Working With Descriptive and Inferential Statistics in R

Chair of Political Methodology
Institut für Politikwissenschaft
University of Zurich

Fall 2015[1]

[1]Compiled and tested with 'checkpoint': `2017-10-16`.

Contents

Preface

This book introduces R in the context of the angewandte Methoden der Politikwissenschaft. The objective is to turn you into a comfortable R user, someone who is able to perform basic data manipulation and who can obtain basic graphs, tables, descriptive statistics, and who can perform basic hypothesis tests. Based on this foundation, you can then develop expert R skills.

By now, R is one of the most widely used statistical programming platforms in political science. There are several reasons for this. Most important perhaps is R's versatility. There is very little that R cannot do and cutting edge developments in statistical methodology are often first introduced in R. This has a major advantage: R is probably the only statistical program that you will ever have to learn. A second major benefit of R is that it produces amazing graphics, better than any other statistical package. Third, all of these benefits are available for all of the major operating systems and they are available for free!

This may all sound to good to be true—there has to be a catch. Well, there sort of is. R is not an easy program to master. The learning curve is steep and there will be times that you will frankly be quite frustrated with the program, wondering what sin you committed that you have been condemned to learning R. This is where the present work book comes into play. In this book, we try to give you a sense of the architecture of R. What is more, we also provide you with recipes for tackling common tasks such as reading and saving data, recoding old variables, and creating new ones. Our sincere hope is that this information will take some of the frustration out of learning R. Once you master the basic tricks, we are convinced that you will find R to be quite wonderful, albeit perhaps not to the extent that us method geeks enjoy the program.

The work book is organized as follows. Each chapter starts off with a worked example illustrating how to perform a particular task in R. You can read this example ahead of the exercise session and, if you want, you can even replicate the example by running the code yourself. Indeed, this is the recommended strategy for preparing yourself for the exercise sessions. The worked example is followed by a new set of problems, which are similar in nature but require that you write your own code. The worked examples serve as a template for these new problems, but you will have to make adjustments to answer the questions. The workbook does not walk through the new problem sets but the lab sessions will review the solutions.

We hope that this work book will proof to be helpful with learning R. Although R requires quite an investment of time initially, the benefits you will reap more than compensate for this. Moreover, with the help of this book, the investment will hopefully be smaller. Hence, we wish you all much pleasure with mastering R.

Zurich, August 2015
Marco R. Steenbergen
Kushtrim Veseli
Benjamin Schlegel

Part I

Basics

Chapter 1

The R-chitecture

The statistical program R has been around since 1993. In some ways, however, the program's roots go back to 1976, when its precursor S was developed at Bell Labs in the United States. Behind the power of R stands a core team of developers. In addition, however, users have contributed numerous add-ons in the form of so-called **libraries**. There also is an improved graphical user interface (GUI), called Rstudio, which sits on top off R and greatly simplifies programming in the R language. The latest versions even includes an auto-complete of R syntax.

1.1 Finding and Installing the Software

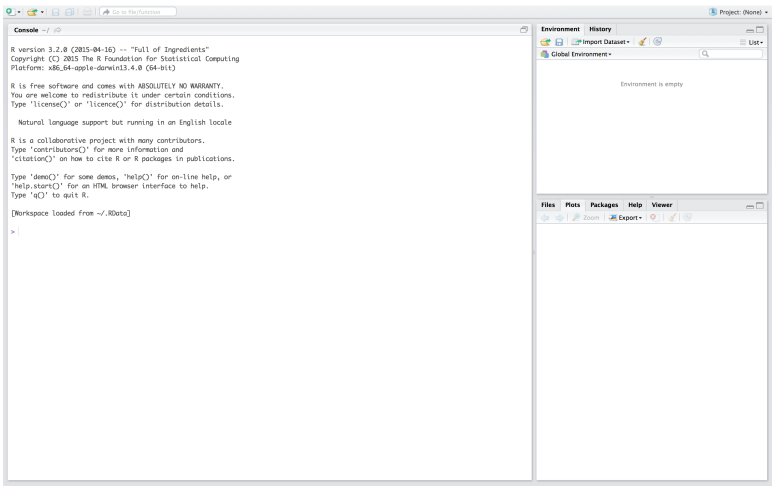
The R software can be found on the web site of the Comprehensive R Network Archive, or CRAN for short, which also is the repository of most of the user written libraries. The software is available for OSX (Mac), Unix, as well as Windows. Just find the version that is right for your operating system and follow the installation instructions.

Rstudio is a GUI for working with R. It can be downloaded—again for free and for all major operating systems—at www.rstudio.org. There are two versions, one for commercial use and one for individual use, and you should pick the latter, as this is the free version. Find the version that is right for your operating system and follow the installation instructions. Important: Install Rstudio only after you have installed R.

1.2 The Rstudio Environment

When you open Rstudio for the first time, you will notice that there are three windows (see Figure 1.1). The largest window appears on the left and is called the **console**. This is where you can write syntax and where the results—as well as the occasional error message—will be displayed. On the top right, you will find a window with two tabs: Environment and History. The Environment tab

Figure 1.1: The Rstudio Environment



Note: When you first open Rstudio, the desktop is divided into three sections: (1) the console on the left; (2) the Environment/History window on the top right; and (3) miscellaneous windows, including Help, on the bottom right.

shows all of the objects that are available to you (more on that below). The History tab shows the commands that you have issued. It is useful because it allows you to copy-paste commands and thus eliminates the need to re-type them. The final window appears at the bottom right and has multiple tabs: Files, Plots, Packages, Help, and Viewer. Files, as the name implies, shows all of the files in the current directory. Plots will not show anything when you start. However, any plots that you generate can be found here. The same is true for the Viewer tab, which becomes relevant if you are interested in accessing local web content. Packages is a tab that we will be using frequently. This is where you install, update, and activate libraries. It is so important that we dedicate an entire section to this tab. Finally, the Help tab provides help, as the name suggests. One of the very nice aspects of Rstudio is that it is much easier to get access to the help files than in R itself. If you are ever uncertain what command to use or how to use it, then the Help tab is your best friend.

