, formula = . ~ L(FDD, 0:18))


# estimation loop over break dates
for(i in 1:length(tau)) {

  # set up dummy variable
  D <- time(FOJC_pctc) > tau[i]

  # estimate DL model with intercations
  unres_model <- dynlm(FOJC_pctc ~ L(FDD, 0:18) + D*L(FDD, 0:18))

  # compute and save F-statistic
  Fstats[i] <- waldtest(unres_model,
                        res_model,
                        vcov = NeweyWest(unres_model, lag = 7, prewhite = F))$F[2]

}
```

Note that this code takes a couple of seconds to run since a total of 429 regressions with 40 model coefficients each are estimated.

```
# QLR test statistic
max(Fstats)
```

```
## [1] 36.76819
```

The QLR statistic is 36.77. From table 14.5 of the book we see that the 1% critical value for the QLR test with 15% trimming and $q = 20$ restrictions is 2.43. Since this is a right-sided test, the QLR statistic clearly lies in the region of rejection so we can discard the null hypothesis of no break in the population regression function.

See Chapter 15.7 of the book for a discussion of empirical examples where it is questionable whether the assumption of (past and present) exogeneity of regressors is plausible.

16.7 Summary

- We have seen how R can be used to estimate the time path of the effect on Y of a change in X (the dynamic causal effect on Y of a change in X) using time series data on both. The corresponding model is called the *distributed lag model*. Distributed lag models are conveniently estimated using the function dynlm() from the package dynlm.
- The regression error in distributed lag models is often serially correlated such that standard errors which are robust to heteroskedasticity and autocorrelation should be used to obtain valid inference

based on t-statistics and confidence intervals. The package sandwich has functions for compuation of so-called HAC covariance matrix estimators, for example vcovHAC() and NeweyWest().

- When X is *strictly exogeneous*, more efficient estimates can be obtained using an ADL model or by GLS estimation. Feasible GLS algorithms can be found in the R packages orcutt and nlme. We have stressed that the assumption of strict exogeneity is often implausible in empirical applications. See chapter 15.7 of the book for further examples.

Chapter 17

Additional Topics in Time Series Regression

This chapter introduces the following advanced topics in time series regression and demonstrates how core techniques can be applied using R:

- Vector autoregressions (VARs). We focus on using VARs for forecasting. Another branch of the literature is concerned with so-called *Structural VARs* which are, however, beyond the scope of this chapter.
- Multiperiod forecasts. This includes a discussion of iterated and direct (multivariate) forecasts.
- The DF-GLS test, a more powerful method for unit root testing than the ADF test.
- Cointegration analysis with an application to short- and long-term interest rates. We demonstrate how to estimate a vector error correction model.
- Autoregressive conditional heteroskedasticity. We show how a GARCH(1,1) model can be used for quantifying the risk associated with investing in the stock market in terms of estimation and forecasting of the volatility of asset returns.

If you are intersted in reproducing the code examples on your own device, install the R packages listed below and make sure that the subsequent code chunk executes without any errors.

```
library(AER)
library(readxl)
library(dynlm)
library(vars)
library(quantmod)
library(scales)
library(fGarch)
```

17.1 Vector Autoregressions

A Vector autoregressive (VAR) model is useful when one is interested in predicting multiple time series variables using a single model. At its core, the VAR model is an extension of the univariate autoregressive model we have dealt with in Chapters 14 and 15. Key Concept 16.1 summarizes the essentails of VAR.

Key Concept 16.1

Vector Autoregressions

The vector auoregression (VAR) model extends the idea of univariate autoregression to k time series regression where the lagged values of *all* k series appear as regressors. Put differently, in a VAR model we regress a *vector* of time series variables on lagged vectors of these variables. As for AR(p) models, the lag order is denoted by p so the VAR(p) model of two variables X_t and Y_t ($k = 2$) is given by the equations

$$\begin{aligned} Y_t =& \beta_{10} + \beta_{11} Y_{t-1} + \cdots + \beta_{1p} Y_{t-p} + \gamma_{11} X_{t-1} + \cdots + \gamma_{1p} X_{t-p} + u_{1t} \\ X_t =& \beta_{20} + \beta_{21} Y_{t-1} + \cdots + \beta_{2p} Y_{t-p} + \gamma_{21} X_{t-1} + \cdots + \gamma_{2p} X_{t-p} + u_{2t}. \end{aligned}$$

The αs and βs can be estimated using OLS in each equation. The assumptions for VARs are the time series assumptions presented in Key Concept 14.6 applied to teach equation.

It is straightforward to estimate VAR models in R. A valid way is to simply use lm() for estimation of the individual equation. Furthermore, the R package VARs provides standard tools for estimation, diagnostic testing and prediction using this type of models.

When the assumptions referred to by Key Concept 16.1 hold, the OLS estimators of the VAR coefficients are consistent and jointly normal in large samples so that usual inferential methods such as confidence intervals und t-statistics can be used.

Notice that, owing to the structure of VARs, we may jointly test restrictions across multiple equations. For instance, it may of interest to test whether the coefficients on all regressors on the lag p are zero. This corresponds to testing the null hypothesis that the lag length $p - 1$ is correct. Large sample joint normality of the coefficient estimates is a convenient thing because this implies that we may use an F-test for this testing problem. The explicit formula for such a test statistic is rather complicated but fortunately such computations are easily done using R functions we will work with for the course of this chapter. Another way for determining optimal lag lengths are information criteria like the BIC we have introduced for univariate time series regressions. For a multiple equation model, we choose the specification which has the smallest $BIC(p)$ where

$$BIC(p) = \log \left[\det(\widehat{\Sigma}_u) \right] + k(kp + 1) \frac{\log(T)}{T}.$$

with $\widehat{\Sigma}_u$ the $k \times k$ estimate of the covariance matrix of the VAR errors and $\det(\cdot)$ denotes the determinant.

As for univariate distributed lag models, one should think carefully which variables to include in a VAR because adding unrelated variables reduces the forecast accuracy by increasing the estimation error. This is particulary important because the number of parameters to be estimated grows proportionally to the number of variables used. In the application below we shall see that economic thoery and empirical evidence are helpful for the decision.

17.1.0.0.1 A VAR Model of the Growth Rate of GDP and the Term Spread

In the following we show how to estimate a VAR model of the GDP growth rate, $GDPGR$, and the term spread, $TSpread$. As a consequence to the discussion on nonstationarity of GDP growth in Chapter 14.7 (remember the possible break in the early 1980s detected by the QLR test statistic), we use data from 1981:Q1 to 2012:Q4. The two model equations are

$$\begin{aligned} GDPGR_t =& \beta_{10} + \beta_{11} GDPGR_{t-1} + \beta_{12} GDPGR_{t-2} + \gamma_{11} TSpread_{t-1} + \gamma_{12} TSpread_{t-2} + u_{1t} \\ TSpread_t =& \beta_{20} + \beta_{21} GDPGR_{t-1} + \beta_{22} GDPGR_{t-2} + \gamma_{21} TSpread_{t-1} + \gamma_{22} TSpread_{t-2} + u_{2t} \end{aligned}$$

The data set us_macro_quarterly.xlsx is provided by the authors and can be downloaded here. It provides data on quarterly data on US real (i.e. inflation adjusted) GDP from years 1947 to 2004. We begin by

importing the data set and do some formatting (we already worked with this data set in Chapter 14 so you may skip these steps if you have already loaded the data in your working environment).