1.3 R—The Expandable Language

Most R users describe the program as a programming language rather than a statistical package. A statistical package performs a limited set of tasks but R is much more versatile than that. It can be used to produce documents such as this book, HTML code, animations and much more. It can be parallelized to speed up computations. And it interfaces with other programming languages

such as C++.

Much of this added functionality comes from user-written programs, which R calls libraries or, alternatively, packages. Libraries are collections of commands, sometimes tightly connect to a particular methodology and sometimes more versatile (e.g., a set of utilities). There are two things you need to know about libraries. First, when you install R it comes with a very limited number of libraries. Second, when you start up R or Rstudio, then only a very limited number of libraries will be active. This means that you will have to activate most libraries if you want to use them.

1.3.1 Installing Libraries

All of the libraries that we shall need in angewandte Methoden can be found on CRAN. By far the easiest way of installing these packages is to utilize the Packages tab in Rstudio. When you open this tab, you will notice two options: Install and Update. Update is for libraries that you have already installed and is useful for updating them. Install is for adding new libraries. If you click on the install option, a new window appears (see Figure 2). By default, Rstudio will search for libraries on CRAN. The only information you need to enter is the name of he library you want to install. It is usually quite easy to find this out via a search on the Internet and, in any case, we will provide the names of all of the packages that you will need. Important is to make sure that the box titled "install dependencies" is checked. Many libraries themselves rely on other libraries, which may not yet be in your system library. By clicking the box, you ensure that these libraries are also installed so that everything will run smoothly. When you now click on Install, you'll see some activity in the console indicating that the libraries are being installed.

1.3.2 Activating Libraries

There are several ways in which an R library can be activated. One way is to go into the Packages tab and click the box next to the package name. Alternatively, you could type `library` or `require` to activate the package. As an example consider the following two commands:

```
library(foreign)
require(foreign)
```

Both of these options activate the `foreign` library, which allows you to read in data in a non-native format such as `.csv`. Whereas installation is forever—you only need to install a library once—activation is ephemeral. It applies only to the current session and once you exit R or Rstudio, you will have to activate the library again.

1.3.3 Deactivating Libraries

Occasionally, you will find that different libraries do not play together well. This happens, for example, when a library calls on an earlier version of another library, while you have loaded a more recent version of that library. You can often tell there is a problem because you get a warning or the library of interest fails to produce results. Normally, libraries are unloaded only after you exit R. To unload them without interrupting the R session, you can use the `detach` function. For example, to detach the `foreign` library, we could issue the command

```
detach("package:foreign", unload = TRUE)
```

This will deactivate the foreign library, should that interfere with the normal functioning of another library.

1.3.4 Maintaining Libraries

Libraries are updated continuously. It is good practice to check for updates regularly, say once a month. It is easy to do this in `RStudio`. Just go to the Packages tab and click on Update. This will guide you through a simple process for updating your libraries.

1.4 R—The Object-Oriented Language

R is an object-oriented language. This means that you create objects. These can be data, matrices and arrays, results from functions, or objects created by commands. All of the objects are visible in the Environment tab in Rstudio. At any point in time, you may have just one or dozens of objects available to you. This is a major difference compared to other statistical programs, which usually allow only for one object, to wit the data.

As an example, consider a simple computation: $2 + 3$. We could enter this as

```
2 + 3

## [1] 5
```

In this case, we obtain the result directly in the console window. However, we could also type the following:

```
x <- 2 + 3
```

Now no result is shown. Instead, the result is stored in the object x, which now appears in the Environment tab, ready for future use. The arrow assigns 2 plus 3 to the object x. Alternatively, we might also have typed

```
x = 2 + 3
```

although this work book will rely mostly on arrows, since these can be used under a wide variety of conditions.

The great benefit of creating objects is that we can store the results from statistical procedures and functions for future use. We shall be doing this a great deal in this book. Future use here can also mean taking the results and turning them into beautiful tables that you can use in papers that you may be writing.

As so often, the flexibility that R offers also brings with it some complications. Imagine that we have two data sets, `bogus1` and `bogus2`, each containing the variable age. If I now ask R to compute the median age, then it does not know where to look. In most statistical programs, this problem would not happen because there is only one data set that is loaded into memory. We can overcome the problem in several ways. For example, we can type

```
median(bogus1$age)

## [1] 50
```

if our interest is in the median age in the first data set. The bit that comes prior to the dollar sign declares the data frame; this is an object we shall discuss in greater detail in the next chapter. The bit that comes after the dollar sign declares the variable of interest. Now R knows precisely which variable in which object it should look at. Alternatively, you can use the `attach` command.

```
attach(bogus1)
median(age)

## [1] 50
```

You now see that it suffices to specify the age variable when we wish to compute the median; it is no longer necessary to also specify the data object. While it would seem that attach is the simpler option, you should keep in mind that R will now only perform operations on the attached data object. If you want to perform operations on a different object, you now have to first detach the original object:

```
detach(bogus1)
```

If you forget to do this, then you may be applying operations to the wrong object. For this reason, we tend to prefer the first approach.

1.5 Types of Objects

R knows three fundamental types of objects: (1) vectors; (2) lists; and (3) functions. It is useful to know the distinctions between those objects, so we

review them briefly.

1.5.1 Vectors

In R, vectors are collections of elements of the same type. The elements may be characters, they may be logical, or numeric, as long as these types are not mixed together. Creating vectors is easy and the following are all examples of the process.

```
a <- 1
b <- c(7411.3, 1035.3, 5614.6, 6886.3, 2825.3)
c <- c("ZH", "BE", "LU", "UR", "SZ", "OW", "NW", "GL", "ZG", "FR",
  "SO", "BS", "BL", "SH", "AR", "AI", "SG", "GR", "AG", "TG", "TI",
  "VD", "VS", "NE", "GE", "JU")
```

On the first line, we create a vector with just one element—mathematicians call this a scalar—to wit the number 1. On the second line, we create a vector of five elements, to with the 2013 per capita GDP values in southern Africa. Note the use of "c," which stands for **concatenate**. This operator allows us to enter multiple elements, together enclosed by parentheses and individually separated by commas. The final three lines result in a 26-element vector, in this case of abbreviations of the Swiss cantons. Since these abbreviations are given in characters, they have to be included in apostrophes. The resulting vector is a string vector.

In R, matrices and arrays are variants of vectors that have a `dim` argument to indicate their dimensionality (e.g., the number of rows and columns). The following is an example that creates a matrix.

```
d <- matrix(c(7411.3, 85, 1035.3, 58, 5614.6, 77, 6886.3, 67,
  3825.3, 98), nrow = 5, byrow = TRUE)
d

##        [,1] [,2]
## [1,] 7411.3   85
## [2,] 1035.3   58
## [3,] 5614.6   77
## [4,] 6886.3   67
## [5,] 3825.3   98
```

The `matrix` command generates the matrix. It declares the elements (using concatinate) as well as the dimensionality (the `nrow` option tells us that there are to be 5 rows and the `byrow = TRUE` option that the elements are to be organized by row). The last line causes R to show the matrix, which we have called `d`. This matrix contains the 2013 per capita GDP values, as well as a measure of political stability. In the next example, we create an array using the same information but for two years: 2008 and 2013.

```
e <- array(c(5660.1, 826.8, 4008.9, 5811.6, 2617.0, 81, 36, 93, 46,
  95, 7411.3, 1035.3, 5614.6, 6886.3, 2825.3, 85, 58, 77, 67, 98),
  dim = c(5, 2, 2))
e
```

```
## , , 1
##
##        [,1] [,2]
## [1,] 5660.1   81
## [2,]  826.8   36
## [3,] 4008.9   93
## [4,] 5811.6   46
## [5,] 2617.0   95
##
## , , 2
##
##        [,1] [,2]
## [1,] 7411.3   85
## [2,] 1035.3   58
## [3,] 5614.6   77
## [4,] 6886.3   67
## [5,] 2825.3   98
```

This array has 5 rows, 2 columns, and 2 slices. The rows indicate countries, whereas the columns capture the two variables (per capita GDP and stability), and each slice reflects a particular time period (2008 or 2013). For example, the 5th country (Swaziland) scored 95 on stability in 2008 and 98 in 2013.

1.5.2 Lists

A list is a collection of elements that may be of different types. Many statistical procedures in R produce results that are lists. It is also not that difficult to create a list yourself, as the following example shows.

```
album <- list(title = 'Kind of Blue', year = 1959, genre = 'Jazz',
  musicians = c('Miles Davis', 'Julian Adderley', 'John Coltrane',
    'Bill Evans', 'Wynton Kelly', 'Paul Chambers', 'Jimmy Cobb'),
  songs = c('So What', 'Freddie Freeloader', 'Blue in Green',
    'All Blues', 'Flamenco Sketches'))
album
```

```
## $title
## [1] "Kind of Blue"
##
## $year
## [1] 1959
```

```
##
## $genre
## [1] "Jazz"
##
## $musicians
## [1] "Miles Davis"     "Julian Adderley" "John Coltrane"
## [4] "Bill Evans"      "Wynton Kelly"    "Paul Chambers"
## [7] "Jimmy Cobb"
##
## $songs
## [1] "So What"            "Freddie Freeloader"
## [3] "Blue in Green"      "All Blues"
## [5] "Flamenco Sketches"
```

You see that the list `album` contains several items: the title, the year it was released, the genre, etc.

One of the most important lists in R is the so-called **data frame**. A data frame is a list consisting of vectors that may be of different types (numeric and string, for example) but are all of the same length. It is the primary representation of data in R. Because it is so important, we dedicate the entire next chapter to data frames.

1.5.3 Functions

Functions constitute the third type of object in R, and are of great importance when using the program. Through the use of functions, we can manipulate data and generate statistical estimates. At a minimum, functions contain two kinds of elements. The first are the arguments to which the function should be applied. The second are the instructions concerning the computations that have to be performed with the arguments. Let us consider a simple example:

```
f1 <- function(x) {
  1/x
}
```

This function has only one argument, to wit x, and its instruction is simply to take the reciprocal of this argument. To apply the function, we type `f1` and declare a particular value of x in parentheses. For example,

```
f1(2)

## [1] 0.5

f1(0)

## [1] Inf
```

In the second example, we try to take the inverse of 0, which is positive infinity or, in R's notation, Inf.

Functions can have multiple arguments. As an example consider the following function, which contains three arguments.

```
hdi <- function(lei, ei, ii) {
  (lei * ei * ii)^(1/3)
}
hdi(0.963, 0.844, 0.950)

## [1] 0.9174114
```

This is the function for the human development index (HDI), which is a geometric mean, as we shall see later. The arguments are the life expectancy index (lei), the education index (ei), and the income index (ii). In the application of the function, we have used the 2013 values of the arguments for Switzerland.