```
# load US macroeconomic data
USMacroSWQ <- read_xlsx("Data/us_macro_quarterly.xlsx",
                        sheet = 1,
                        col_types = c("text", rep("numeric", 9))
                       )

# set column names
colnames(USMacroSWQ) <- c("Date", "GDPC96", "JAPAN_IP", "PCECTPI", "GS10", "GS1", "TB3MS", "UNRATE", "EX

# formate date column
USMacroSWQ$Date <- as.yearqtr(USMacroSWQ$Date, format = "%Y:0%q")

# define GDP as ts object
GDP <- ts(USMacroSWQ$GDPC96,
          start = c(1957, 1),
          end = c(2013, 4),
          frequency = 4)

# define GDP growth as ts object
GDPGrowth <- ts(400*log(GDP[-1]/GDP[-length(GDP)]),
                start = c(1957, 2),
                end = c(2013, 4),
                frequency = 4)

# 3 months Treasury bill interest rate as ts object
TB3MS <- ts(USMacroSWQ$TB3MS,
            start = c(1957, 1),
            end = c(2013, 4),
            frequency = 4)

# 10 years Treasury bonds interest rate as ts object
TB10YS <- ts(USMacroSWQ$GS10,
             start = c(1957, 1),
             end = c(2013, 4),
             frequency = 4)

# term spread series
TSpread <- TB10YS - TB3MS
```

We estimate both equations separately by OLS and use `coeftest()` in conjunction with `NeweyWest()` to obtain HAC standard errors.

```
# Estimate both equations using 'dynlm()'
VAR_EQ1 <- dynlm(GDPGrowth ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2), start = c(1981, 1), end = c(2012, 4))
VAR_EQ2 <- dynlm(TSpread ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2), start = c(1981, 1), end = c(2012, 4))

# rename regressors for better readability
names(VAR_EQ1$coefficients) <- c("Intercept","Growth_t-1","Growth_t-2","TSpread_t-1","TSpread_t-2")
names(VAR_EQ2$coefficients) <- names(VAR_EQ1$coefficients)

# HAC coefficient summaries
coeftest(VAR_EQ1, vcov. = NeweyWest(VAR_EQ1, prewhite = F, adjust = T))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## Intercept    0.51634    0.46481  1.1109 0.268790
## Growth_t-1   0.28955    0.12519  2.3130 0.022386 *
## Growth_t-2   0.21639    0.09169  2.3600 0.019846 *
## TSpread_t-1 -0.90255    0.38939 -2.3178 0.022111 *
## TSpread_t-2  1.32983    0.46117  2.8836 0.004643 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(VAR_EQ2, vcov. = NeweyWest(VAR_EQ2, prewhite = F, adjust = T))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## Intercept    0.4557740  0.1292772  3.5256 0.0005945 ***
## Growth_t-1   0.0099785  0.0264497  0.3773 0.7066285
## Growth_t-2  -0.0572451  0.0303573 -1.8857 0.0616927 .
## TSpread_t-1  1.0582279  0.0924398 11.4478 < 2.2e-16 ***
## TSpread_t-2 -0.2191902  0.0840014 -2.6094 0.0101965 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We end up with the following results:

$$GDPGR_t = \underset{(0.46)}{0.52} + \underset{(0.13)}{0.29} GDPGR_{t-1} + \underset{(0.09)}{0.22} GDPGR_{t-2} - \underset{(0.39)}{0.90} TSpread_{t-1} + \underset{(0.46)}{1.33} TSpread_{t-2}$$

$$TSpread_t = \underset{(0.13)}{0.46} + \underset{(0.03)}{0.01} GDPGR_{t-1} - \underset{(0.03)}{0.06} GDPGR_{t-2} + \underset{(0.09)}{1.06} TSpread_{t-1} - \underset{(0.08)}{0.22} TSpread_{t-2}$$

The function VAR() can be used to obtain the same coefficient estimates as presented above since it applies OLS per equation, too.

```
# set up data for estimation using 'VAR()'
VAR_data <- window(ts.union(GDPGrowth, TSpread), start = c(1980,3), end = c(2012,4))

# estimate model coefficients using 'VAR()'
VAR_est <- VAR(y = VAR_data, p = 2)
VAR_est
```

```
##
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation GDPGrowth:
## ==============================================
## Call:
## GDPGrowth = GDPGrowth.l1 + TSpread.l1 + GDPGrowth.l2 + TSpread.l2 + const
##
## GDPGrowth.l1   TSpread.l1 GDPGrowth.l2   TSpread.l2        const
##    0.2895533   -0.9025493    0.2163919    1.3298305    0.5163440
##
```

```
##
## Estimated coefficients for equation TSpread:
## ============================================
## Call:
## TSpread = GDPGrowth.l1 + TSpread.l1 + GDPGrowth.l2 + TSpread.l2 + const
##
## GDPGrowth.l1   TSpread.l1 GDPGrowth.l2   TSpread.l2        const
##  0.009978489  1.058227945 -0.057245123 -0.219190243  0.455773969
```

Notice that VAR() returns a list of `lm` objects which can be passed to the usual function, for example summary() and so it is straightforward to obtain model statistics for the individual equations.

```
# obtain the adj. R^2 from the output of 'VAR()'
summary(VAR_est$varresult$GDPGrowth)$adj.r.squared
```

```
## [1] 0.2887223
```

```
summary(VAR_est$varresult$TSpread)$adj.r.squared
```

```
## [1] 0.8254311
```

We may use the individual model objects to conduct granger causality tests.

```
# granger causality test:

# test if term spread has no power in explaining GDP growth
linearHypothesis(VAR_EQ1,
                 hypothesis.matrix = c("TSpread_t-1", "TSpread_t-2"),
                 vcov. = NeweyWest(VAR_EQ1, prewhite = F, adjust = T))
```

```
## Linear hypothesis test
##
## Hypothesis:
## TSpread_t - 0
## TSpread_t - 2 = 0
##
## Model 1: restricted model
## Model 2: GDPGrowth ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F  Pr(>F)
## 1    125
## 2    123  2 4.5584 0.01231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# test if GDP growth has no power in explaining term spread
linearHypothesis(VAR_EQ2,
                 hypothesis.matrix = c("Growth_t-1", "Growth_t-2"),
                 vcov. = NeweyWest(VAR_EQ2, prewhite = F, adjust = T))
```

```
## Linear hypothesis test
##
## Hypothesis:
## Growth_t - 0
## Growth_t - 2 = 0
##
```

```
## Model 1: restricted model
## Model 2: TSpread ~ L(GDPGrowth, 1:2) + L(TSpread, 1:2)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F  Pr(>F)
## 1    125
## 2    123  2 3.4806 0.03386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We conclude that both granger causality tests reject at the level of 5%.

17.1.0.0.2 Iterated Multivariate Forecasts using an iterated VAR

The idea of an iterated forecast for period $T+2$ based on observations up to period T is to use the one-period-ahead forecast as an intermediate step, that is the forecast for period $T+1$ is used as an observation when predicting the level of a series for period $T+2$. This can be generalized to a h period ahead forecast where all intervening periods between T and $T+h$ must be forecasted as they are used as observations in the process (see Chapter 16.2 of the book for a more detailed argument with this topic). Iterated Multiperiod forecasts are summarized in Key Concept 16.2.

Key Concept 16.2

Interated Multiperiod Forecasts

The steps fot an **interated multiperiod AR forecast** are:

1. Estimate the AR(p) model using OLS and compute the one-period-ahead forecast.
2. Use the one-period-ahead forecast to obtain the two-period-ahead forecast.
3. Continue by iterating to obtain forecasts further into the future.

An **interated multiperiod VAR forecast** is done as follows:

1. Estimate the VAR(p) model unsing OLS per equation and compute the one-period-ahead forecast for *all* variables in the VAR.
2. Use the one-period-ahead forecasts to obtain the two-period-ahead forecasts.
3. Continue by iterating to obtain forecasts of all variables in the VAR further into the future.

Since in a VAR the variables are modeled using lags of the repsective other variables, we need to compute forecasts for *all* variables. This can be cumbersome when the VAR is large but furtnunately there are R functions that facilitate this. For example, the function predict() can be used to obtain iterated multivariate forecasts for VAR models estimated by the function VAR().

The following code chunk shows how to compute forecasts for GDP growth and the term spread up to period 2015:Q1, that is $h = 10$, using the model object `VAR_est`.