We can also specify multiple computational instructions in a single command. In the following example, we use the raw data for the human development index, which consist of: (1) life expectancy at birth (life); (2) mean years of schooling (educ1); (3) expected years of schooling (educ2); and (4) Gross National Income per capita (gni).[1] The function computes the life expectancy, education, and income indices, as well as the HDI. The sample data are again taken from Switzerland in 2013.

```
hdi2 <- function(life, educ1, educ2, gni) {
  z1 <- (life - 20)/65
  z2 <- .5 * ((educ1/15) + (educ2/18))
  z3 <- (log(gni) - log(100))/(log(75000) - log(100))
  z4 <- (z1 * z2 * z3)^(1/3)
  return(z4)
}
hdi2(82.6, 12.2, 15.7, 53762)

## [1] 0.9168995
```

The `return` argument controls which computations are actually shown. In this case, we have asked only for the final index to be displayed. However, we could easily display all of the intermediate results as well.

```
hdi2 <- function(life, educ1, educ2, gni) {
  z1 <- (life - 20)/65
  z2 <- .5 * ((educ1/15) + (educ2/18))
  z3 <- (log(gni) - log(100))/(log(75000) - log(100))
  z4 <- (z1 * z2 * z3)^(1/3)
```

[1] The formulas for the various components can be found on the website of the United Nations Development Programme.

```
  return(c(z1, z2, z3, z4))
}
hdi2(82.6, 12.2, 15.7, 53762)

## [1] 0.9630769 0.8427778 0.9497103 0.9168995
```

The whole set of results is returned as a vector, with the elements corresponding to the life expectancy, education, income, and human development indices, respectively. We can clarify the output by turning it into an annotated list.

```
hdi2 <- function(life, educ1, educ2, gni) {
  z1 <- (life - 20)/65
  z2 <- .5 * ((educ1/15) + (educ2/18))
  z3 <- (log(gni) - log(100))/(log(75000) - log(100))
  z4 <- (z1 * z2 * z3)^(1/3)
  list(LEI = z1, EI = z2, II = z3, HDI = z4)
}
hdi2(82.6, 12.2, 15.7, 53762)

## $LEI
## [1] 0.9630769
##
## $EI
## [1] 0.8427778
##
## $II
## [1] 0.9497103
##
## $HDI
## [1] 0.9168995
```

1.6 Saving Output

By default, R outputs all of the results to the screen. To obtain a permanent record of the results, one can proceed in two different ways. First, one can just copy-paste the results into a text editor and save them from there. Second, one can use R's `sink` command. Although the command has various advanced options, it usually suffices to decide on two parameters: the name of the output file and whether results should be appended or not. Imagine that we run one analysis that we wish to sink and then another one that we also wish to sink. If we provide the same output file name and set `append = FALSE`, then the first analysis will be overwritten by the second one. If we set `append = TRUE`, however, then the second analysis will be appended to the first one. You should open sink before you execute the command that you want to save. For example,

```
sink("example", append = TRUE)
1+2

## [1] 3

sink()
```

sinks the addition 1 plus 2 to the file `example` in the working directory. The last statement—`sink()`—stops the sinking process, so that subsequent operations will not be saved to a file.

1.7 Getting Help

R offers extensive help when you are uncertain about the syntax. In `RStudio` the help files can be searched by using the "Help" tab in the window on the lower right. If this help does not suffice, then there are several excellent web sites that you can consult. We particularly like Cookbook for R, UCLA's Institute for Digital Research and Education R site, and Quick-R: Accessing the Power of R. If that is still not enough, you should try consulting the "Use R!" book series from Springer.

1.8 Conclusion

In this chapter, we have introduced the basic architecture of R, focusing on the object-oriented and expandable nature of the program. The basic R objects were introduced, to wit vectors, lists, and functions. The interplay of these objects is a key aspect of the R programming environment and is one of the reasons it is so flexible. In the next chapter, we shall focus extensively on one type of list object—the data frame. Before doing so, however, you should try to work through the exercises from this chapter.

1.9 Exercises

(1) Install and activate the `moments` library, which we shall be using in the chapter on summary statistics.

(2) Table 1 contains the percentage of voter turnout in the ten largest cities in Switzerland. First create a vector of city names. Then create a vector of turnout rates.

(3) Generate a function that takes the input x and then computes $x^2/(2-x)$. (Note that the caret symbol means "raise to the power of" whatever follows it.)

Table 1.1: Turnout in the Largest Swiss Cities in 2011

City	Turnout	City	Turnout
Zurich	46.0	Winterthur	47.8
Geneva	42.5	Lucerne	48.2
Basel	50.6	St. Gallen	46.6
Lausanne	40.3	Lugano	50.9
Bern	55.6	Biel	39.0

Note: Data from the Swiss Federal Statistical Office. Turnout in percent of eligible voters.

(4) A person's body mass index or BMI can be computed through the following formula:

$$BMI = \frac{\text{Mass}_{kg}}{\text{Height}^2_m}$$

Write a function in R that takes the measures of mass in kilogram and height in meters as its arguments and returns the BMI. Then apply the function to a person who weighs 110 kilogram and is 2 meters tall. What result do you get?

(5) Building on question (4), there are some countries that measure weight in lbs and height in inches. The conversion is

$$\begin{array}{ll} 1lb & 0.453592kg \\ 1in & 0.0254m \end{array}$$

Expand the function of (4) so that it converts pounds to kilograms and inches to meters, before it computes the BMI. The function should output a list of the kilograms, meters, and BMI. It should take height in inches and weight in pounds as its arguments. Then apply the function to someone who is 66.93 inches tall and weighs 264.56 pounds. What results do you get for the height in meters, weight in kilograms, and the BMI?

Chapter 2

Data Frames

Data are the most important ingredient for a statistical analysis. In R, the data can come in the form of lists, vectors, matrices, and arrays. The most common format, however, is that of a data frame. As objects, data frames are ubiquitous in R and it is important for you to know how to display, create, and save them. Those topics are the focus of this chapter.

2.1 Required Packages

For the exercises in this chapter, you will need to install and/or activate the following libraries:

- `foreign` to read data from SAS, Stata, and other packages.[1]
- `xlsx` to read Excel spreadsheets.