```
# compute iterated forecasts for GDP growth and term spread for the next 10 periods
forecasts <- predict(VAR_est)
forecasts
```

```
## $GDPGrowth
##           fcst     lower    upper       CI
##  [1,] 1.738653 -3.006124 6.483430 4.744777
##  [2,] 1.692193 -3.312731 6.697118 5.004925
##  [3,] 1.911852 -3.282880 7.106583 5.194731
##  [4,] 2.137070 -3.164247 7.438386 5.301317
```

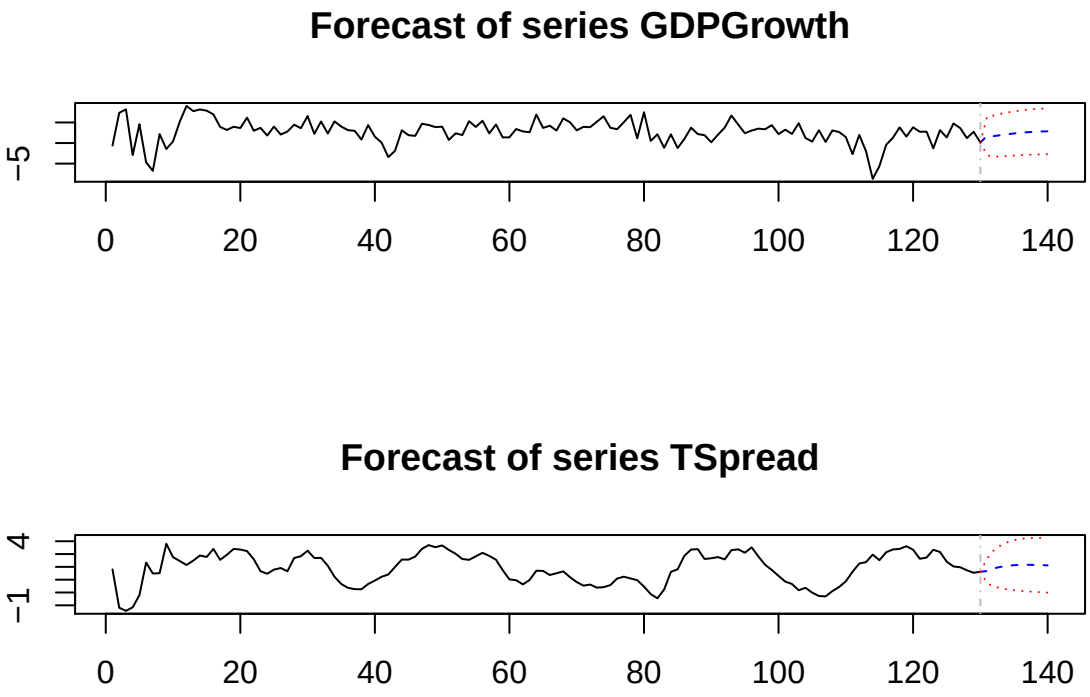
```
##  [5,] 2.329667 -3.041435 7.700769 5.371102
##  [6,] 2.496815 -2.931819 7.925449 5.428634
##  [7,] 2.631849 -2.846390 8.110088 5.478239
##  [8,] 2.734819 -2.785426 8.255064 5.520245
##  [9,] 2.808291 -2.745597 8.362180 5.553889
## [10,] 2.856169 -2.722905 8.435243 5.579074
##
## $TSpread
##           fcst        lower    upper        CI
##  [1,] 1.676746  0.708471226 2.645021 0.9682751
##  [2,] 1.884098  0.471880228 3.296316 1.4122179
##  [3,] 1.999409  0.336348101 3.662470 1.6630609
##  [4,] 2.080836  0.242407507 3.919265 1.8384285
##  [5,] 2.131402  0.175797245 4.087008 1.9556052
##  [6,] 2.156094  0.125220562 4.186968 2.0308738
##  [7,] 2.161783  0.085037834 4.238528 2.0767452
##  [8,] 2.154170  0.051061544 4.257278 2.1031082
##  [9,] 2.138164  0.020749780 4.255578 2.1174139
## [10,] 2.117733 -0.007139213 4.242605 2.1248722
```

From this we can tell that the two-quarter-ahead forecast of GDP growth in 2013:Q2 using data through 2012:Q4 is 1.69. For the same peiod, the iterated VAR forecast for the term spread is 1.88.

Notice that the matrices returned by `predict(VAR_est)` also include 95% prediction intervals (however, the function does not adjust for autocorrelation or heteroskedasticity of the errors!).

We may also plot the iterated forecasts for both variables by simply calling `plot()` on the output of `predict(VAR_est)`.

```
# visualize the iterated forecasts
plot(forecasts)
```



17.1.0.0.3 Direct Multiperiod Forecasts

A direct multiperiod forecast takes the model as a starting point where the predictor variables are lagged appropriately such that available observation can be used *directly* to do the forecast. The idea of direct multiperiod forecasting is summarized in Key Concept 16.3.

For example, to obtain two-quarter-ahead forecasts of GDP growth and term spread we first estimate the models

$$\begin{aligned} GDPGR_t &= \beta_{10} + \beta_{11} GDPGR_{t-2} + \beta_{12} GDPGR_{t-3} + \gamma_{11} TSpread_{t-2} + \gamma_{12} TSpread_{t-3} + u_{1t} \\ TSpread_t &= \beta_{20} + \beta_{21} GDPGR_{t-2} + \beta_{22} GDPGR_{t-3} + \gamma_{21} TSpread_{t-2} + \gamma_{22} TSpread_{t-3} + u_{2t} \end{aligned}$$

and then substitute the values of $GDPGR_{2012:Q4}$, $GDPGR_{2012:Q3}$, $TSpread_{2012:Q4}$ and $TSpread_{2012:Q3}$ into both equations. This is easily done manually.

```
# estimate models for direct two-quarter-ahead forecasts
VAR_EQ1_direct <- dynlm(GDPGrowth ~ L(GDPGrowth, 2:3) + L(TSpread, 2:3),
                        start = c(1981, 1), end = c(2012, 4))

VAR_EQ2_direct <- dynlm(TSpread ~ L(GDPGrowth, 2:3) + L(TSpread, 2:3),
                        start = c(1981, 1), end = c(2012, 4))

# compute direct two-quarter-ahead forecasts
coef(VAR_EQ1_direct) %*% c(1,
                           window(GDPGrowth, start = c(2012,3), end = c(2012,4)),
                           window(TSpread, start = c(2012,3), end = c(2012,4)))
```

```
##          [,1]
## [1,] 2.439497
```

```
coef(VAR_EQ2_direct) %*% c(1,
                           window(GDPGrowth, start = c(2012,3), end = c(2012,4)),
                           window(TSpread, start = c(2012,3), end = c(2012,4)))
```

```
##         [,1]
## [1,] 1.66578
```

Key Concept 16.3

Direct Multipreiod Forecasts

A **direct multiperiod forecast** that forecasts h periods into the future using a model of Y_t and an additional predictor X_t with p lags is done by first estimating

$$\begin{aligned} Y_t = \delta_0 + \delta_1 Y_{t-h} + \cdots + \delta_p Y_{t-p-h+1} + \delta_{p+1} X_{t-h} \\ + \cdots + \delta_{2p} Y_{t-p-h+1} + u_t \end{aligned}$$

which is then used to compute the forecast of Y_{T+h} based on observations throuth period T.

Applied economists often use the iterated method since this forecasts are more reliable in terms of $MSFE$ provided the one-period-ahead model is correctly specified. If this is not the case, for example because one equation in a VAR is believed to be misspecified, it can be beneficial to use direct forecasts since the iterated method will then be biased and thus have a higher $MSFE$ then the direct method. See Chapter 16.2 for a more detailed discussion on advantages and disadvantages of both methods.

17.2 Orders of Integration and the DF-GLS Unit Root Test

Some economic variables have smoother trends than variables that can be described by random walk models. A way to model these time series is

$$\Delta Y_t = \beta_0 + \Delta Y_{t-1} + u_t$$

, where u_t is a serially uncorrelated error term. This model states that the first differences of a series follow a random walk. Consequently, the series of second differences of Y_t is stationary. Key Concept 16.4 summarizes the notation.

Key Concept 16.4

Orders of Integration, Differencing and Stationarity

- When a time series Y_t has a unit autoregressive root, we say that Y_t is integrated of order one. This is often denoted by $Y_t \sim I(1)$. We simply say that Y_t is $I(1)$. If Y_t is $I(1)$, its first difference ΔY_t is stationary.

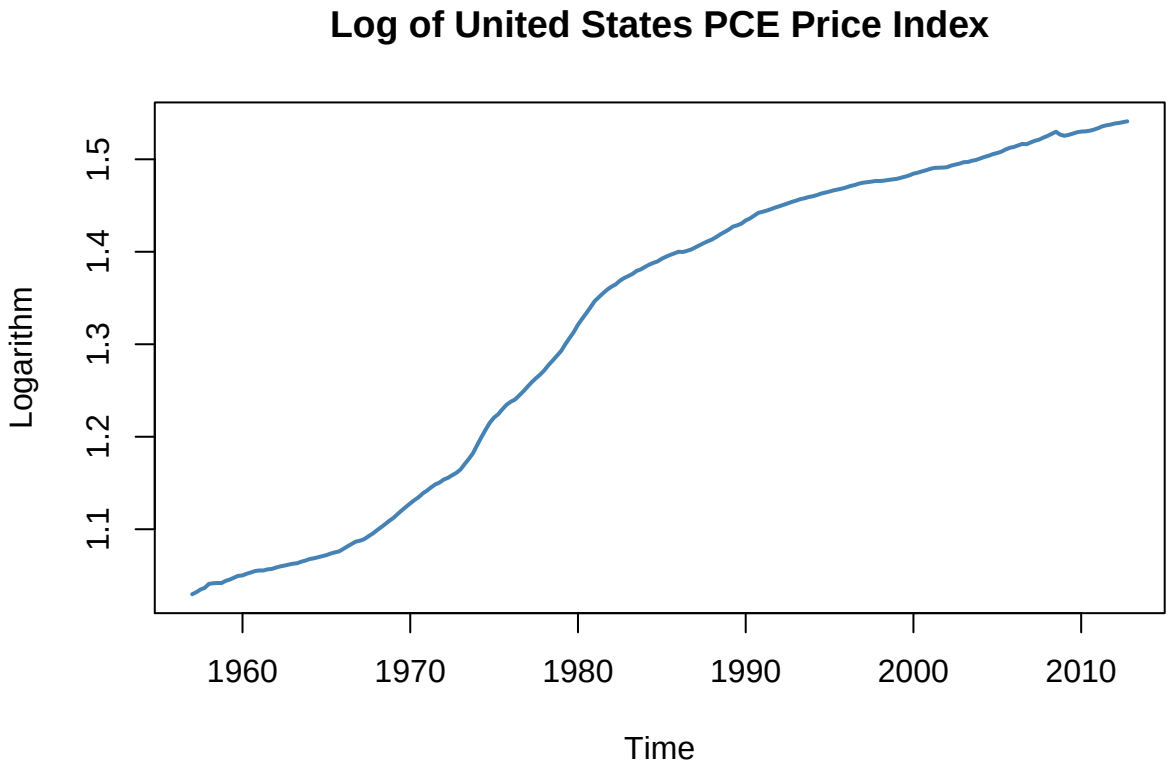
- We say that Y_t is $I(2)$ when Y_t needs to be differenced to time to obtain a stationary series. Using the notation introduced here we say that if Y_t is $I(2)$, it difference first ΔY_t is $I(1)$ and its second difference $\Delta^2 Y_t$ is stationary. We say that Y_t is $I(d)$ when Y_t must be differenced d times to obtain a stationary series.

- When Y_t is stationary, we say it is integrated of order 0 so Y_t is $I(0)$.

It is fairly easy to obtain differences of time series in R. For example, the function diff() returns suitably lagged and iterated differences of numeric vectors, matrices and time series objects of the class ts.

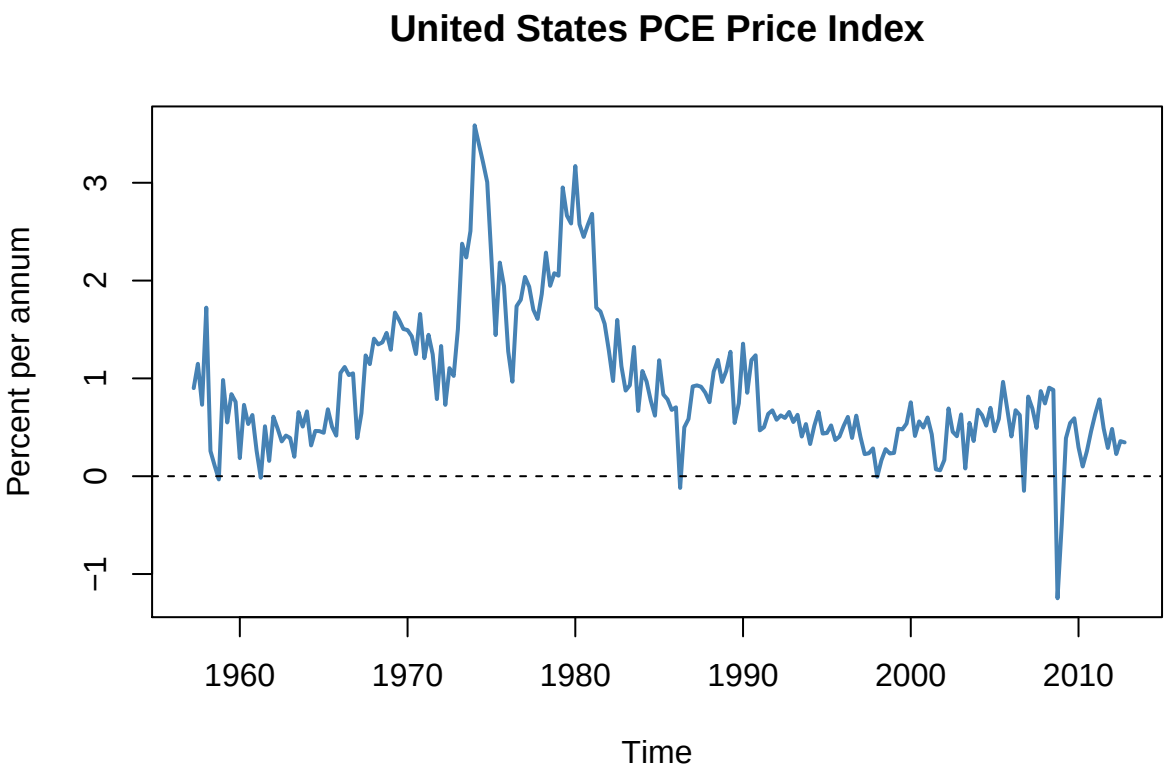
Following the book, we take the price level of the united states measured by the personal consumption expenditures price index as an example.

```
# define ts object of the United States PCE Price Index
PCECTPI <- ts(log(USMacroSWQ$PCECTPI), start = c(1957,1), end = c(2012,4), freq = 4)

# plot logarithm of the PCE Price Index
plot(log(PCECTPI),
     main = "Log of United States PCE Price Index",
     ylab = "Logarithm",
     col = "steelblue", lwd = 2)
```



We see that the logarithm of the price level has a smoothly varying trend. This is typical for an $I(2)$. If the price level is indeed $I(2)$, the first differences of this series should be $I(1)$. Since we are considering the logarithm of the price level, we obtain growth rates by taking first differences, so the differenced price level series is the series of quarterly inflation rates. This is quickly done in R using the function Delt() from the package quantmod As explained in Chapter 14.2, multiplying the quarterly inflation rates by 400 yields the quarterly rate of inflation, measuared in percentage points at an annual rate.

```
# plot United States PCE price inflation
plot(400*Delt(PCECTPI),
     main = "United States PCE Price Index",
     ylab = "Percent per annum",
     col = "steelblue", lwd = 2)
abline(0,0, lty = 2)
```



Obviously, the rate of inflation behaves much more erratic than the smooth graph of the logarithm of the PCE price index.

17.2.0.0.1 The DF-GLS Test for a Unit Root

The DF-GLS test for a unit root has been developed by Elliot, Rothenberg and Stock (1996) and has higher power than the ADF test when the autoregressive root is large but less than one. That is, the DF-GLS has a higher probability of rejecting the false null hypothesis of a stochastic trend when the sample data stems from time series that is close to being integrated.

The idea of the DF-GLS test is to test for an autoregressive unit root in the detrended series whereby GLS estimates of the deterministic components are used to obtain the detrended version of the original series. See Chapter 16.3 of the book for a more detailed explanation of the approach.

A function that performs the DF-GLS test is implemented in the package urca (this package is a dependency of the package vars so it should be already loaded if vars is attached). The function that computes the test statistic is ur.ers.