2.2 What Is a Data Frame?

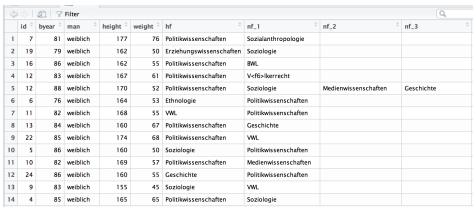
As we discussed in the previous chapter, a data frame is a list consisting of vectors that may be of different types (numeric and string, for example) but are all of the same length. Visually, it looks like a spreadsheet. The rows typically represent (sampling) units, whereas the columns represent variables.

2.3 Displaying the Contents of a Data Frame

In Rstudio, data frames appear in the Environment tab under the rubric "Data." If you want to view the data, you can click on the object name. This opens a new tab in the Console displaying the data in spreadsheet form. The same effect is obtained by typing the `View` command. As an example, assume we have

[1] You should keep in mind that `foreign` may be compatible only with older versions of other programs. Check the help facilities to check the compatibility.

Figure 2.1: View of a Data Frame



	id	byear	man	height	weight	hf	nf_1	nf_2	nf_3
1	7	81	weiblich	177	76	Politikwissenschaften	Sozialanthropologie		
2	19	79	weiblich	162	50	Erziehungswissenschaften	Soziologie		
3	16	86	weiblich	162	55	Politikwissenschaften	BWL		
4	12	83	weiblich	167	61	Politikwissenschaften	Vlkerrecht		
5	12	88	weiblich	170	52	Politikwissenschaften	Soziologie	Medienwissenschaften	Geschichte
6	6	76	weiblich	164	53	Ethnologie	Politikwissenschaften		
7	11	82	weiblich	168	55	VWL	Politikwissenschaften		
8	13	84	weiblich	160	67	Politikwissenschaften	Geschichte		
9	22	85	weiblich	174	68	Politikwissenschaften	VWL		
10	5	86	weiblich	160	50	Soziologie	Politikwissenschaften		
11	10	82	weiblich	169	57	Politikwissenschaften	Medienwissenschaften		
12	24	86	weiblich	160	55	Geschichte	Politikwissenschaften		
13	9	83	weiblich	155	45	Soziologie	VWL		
14	4	85	weiblich	165	65	Politikwissenschaften	Soziologie		

Note: The figure shows a data frame in spreadsheet form.

read in a set of characteristics of students at the University of Zurich, calling the object `dat1`. Then

```
View(dat1)
```

causes the data to show in spreadsheet form (see Figure 2.1).

There are other ways to view the contents of the data frame. For example, if we want to display the data for the first 5 observations, we can use the `head` command.

```
head(dat1, n = 5)
```

```
##   id byear      man height weight                       hf
## 1  7    81 weiblich    177     76    Politikwissenschaften
## 2 19    79 weiblich    162     50 Erziehungswissenschaften
## 3 16    86 weiblich    162     55    Politikwissenschaften
## 4 12    83 weiblich    167     61    Politikwissenschaften
## 5 12    88 weiblich    170     52    Politikwissenschaften
##                  nf_1                 nf_2       nf_3 cigs
## 1 Sozialanthropologie                                   60
## 2          Soziologie                                    0
## 3                BWL                                     0
## 4         Vlkerrecht                                    70
## 5          Soziologie Medienwissenschaften Geschichte    0
##             taken estheight estweight br gruppe smoke
## 1       Beziehung       170        65  7      1     1
## 2       Beziehung       170        70  2      1     0
## 3 keine Beziehung       170        65  7      4     0
## 4       Beziehung       171        75  7      1     1
## 5 keine Beziehung       175        68  7      5     0
```

(If we want to see more lines, then we can change `n = 5` into some other number.) If we want to display the last 5 observations, then we issue

```
tail(dat1, n = 5)
```

```
##    id byear       man height weight                    hf
## 82  6    85 maennlich    178     85 Politikwissenschaften
## 83  1    86 maennlich    185     78                   VWL
## 84 17    86 maennlich    170     69 Politikwissenschaften
## 85 17    84 maennlich    178     70                   VWL
## 86  5    78 maennlich    173     75            Geschichte
##                     nf_1 nf_2 nf_3 cigs           taken
## 82                   Jus              0       Beziehung
## 83 Politikwissenschaften              0 keine Beziehung
## 84                   Jus             50 keine Beziehung
## 85 Politikwissenschaften              1       Beziehung
## 86            Soziologie              0 keine Beziehung
##    estheight estweight br gruppe smoke
## 82       181        77  7      5     0
## 83       180        75  7      5     0
## 84       165        75  7      5     1
## 85       168        63  7      1     1
## 86       172        65  7      5     0
```

If you want to have more control over the contents from the data frame that are being displayed, then the following command comes in handy:

```
dat1[1:5,1:3]
```

```
##   id byear      man
## 1  7    81 weiblich
## 2 19    79 weiblich
## 3 16    86 weiblich
## 4 12    83 weiblich
## 5 12    88 weiblich
```

This command shows the first 5 rows and first 3 columns of the data frame. Had we wanted to display the first 10 rows and first 5 columns, then we could have issued `dat1[1:10,1:5]`.

There is one more, extremely useful command, to wit `summary`. This command can be used in many different contexts. However, in the case of data frames, the command allows us to obtain descriptive statistics of the variables.

```
summary(dat1)
```

```
##        id            byear              man
##  Min.   : 1.00   Min.   :72.00   weiblich :38
##  1st Qu.: 6.00   1st Qu.:83.00   maennlich:48
```

```
##  Median :11.00   Median :85.00
##  Mean   :11.56   Mean   :83.81
##  3rd Qu.:16.75   3rd Qu.:86.00
##  Max.   :26.00   Max.   :88.00
##
##      height          weight            hf
##  Min.   :153.0   Min.   : 43.00   Length:86
##  1st Qu.:168.0   1st Qu.: 60.00   Class :character
##  Median :173.5   Median : 66.00   Mode  :character
##  Mean   :173.4   Mean   : 66.55
##  3rd Qu.:179.8   3rd Qu.: 73.00
##  Max.   :192.0   Max.   :115.00
##                  NA's   :1
##      nf_1               nf_2              nf_3
##  Length:86          Length:86          Length:86
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##       cigs                    taken      estheight
##  Min.   :  0.00   keine Beziehung:47   Min.   :163.0
##  1st Qu.:  0.00   Beziehung      :39   1st Qu.:170.0
##  Median :  0.00                       Median :173.5
##  Mean   : 28.49                       Mean   :174.0
##  3rd Qu.: 57.50                       3rd Qu.:177.8
##  Max.   :180.00                       Max.   :185.0
##
##    estweight           br            gruppe
##  Min.   :55.00   Min.   :0.000   Min.   :1.000
##  1st Qu.:67.00   1st Qu.:7.000   1st Qu.:2.000
##  Median :70.00   Median :7.000   Median :3.500
##  Mean   :69.69   Mean   :6.541   Mean   :3.302
##  3rd Qu.:73.00   3rd Qu.:7.000   3rd Qu.:5.000
##  Max.   :85.00   Max.   :7.000   Max.   :5.000
##  NA's   :1       NA's   :1
##      smoke
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4651
##  3rd Qu.:1.0000
##  Max.   :1.0000
##
```

(Do not worry if these numbers do not make sense yet; we'll get to that later in the work book.)

2.4 Variable and Value Types

2.4.1 Types of Variables

In statistics, a common distinction is made between nominal, ordinal, interval, and ratio variables. In R, the first two types are known as **factors**, whereas the latter two types are called **numeric**.[2] It is easy tell what kind of variable you are dealing with. When you issue the `summary` command, you will see a listing of the discrete values of a factor, whereas you will see summary statistics for numeric values. You can verify this by looking at the example on the previous page. Another way to check is to issue the following syntax:

```
is.factor(dat1$hf)

## [1] FALSE

is.numeric(dat1$height)

## [1] TRUE
```

If the logical operator `TRUE` is obtained, then it means that the variable is of the particular type. If the logical operator `FALSE` shows, then we know that the variable is not of the type. For example,

```
is.numeric(dat1$nf_1)

## [1] FALSE
```

shows that the variable `subcontinent` is not a numerical variable. What type a variable determines how it is handled by R, for example what statistics can be computed.

2.4.2 Types of Values

Frequently, the cells in the spreadsheet contain the value `NA`. This stands for "not available." When collecting data, we may not always be able to obtain (valid) information for all of the units. When (valid) information is missing, we need to indicate this to the statistical software we are using. In R, this is done by entering `NA`. When we see this symbol, then we know that there is a **missing value**. Missing values cause some real issues in statistics but they are a fact of life. In the practice of using statistics, you will encounter `NA` more often than

[2] A third type are string variables, which are variables made up entirely of alphabetical characters. Examples are names of people and locations.

Table 2.1: Economic and Political Indicators for Southern Africa

Country	Per Capita GDP	Polity IV Score
Botswana	7411.3	8
Lesotho	1035.3	8
Namibia	5614.6	6
South Africa	6886.3	8
Swaziland	2825.3	-9

Note: Data from the World Bank and the Center for Systemic Peace. Per capita GDP is measured in US Dollars for 2013. The Polity IV score is based on data from 2010.

you may care for. When you see it, then you should be prepared to adjust some of the statistical commands in R so that they can handle the missing values. But we are getting ahead of ourselves.

Note that the `summary` command shows the number of `NA`s. This information is useful both to realize whether there are missing values and the extent of them. In general, it is recommended that you always run the `summary` command before engaging in any type of statistical analysis.[3]

2.5 Generating a Data Frame

2.5.1 Generating a Data Frame from Vectors

Imagine, we have obtained the data shown in Table 2.1. We would like to turn this data into a data frame. One way to do this, is to create vectors of country names, per capita GDP, and Polity scores. These can then be merged into a data frame by binding the columns together. Here goes.

```
country <- c("Botswana", "Lesotho", "Namibia", "South Africa",
  "Swaziland")
gdp <- c(7411.3, 1035.3, 5614.6, 6886.3, 2825.3)
polity <- c(8, 8, 6, 8, -9)
africa <- as.data.frame(cbind(country, gdp, polity))
head(africa, n = 5)
```

```
##        country    gdp polity
## 1     Botswana 7411.3      8
## 2      Lesotho 1035.3      8
## 3      Namibia 5614.6      6
## 4 South Africa 6886.3      8
## 5    Swaziland 2825.3     -9
```

[3]Some other value types that you may encounter in R are `Inf` for infinity and `NaN`, which stands for "not a number."

We have now created the data frame `africa`. The `cbind` command binds the three vectors together into a matrix with 3 columns. The `as.data.frame` command then renders this object into a data frame.

2.5.2 Reading Data from an R Data Object

If the data have already been saved as an R data object, then reading them is extremely simple. One can recognize R data objects through the extensions of `.Rda` or `.Rds`. In this case, all you need to do is to load the data set. For example, for the student data

```
load("studidata.Rds")
```

will do the job. You do not have to assign a name to the object, since the `.Rds` file already contains a label for the data frame, to wit `dat1`. Note that R will look for the file in your working directory, which you can set using the `setwd` or `getwd` commands. If the file resides somewhere else, then you will need to provide a full path or type

```
load(file.choose())
```

which opens up a browser for you to select the file.

2.5.3 Reading Data from Files Generated by Other Programs

Often statistical data are delivered in formats other than `.Rda` or `.Rds`. In political science, for example, it is customary for data sets to be published as Microsoft Excel files, tab or comma delimited text files, or Stata `.dta` files. Reading these files requires the use of the `foreign` or `xlsx` libraries.