```
# DF-GLS test for unit root in GDP
summary(
 ur.ers(log(window(GDP, start = c(1962,1), end = c(2012,4))),
        model = "trend",
        lag.max = 2)
)
```

```
##
## ###############################################
## # Elliot, Rothenberg and Stock Unit Root Test #
## ###############################################
##
## Test of type DF-GLS
```

```
## detrending of series with intercept and trend
##
##
## Call:
## lm(formula = dfgls.form, data = data.dfgls)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.025739 -0.004054  0.000017  0.004619  0.033620
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## yd.lag       -0.01213    0.01012  -1.199  0.23207
## yd.diff.lag1  0.28583    0.07002   4.082 6.47e-05 ***
## yd.diff.lag2  0.19320    0.07058   2.737  0.00676 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007807 on 198 degrees of freedom
## Multiple R-squared:  0.1504, Adjusted R-squared:  0.1376
## F-statistic: 11.69 on 3 and 198 DF,  p-value: 4.392e-07
##
##
## Value of test-statistic is: -1.1987
##
## Critical values of DF-GLS are:
##                  1pct  5pct 10pct
## critical values -3.48 -2.89 -2.57
```

Inspecting the summary of the test we find that the test statistic is about -1.2 which is larger than the 10% critical value of -2.57 (this is the appropriate critical value for the ADF test when an intercept and a time trend are included in the Dickey-Fuller regression and the test is left-sided) so we cannot reject the null hypothesis that inflation is nonstationary, using the DF-GLS test.

17.3 Cointegration

Key Concept 16.5

Cointegration

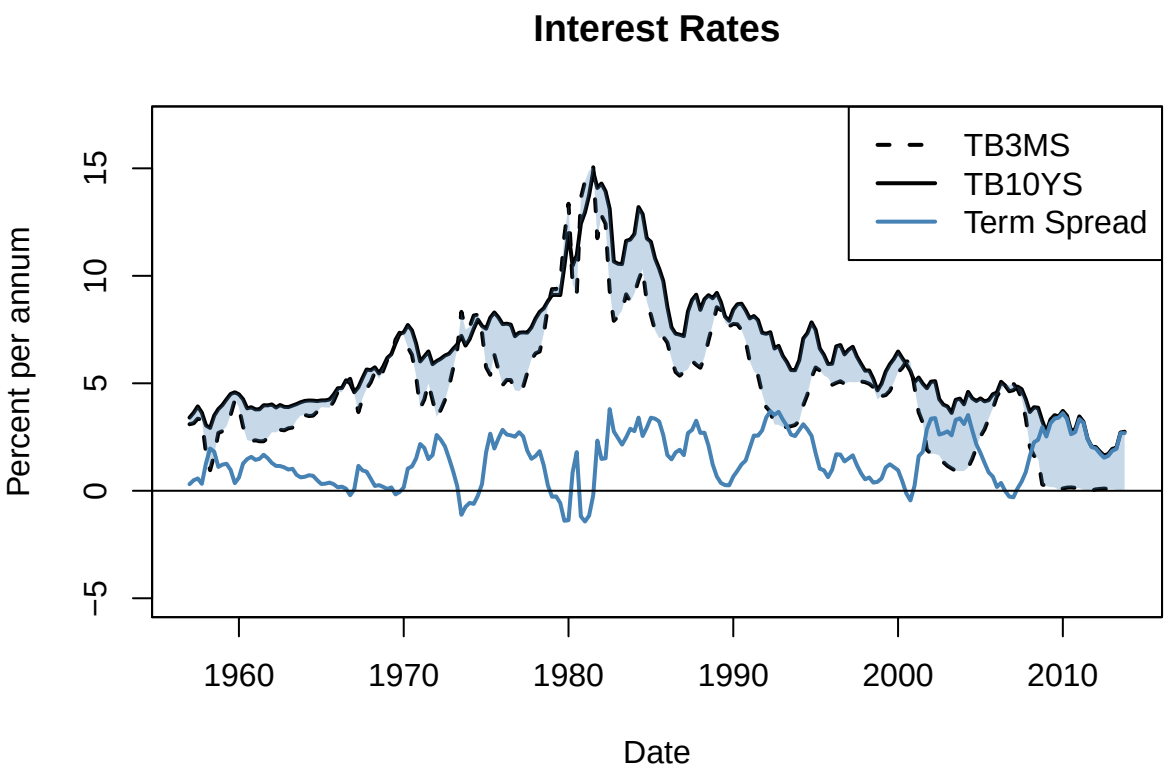
When X_t and Y_t are $I(1)$ and if there is a θ such that $Y_t - \theta X_t$ is $I(0)$, we say that X_t and Y_t are cointegrated. Put differently, cointegration of X_t and Y_t means that X_t and Y_t have the same or a common stochastic trend and this trend can be eliminated by some difference of the series such that the result is stationary.

R functions for cointegration testing are implemented in the package urca.

As an example, reconsider the the relation between short- and long-term interest rates by the example of U.S. 3-month treasury bills, U.S. 10-years treasury bonds and the spread in their interest rates which have been introduced in Chapter 14.4. The next code chunks shows how to reproduce figure 16.2 of the book.

```
# reproduce figure 16.2 of the book

# plot both interest series
plot(merge(as.zoo(TB3MS),as.zoo(TB10YS)),
     plot.type = "single",
```

```
     lty = c(2,1),
     lwd = 2,
     xlab = "Date",
     ylab = "Percent per annum",
     ylim = c(-5, 17),
     main = "Interest Rates"
)

# add the term spread
lines(as.zoo(TSpread),
     col = "steelblue",
     lwd = 2,
     xlab = "Date",
     ylab = "Percent per annum",
     main = "Term Spread"
)

# shade term spread
polygon(c(time(TB3MS), rev(time(TB3MS))),
        c(TB10YS, rev(TB3MS)),
        col = alpha("steelblue", alpha = 0.3),
        border = NA
        )

# add horizontal line add 0
abline(0,0)

# add a legend
legend("topright",
       legend = c("TB3MS","TB10YS","Term Spread"),
       col = c("black", "black", "steelblue"),
       lwd = c(2,2,2),
       lty = c(2,1,1)
       )
```



The plot suggests that long-term and short-term interest rates are cointegrated: notice that both interest series seem to have the same long-run behaviour. In fact, they share a common stochastic trend. The term spread which is obtained by taking the difference between long-term and short-term interest rates seems to be stationary. In fact, the expectations theory of the term structure suggests the cointegrating coefficient θ to be 1. This is consistent with the visual results.

17.3.0.0.1 Testing for Cointegration

Folloing Key Concept 16.5, it seems natural to construct a test for cointegration of two series in the following manner: if two series X_t and Y_t are cointegrated, the series obtained by taking the difference $Y_t - \theta X_t$ must be stationary. If the are not cointegrated, $Y_t - \theta X_t$ is nonstationary. This is an assumption that can be tested using a unit root test. We generally have to distinguish between two cases:

1. **θ is known.**

   Knowledge of θ enables us to compute differences $z_t = Y_t - \theta X_t$ so that Dickey-Fuller and DF-GLS unit root tests can be applied to z_t. For these tests, the critical values are the critical values of the ADF test.

2. **θ is unknown.**

   If θ is unknown it must be estimated before the unit root test can be applied. This is done by estimating the regression

   $$Y_t = \alpha + \theta X_t + z_t$$

   using OLS (this is refered to as the first stage regression). Then, a Dickey-Fuller test is used for testing the hypothesis that z_t is a nonstationary series. This is known as the Engle-Granger Augmented Dickey-Fuller test for cointegration (or **EG-ADF test**) after Engle and Granger (1987). The critical values for this test are special as the associated null distribution is nonnormal and depends on the number of $I(1)$ variables used as regressors in the first stage regression. You may look them up in Table 16.2 of the book. When there are only two cointegrated variables (and thus a single $I(1)$ variable

used in the first stage OLS regression) the critical values for the levels 10%, 5% and 1% are -3.12, -3.41 and -3.96.

Application to Interest Rates

As has been mentioned above, the theory of the term structure suggests that long-term and short-term interest rates are cointegrated with a cointegration coefficient of $\theta = 1$. In the previous section we have seen that there is visual evidence for this conjecture since the spread of 10-year and 3-month interest rates looks stationary.

We now continue by using formal tests (the ADF and the DF-GLS test) to see whether the individual interest rate series are integrated and if their difference is stationary (for now, we assume that $\theta = 1$ is known). Both is conveniently done by using the functions ur.df() for computation of the ADF test and ur.ers for the DF-GLS test. Following the book we use data from 1962:Q1 to 2012:Q4 use models that include a drift component. We set the maximum lag order to five and use the *AIC* for selection of the optimal lag length. Remember that the Dickey-Fuller critical values from table 15.1 are appropriate for both tests.

```
# test for nonstationarity of 3-month treasury bills using ADF test
ur.df(window(TB3MS, c(1962,1), c(2012,4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -2.1004 2.2385
```

```
# test for nonstationarity of 10-years treasury bonds using ADF test
ur.df(window(TB10YS, c(1962,1), c(2012,4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -1.0079 0.5501
```

```
# test for nonstationarity of 3-month treasury bills using DF-GLS test
ur.ers(window(TB3MS, c(1962,1), c(2012,4)),
       model = "constant",
       lag.max = 6)
```

```
##
## ###############################################################
## # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -1.8042
```

```
# test for nonstationarity of 10-years treasury bonds using DF-GLS test
ur.ers(window(TB10YS, c(1962,1), c(2012,4)),
```

```
       model = "constant",
       lag.max = 6)
```

```
##
## ###############################################################
## # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -0.942
```

The corresponding 10% critical value for both tests is -2.57 so we cannot reject the null hypotheses of nonstationary for either the series, even at the 10% level of significance.[1] We conclude that it is plausible to model both interest rate series as $I(1)$.

Next, we may apply the ADF and the DF-GLS test to test for nonstationarity of the term spread series, which means we test for non-cointegration of long- and short-term interest rates with cointegration coefficient $\theta = 1$.

```
# test if term spread is stationairy (cointegration of interest rates) using ADF
ur.df(window(TB10YS, c(1962,1), c(2012,4)) - window(TB3MS, c(1962,1), c(2012,4)),
      lags = 6,
      selectlags = "AIC",
      type = "drift")
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -3.9308 7.7362
```

```
# test if term spread is stationairy (cointegration of interest rates) using DF-GLS test
ur.ers(window(TB10YS, c(1962,1), c(2012,4)) - window(TB3MS, c(1962,1), c(2012,4)),
       model = "constant",
       lag.max = 6)
```

```
##
## ###############################################################
## # Elliot, Rothenberg and Stock Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -3.8576
```

The following table summarizes the results.

Series	ADF Test Statistic	DF-GLS Test Statistic
TB3MS	-2.10	-1.80
TB10YS	-1.01	-0.94
TB10YS - TB3MS	-3.93	-3.93

We find that both test reject the hypothesis on nonstationarity of the term spread series at the 1% level of significance which is strong evidence in favour of the hypothesis that the term spread is stationary, implying

[1] Note: ur.df() reports two test statistics when there is a drift in the ADF regression. The first of which (the one we are interested in here) is the t-statistic for the test that the coefficient on the first lag of the series is 0. The second one is the t-statistic for the hypothesis test that the drift term equals 0.

cointegration of long- and short-term interest rates.

Since theory suggests that $\theta = 1$ there is no need to estimate θ so it is not necessary to use the EG-ADF test which assumes θ to be unknown. However, since it is instructive to do so we follow the book and compute the test statistic. The first stage OLS regression is

$$TB10YS_t = \beta_0 + \beta_1 TB3MS_t + z_t.$$

```
# Estimate first stage regression of EG-ADF test
FS_EGADF <- dynlm(window(TB10YS, c(1962,1), c(2012,4)) ~ window(TB3MS, c(1962,1), c(2012,4)))
FS_EGADF
```

```
##
## Time series regression with "ts" data:
## Start = 1962(1), End = 2012(4)
##
## Call:
## dynlm(formula = window(TB10YS, c(1962, 1), c(2012, 4)) ~ window(TB3MS,
##     c(1962, 1), c(2012, 4)))
##
## Coefficients:
##                              (Intercept)
##                                   2.4642
## window(TB3MS, c(1962, 1), c(2012, 4))
##                                   0.8147
```

So we have

$$\widehat{TB10YS_t} = 2.46 + 0.81 TB3MS_t$$

where $\widehat{\theta} = 0.81$. Next, we take the residual series $\{\widehat{z}_t\}$ and compute the ADF test statistic.

```
# compute residuals
z_hat <- resid(FS_EGADF)

# compute ADF test statistic
ur.df(z_hat, lags=6, type = "drift", selectlags = "AIC")
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -3.1882 5.0859
```

The test statistic is -3.19 which is smaller than the 10% critical value but larger than the 5% critical value (see table 16.2 of the book). Thus, the null of no cointegration can be rejected at the 10% level but not at the 5% level. As pointed out by Stock & Watson, this indicates lower power of the EG-ADF test due to the estimation of θ: when $\theta = 1$ is the correct value, we expect the power of the ADF test for a unit root in the residuals series $\widehat{z} = TB10YS_t - \widehat{\beta}_0 - TB3MS$ to be higher than when some estimate $\widehat{\theta}$ is used.

17.3.0.0.2 A Vector Error correction model for $TB10YS_t$ and $TB3MS$

If two $I(1)$ time series X_t and Y_t are cointegrated, their differences are stationary and can modeled in a VAR which is augmented by the regressor $Y_{t-1} - \theta X_{t-1}$. This is called a **vector error correction model**

(VECM) and $Y_t - \theta X_t$ is called the **error correction term**. Lagged values of the error correction term are useful for predicting ΔX_t and/or ΔY_t.

A VECM can be used to model the two interest rates considered in the previous sections. Following the book we specify the VECM to include 2 lags of both series as regressors and choose $\theta = 1$, as theory suggests (see above.)

```
TB10YS <- window(TB10YS, c(1962,1), c(2012,4))
TB3MS <- window(TB3MS, c(1962,1), c(2012,4))

# set up error correction term
VECM_ECT <- TB10YS - TB3MS

# estimate both equations of the VECM using 'dynlm()'
VECM_EQ1 <- dynlm(d(TB10YS) ~ L(d(TB3MS), 1:2) + L(d(TB10YS), 1:2) + L(VECM_ECT))
VECM_EQ2 <- dynlm(d(TB3MS) ~ L(d(TB3MS), 1:2) + L(d(TB10YS), 1:2) + L(VECM_ECT))

# rename regressors for better readability
names(VECM_EQ1$coefficients) <- c("Intercept","D_TB3MS_l1","D_TB3MS_l2",
                                  "D_TB10YS_l1","D_TB10YS_l2","ect_l1")
names(VECM_EQ2$coefficients) <- names(VECM_EQ1$coefficients)