Let us start with reading Stata files produced with versions 5-12 of the software.[4] To read these files, we proceed as follows.

```
library(foreign)
dat2 <- read.dta("studidata.dta")
```

Here, `studi.dta` is the name of the Stata data file and `dat2` is the name of the data frame that is generated from this file. R reads in the data and assigns `NA` to missing values. It also converts those variables with value labels into factors.[5] Unfortunately, variable names are lost in the conversion process.

The process of reading a tab or comma delimited text file is very similar. Such files are typically saved with the extension `.csv`. They can be read as follows.

[4] For later versions, the package `haven` may be of some help.

[5] Should you want to prevent this, then you should issue `options(stringsAsFactors = FALSE)` at the beginning of the R session or add the option `convert.factors = FALSE`.

```
dat3 <- read.csv("studidata.csv", header = TRUE, sep = ",")
```

Here, `studidata.csv` is the text file, which is read into the data frame titled `dat3`. The option `header = TRUE` indicates that the first row of the `.csv` file contains the variable names. The option `sep = ","` is used when the character separating the values of different variables is a comma. If the text file instead is tab separated, then we would specify `sep = " "`.

The process of reading an Excel file is a bit different. Now we use the `xlsx` library.

```
library(xlsx)

## Loading required package:  rJava
## Loading required package:  xlsxjars

dat4 <- read.xlsx("studidata.xlsx", 1L)
```

The option `1L` indicates that we want to read from the first worksheet. If we want to read from the second work sheet, we would change this to `2L`, etc.

2.6 Saving a Data Frame

2.6.1 Saving to an R Format

If you plan on using R as your main statistical programming platform—and this is what we hope you will decide to do—then it makes a great deal of sense to safe your data frames in a native R format. It is extremely easy to do so. Imagine, for example, that you want to safe the data frame named `dat2` in a native R format, then the following lne of syntax will accomplish this task.

```
save(dat2, file = "studidata.Rda")
```

(You can also use the extension `.Rds`.) The file will now be saved to your working directory. When you load it at a later point in time, then the data object will be called `dat2`.

2.6.2 Exporting to Another Format

Of course, you can also save your data frame in another format. This might be useful, for example, if you are working with a colleague who does not know R. The following syntax accomplishes the job.

```
# Writing to .csv
write.csv(dat1, file = "test.csv", row.names = FALSE)
# Writing to Excel
write.xlsx(dat1, file = "test.xlsx", row.names = FALSE)
```

Both commands save the data frame `dat1` in a non-native R format in the working directory. The first command exports the data frame in `.csv` format, whereas the second command does it in an Excel format. Note that we have suppressed the variable names; these will not be saved on the first row. By changing `row.names = FALSE` to `row.names = TRUE`, we can change that. Also note that if you have a very large data frame, it is better to save it as a `.csv` than as an Excel file, since the former takes up less space.

2.7 Conclusion

In this chapter, we have taken a look at data frames. Data frames are the R objects you will most frequently encounter. We have shown how to generate, save, and view the contents of a data frame. In the next chapter, we shall show how you can manipulate the contents of the data frame. Once you know that, you are in a good position to start doing some real data analysis in R.

2.8 Exercises

(1) The file `world indicators.dta` is a Stata file that contains economic, political, and social indicators for countries around the world. Read this file into a data frame in R.

(2) Now save the data frame in the R native format.

(3) Display the first 3 observations of the world indicators data frame. To which countries do these observations correspond?

(4) How many missing values are there on the `polity` variable?

(5) Generate a data frame from the data shown in Table 1.1.

Chapter 3

Data Manipulation

Data manipulation is the act of transforming and otherwise manipulating data frames, for example, by creating new variables. There is a frequent need for data manipulation. For example, we may seek to create a body mass index from height and weight or a human development index from data about life expectancy, educational attainment, and income. In this chapter, we consider some strategies for engaging in data manipulation.

3.1 Required Packages

For the exercises in this chapter, you will need to install and activate the following library:

- `car`
- `dplyr`
- `tidyr`

3.2 Sorting, Selecting, and Sampling Data

Consider the data in `world indicators.dta`. This is a Stata data file with economic and political indicators of countries. The first 3 observations on the first five variables are given by:

```
library(foreign)
world <- read.dta("world indicators.dta")
world[1:3,1:5]
```

```
##       country continent    subcontinent  pcgdp   hdi
## 1 Afghanistan      Asia   Southern Asia  664.8 0.468
## 2     Albania    Europe Southern Europe 4458.1 0.716
## 3     Algeria    Africa Northern Africa 5360.7 0.717
```

The data is organized by country names (in alphabetical order). It also contains information about the (sub-)continent in which a country is located.

3.2.1 Sorting the Contents of a Data Frame

Imagine that we want to sort the data by continent and subcontinent first and then by country name. We can accomplish this using the `arrange` command in `dplyr`:

```
library(dplyr)

##
## Attaching package:  'dplyr'
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union

world2 <- arrange(world, continent, subcontinent, country)
world2[1:3,1:5]

##    country continent   subcontinent  pcgdp   hdi
## 1  Burundi    Africa Eastern Africa  267.1 0.389
## 2  Comoros    Africa Eastern Africa  841.8 0.488
## 3 Djibouti    Africa Eastern Africa 1668.3 0.467
```

The data frame `world2` now contains a sorted copy of `world`. Since, alphabetically speaking, the first continent is Africa and the first sub-continent is East Africa, this means that the first three observations pertain to Burundi, Comoros, and Djibouti. By default, the data are sorted in ascending order. However, if you want the sort to be done in a descending order, this requires only a small modification of the syntax:

```
world2 <- arrange(world, continent, subcontinent, desc(country))
world2[1:3,1:5]

##    country continent   subcontinent  pcgdp   hdi
## 1 Zimbabwe    Africa Eastern Africa  953.4 0.492
## 2   Zambia    Africa Eastern Africa 1844.8 0.561
## 3   Uganda    Africa Eastern Africa  657.4 0.484
```

Now continent and sub-continent are still ordered alphabetically, but the countries within a sub-continent are listed in the reversed alphabetical order. If we want to to reverse alphabetical ordering on the sub-continents, as well, then the following accomplishes the task:

```
world2 <- arrange(world, continent, desc(subcontinent),
  desc(country))
world2[1:3,1:5]

##        country continent   subcontinent  pcgdp   hdi
## 1         Togo    Africa Western Africa  636.4 0.473
## 2 Sierra Leone    Africa Western Africa  809.0 0.374
## 3      Senegal    Africa Western Africa 1046.6 0.485
```

You see that whichever object is the argument of `desc` will be ranked in descending order.