# HAC coefficient summaries
coeftest(VECM_EQ1, vcov. = NeweyWest(VECM_EQ1, prewhite = F, adjust = T))
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## Intercept    0.1227089  0.0551419  2.2253 0.027205 *
## D_TB3MS_l1  -0.0016601  0.0727060 -0.0228 0.981807
## D_TB3MS_l2  -0.0680845  0.0435059 -1.5649 0.119216
## D_TB10YS_l1  0.2264878  0.0957071  2.3665 0.018939 *
## D_TB10YS_l2 -0.0734486  0.0703476 -1.0441 0.297740
## ect_l1      -0.0878871  0.0285644 -3.0768 0.002393 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(VECM_EQ2, vcov. = NeweyWest(VECM_EQ2, prewhite = F, adjust = T))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## Intercept   -0.060746   0.107937 -0.5628  0.57422
## D_TB3MS_l1   0.240003   0.111611  2.1504  0.03276 *
## D_TB3MS_l2  -0.155883   0.153845 -1.0132  0.31220
## D_TB10YS_l1  0.113740   0.125571  0.9058  0.36617
## D_TB10YS_l2 -0.147519   0.112630 -1.3098  0.19182
## ect_l1       0.031506   0.050519  0.6236  0.53359
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus the two estimated equations of the VECM are

$$\begin{aligned}\widehat{\Delta TB3MS_t} = &- \underset{(0.11)}{0.06} + \underset{(0.11)}{0.24}\,\Delta TB3MS_{t-1} - \underset{(0.15)}{0.16}\,\Delta TB3MS_{t-2} \\ &+ \underset{(0.13)}{0.11}\,\Delta TB10YS_{t-1} - \underset{(0.11)}{0.15}\,\Delta TB10YS_{t-2} + \underset{(0.05)}{0.03}\,ECT_{t-1} \\ \widehat{\Delta TB10YS_t} = &\underset{(0.06)}{0.12} - \underset{(0.07)}{0.00}\,\Delta TB3MS_{t-1} - \underset{(0.04)}{0.07}\,\Delta TB3MS_{t-2} \\ &+ \underset{(0.10)}{0.23}\,\Delta TB10YS_{t-1} - \underset{(0.07)}{0.07}\,\Delta TB10YS_{t-2} - \underset{(0.03)}{0.09}\,ECT_{t-1}.\end{aligned}$$

The output produced by coeftest() shows that there is little evidence that lagged values of the differenced interest series' are useful for prediction. This finding is more pronounced for the equation of the differenced series of the 3-month treasury bill rate where the error correction term (the lagged term spread) is not significantly different from zero at any common level of significance. However, for the model equation of the differenced 10-years treasury bonds rate we find that the error correction term is statistically significant at the 1% level with an estimate of -0.09. This can be interpreted as follows: although both interest rates are nonstationary, their conintegrating relationship allows to predict the *change* in the 10-years treasury bonds rate using the VECM. In particular, the negative coefficient estimate on the error correction term indicates that there will be a negative change in the next period's 10-years treasury bonds rate when the 10-years treasury bonds rate is higher than the 3-month treasury bill rate in the current period.

17.4 Volatility Clustering and Autoregressive Conditional Heteroskedasticity

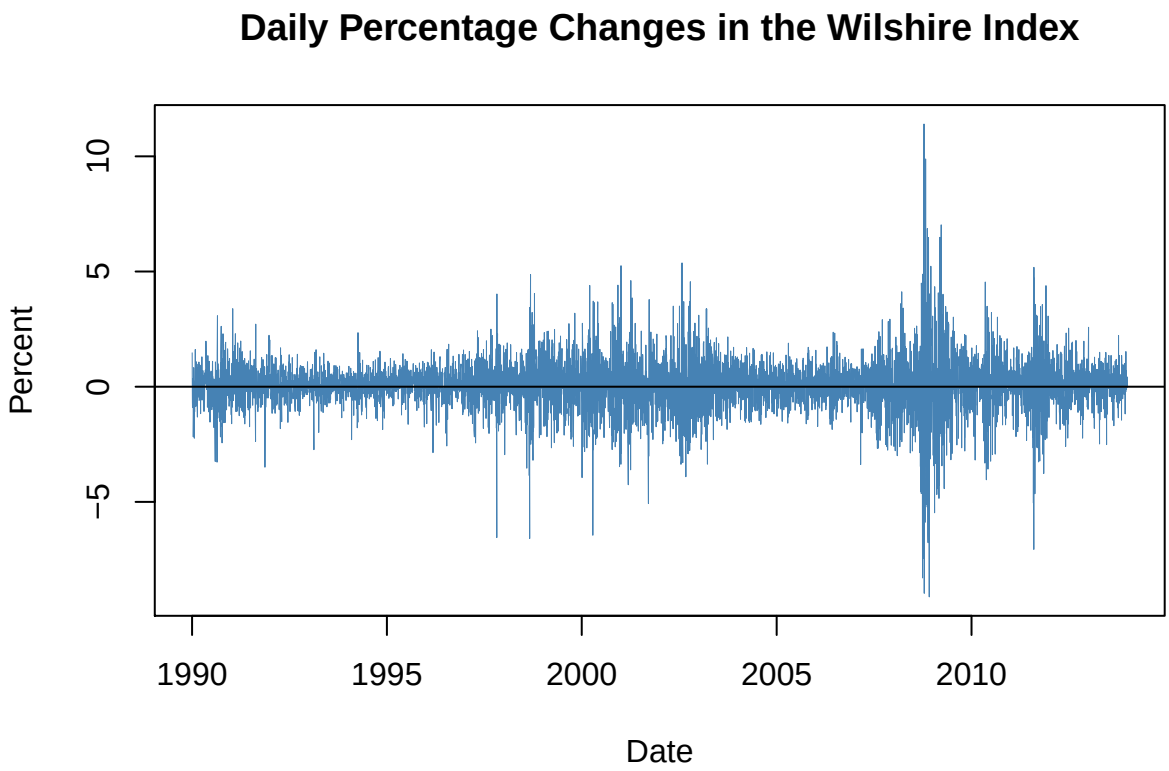
In particular financial time series often exhibit a behaviour that is known as **volatility clustering**: the volatility changes over time but its degree shows a tendency to persist i.e. there are periods of low volatility and some where volatility is high. Econometricians call this the conditional heteroskedasticity. Conditional heteroskedasticity is an intersting property because it can be exploited for forecasting the variance of future periods. A good forecast of a series' variance allows to quantify uncertainty (for example by computing accrurate forecast confidence intervals) which is especially important when it is hard to come up with a good prediction of future realisations (see section 14.4).

As an example, we consider daily changes in the Whilshire 5000 stock index. The data is available for download at the Federal Reserve Economic data Base. For consistency with the book we download data from 1989-29-12 to 2013-12-31 (choosing this somewhat larger time span is necessary since later on we will be working with daily changes of the series).

The following code chunk shows how to format the data accordingly and how to reproduce Figure 16.3 of the book.

```
# import data on Wilshire 5000 index
W5000 <- read.csv2("data/Wilshire5000.csv", stringsAsFactors = F, header = T, sep = ",", na.strings = "

# convert columns
W5000$DATE <- as.Date(W5000$DATE)
W5000$WILL5000INDFC <- as.numeric(W5000$WILL5000INDFC)

# remove NAs
W5000 <- na.omit(W5000)

# daily percentage changes
W5000_PC <- data.frame("Date" = W5000$DATE, "Value" = as.numeric(Delt(W5000$WILL5000INDFC)*100))
W5000_PC <- na.omit(W5000_PC)
```

```
# plot percentage changes
plot(W5000_PC,
     ylab = "Percent",
     main = "Daily Percentage Changes in the Wilshire Index",
     type="l", col = "steelblue", lwd = 0.5)
abline(0,0)
```



The level of the series of daily percentage changes in the Wilshire Index seems to randomly fluctuate near zero, meaning there is little amount autocorrelation so that it is difficult to predict future outcomes using e.g. an autoregressive model. However there is visual evidence that the series exhibits conditional heteroskedasticity since we observe volatility clustering and for some applications it is useful to measure and forecast these patterns. This can be done using models which assume that the volatility can be described by an autoregressive processe.

ARCH and GARCH Models

Consider

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \gamma_1 X_{t-1} + u_t$$

, a simple ADL(1,1) regression model. The econometrician Robert Engle (1982) proposed to model σ_t^2, the variance of the normally distributed error u_t by an order p distributive lag model

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \cdots + \alpha_p u_{t-p}^2, \tag{17.1}$$

which is called the **autoregressive conditional heteroskedasticity** (ARCH) model of order p, or short ARCH(p).[2] The general idea is apparent from the model structure: positive coefficients $\alpha_0, \alpha_1, \ldots, \alpha_p$ imply that recent large squared errors lead to a large variance, and thus large squared errors, in the current period.

[2] Although we introduce the ARCH model as a component in an ADL(1,1) model (a model of the conditional error variances), it can be used for modelling the conditional zero-mean error term of any time series model.

The generalized ARCH (GARCH) model, developed by Tim Bollerslev (1986) is an extension of the ARCH model where σ_t^2 is allowed to depend on its on lags and lags of the squared error term. The GARCH(p,q) model is given by

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \cdots + \alpha_p u_{t-p}^2 + \phi_1 \sigma_{t-1}^2 + \cdots + \phi_p \sigma_{t-q}^2. \tag{17.2}$$

The GARCH model is an ADL(p,q) model and thus can provide more parsimonious parameterizations than an ARCH model (see the discussion in Appendix 15.2 of the book).

Application to Stock Price Volatility

Maximum likelihood estimates of ARCH and GARCH models are efficient and have normal distributions in large samples such that the usual methods for conducting inference about the unknown parameters can be applied. The R package fGARCh is a collection of function for analyzing and modelling the heteroskedastic behaviour in time series models. In the following application we reproduce the results presented in Chapter 16.5 of the book by means of the function garchFit(). This function is somewhat more sophisticated. It allows for different specifications of the optimization procedure, different error distributions and much more (use `?GarchFit` for a detailed description of the arguments), in particular, the reported standard errors are robust.

The GARCH(1,1) model of daily changes in the Wilshire 5000 index we estimate is

$$R_t = \beta_0 + u_t \ , \ u_t \sim \mathcal{N}(0, \sigma_t^2) \tag{17.3}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \phi_1 \sigma_{t-1}^2 + \nu_t \tag{17.4}$$

where R_t are is the percentage change in period t, β_1, α_0, α_1 and ϕ_1 are unknown coefficients and u_t and ν_t are error terms with conditional mean of zero. We do not include any lagged predictors in the model of R_t because the daily changes in the Wilshire 5000 index reflect daily stock returns wich are essentially unpredictable. Note that u_t is assumed to be normally distributed and the variance σ_t^2 depends on t as it follows the GARCH(1,1) recursion (17.4).

It is straightforward to estimate this model using garchFit().

```
# estimate GARCH(1,1) model of daily percentage changes
GARCH_Wilshire <- garchFit(data = W5000_PC$Value, trace = F)
```

We obtain

$$\widehat{R}_t = \underset{(0.010)}{0.068} \tag{17.5}$$

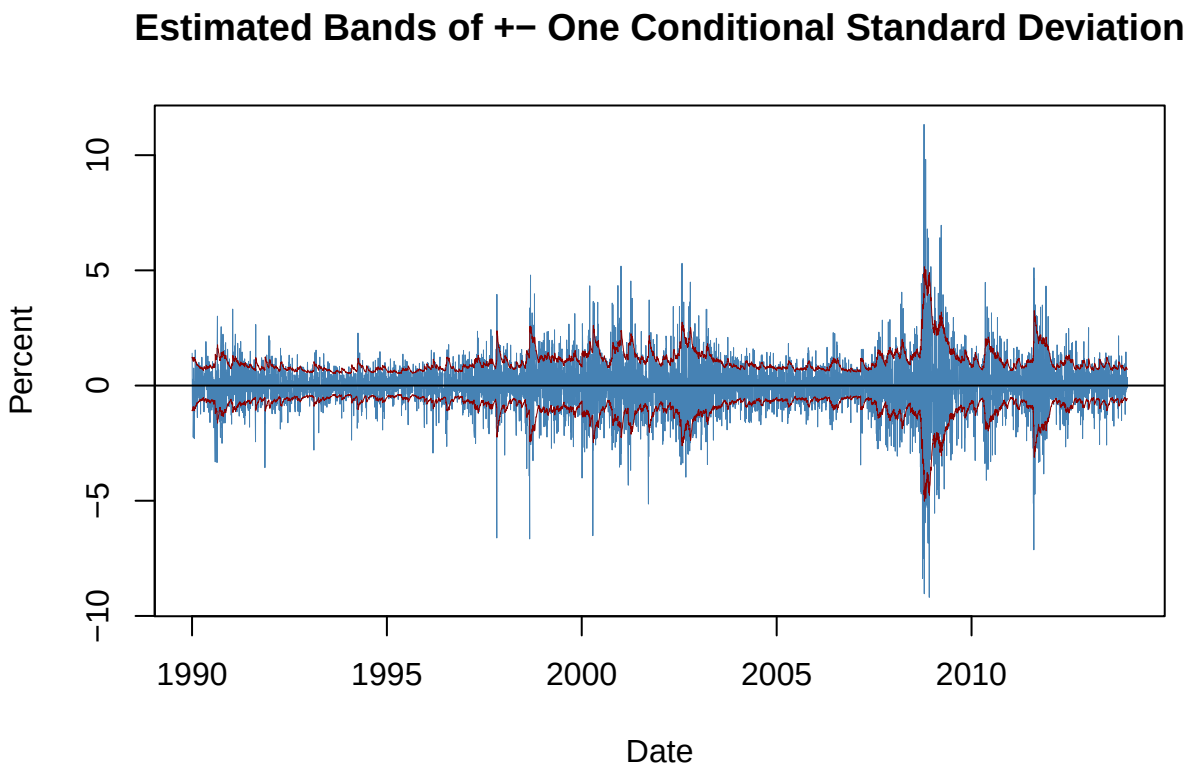
$$\widehat{\sigma}_t^2 = \underset{(0.002)}{0.011} + \underset{(0.007)}{0.081} u_{t-1}^2 + \underset{(0.008)}{0.909} \sigma_{t-1}^2 \tag{17.6}$$

so the coefficient on u_{t-1}^2 and σ_{t-1}^2 are statistically significant at any common level of significance. One can show that the persistence of movements in σ_t^2 is determined by the sum of both coefficients which is 0.99 here. This indicates that movements in the conditional variance are highly persistent, implying long-lasting periods of high volatility which is consistent with the visual evidence for volatility clustering presented above.

The estimated conditional variance $\widehat{\sigma}_t^2$ can be computed by plugging the residuals from (17.5) into equation (17.6). This is performed automatically by garchFit() so to obtain a vector of the estimated conditional standard deviations all we have to do is to read out the values from `GARCH_Wilshire` by appending `@sigma.t`.

Using the vector containing the $\widehat{\sigma}_t$ we plot bands of $\pm$ one conditional standard deviation along with deviations of the series of daily percentage changes in the Wilshire 5000 index from its mean. The following code chunk reproduces Figure 16.4 of the book.

```
# compute deviations of the percentage changes from their mean
dev_mean_W5000_PC <- W5000_PC$Value - GARCH_Wilshire@fit$coef[1]

# plot deviation of percentage changes from mean
plot(W5000_PC$Date, dev_mean_W5000_PC,
     type = "l",
     col = "steelblue",
     ylab = "Percent",
     xlab = "Date",
     main = "Estimated Bands of +- One Conditional Standard Deviation",
     lwd = 0.2)
abline(0, 0)

# add GARCH(1,1) confidence bands (one standard deviation) to the plot
lines(W5000_PC$Date, GARCH_Wilshire@fit$coef[1] + GARCH_Wilshire@sigma.t, col = "darkred", lwd = 0.5)
lines(W5000_PC$Date, GARCH_Wilshire@fit$coef[1] - GARCH_Wilshire@sigma.t, col = "darkred", lwd = 0.5)
```



We see that the bands of estimated conditional standard deviation describe the observed heteroskedasticity in the series of daily changes of the Wilshire 5000 index quite well and are useful for quantifying time-varying volatility and the emerging risk for investors holding stocks summarized by the index. This estimated GARCH model may also be used to produce forecast intervals whose widths depend on the volatility of the most recent regression residuals.

17.5 Summary

- We have discussed how vector autoregressions are conveniently estimated and used for forecasting in R by means of functions from the VARS package.
- The package urca which is a dependency of VARS supplies advanced methods for unit root and cointegration analysis like the DF-GLS and the EG-ADF tests. In an application we have found evidence that 3-months and 10-year interest rates have a common stochastic trend (that is they are cointegrated) and thus can be modelled using a vector error correction model.
- At last we have introduced the concept of volatility clusering and demonstrated how the function garchFit() from the package fGarch can be employed to estimate a GARCH(1,1) model of the conditional heteroskedasticity inherent to returns of the Wilshire 5000 stock index.