3.2.2 Selecting Rows from a Data Frame

In many cases, we do not just want to sort on a variable but actually select on that variable. Selection means that we create a copy of the data frame that consists solely of cases that meet some criterion. With the world indicators, for example, we may wish to retain only the countries of Eastern Asia. This can be done again using `dplyr`:

```
library(dplyr)
world2 <- filter(world, continent == "Asia",
  subcontinent == "Eastern Asia")
world2[1:8,1:5]

##       country continent subcontinent   pcgdp   hdi
## 1       China      Asia Eastern Asia  6991.9 0.719
## 2   Hong Kong      Asia Eastern Asia 38364.2 0.891
## 3       Japan      Asia Eastern Asia 38633.7 0.890
## 4       Macao      Asia Eastern Asia 90600.4    NA
## 5    Mongolia      Asia Eastern Asia  4418.8 0.698
## 6 North Korea      Asia Eastern Asia      NA    NA
## 7 South Korea      Asia Eastern Asia 25997.9 0.891
## 8      Taiwan      Asia Eastern Asia      NA    NA
```

Here `==` is the logical operator "equal to." You see that `world2` now only contains data from the Eastern Asian sub-continent. This is useful, if we want to restrict a set of analyses to this sub-continent only.

3.2.3 Selecting Columns from a Data Frame

Just like we can select a subset of cases, we can also select a subset of variables. That is, we create a subset of variables for all of the sampling units. This can be done using `dplyr`'s `select` command.

```
world2 <- dplyr::select(world, country, polstab, goveff,
  corrupt, law,regul, voice)
head(world2, n = 3)

##       country polstab goveff corrupt law regul voice
## 1 Afghanistan       1      7       2   1     9    13
## 2     Albania      48     44      26  36    57    51
## 3     Algeria      13     32      39  29    11    23
```

The data frame `world2` now retains only the country name and the six governance indicators collected by the World Bank.
We call `select` explicitly (`packagename::function`) because `select` is used by many other packages. If the package `MASS` is loaded for example, R would take the `select` from `MASS` instead of `dyplr` and we would get an error. If a package is called explicitly, it does not have to be loaded.

3.2.4 Sampling from a Data Frame

Finally, we can draw samples from data frames. Generally speaking, there is no need to do this. However, with very large data sets, it may be useful to test functions and statistical models on a sub-sample, as this will take less computing time. In R, we can sample a fraction as well as a specified number of cases. To sample a specific number of cases, the following syntax works well.

```
library(dplyr)
world2 <- sample_n(world, size = 10)
world2[,1:3]

##                               country continent
## 91                               Iran      Asia
## 68                             France    Europe
## 30                           Bulgaria    Europe
## 42                           Colombia  Americas
## 197                          Thailand      Asia
## 167 Saint Vincent and the Grenadines  Americas
## 93                            Ireland    Europe
## 103                          Kiribati   Oceania
## 35                           Cameroon    Africa
## 60                  Equatorial Guinea    Africa
##           subcontinent
## 91       Southern Asia
## 68      Western Europe
## 30      Eastern Europe
## 42       South America
## 197 South-Eastern Asia
## 167          Caribbean
```

```
## 93     Northern Europe
## 103         Micronesia
## 35       Middle Africa
## 60       Middle Africa
```

This command samples 10 cases from the data frame `world` and stores them in the data frame `world2`. Note that the original data are still available. This is the advantage of an object-oriented language that allows for multiple objects. Also note the structure of the last command. Note that we do not specify anything before the comma, which tells R that all rows should be listed. We do specify something after the comma, so that specific columns (1 through 3) will be listed.[1] To sample a fraction, we issue:

```
world2 <- sample_frac(world, size = .05)
world2[,1:3]

##            country continent              subcontinent
## 133        Morocco    Africa           Northern Africa
## 43         Comoros    Africa            Eastern Africa
## 141    New Zealand   Oceania Australia and New Zealand
## 51  Czech Republic    Europe            Eastern Europe
## 45      Costa Rica  Americas           Central America
## 104         Kosovo    Europe           Southern Europe
## 122           Mali    Africa            Western Africa
## 126     Mauritania    Africa            Western Africa
## 115     Luxembourg    Europe            Western Europe
## 174     Seychelles    Africa            Eastern Africa
## 21          Belize  Americas           Central America
```

This command draws a sample of 5 percent of the rows from the data frame `world`.

3.3 Conversions between Variable Types

As we saw in Chapter 2, variables come in two basic varieties in R: numeric and factor. Occasionally, you may wish to convert a factor to a numeric type or vice versa. R has built-in commands for accomplishing these tasks.

3.3.1 Converting from Factor to Numeric Type

To convert a factor variable into a numeric type is relatively straightforward. The data frame `world` contains an indicator distinguishing between democracies and non-democracies. This is a factor variable:

[1] Similarly, if you want to list the first 10 rows but all of the columns, you could write `world2[1:10,]`.

```
is.factor(world$demo)

## [1] TRUE
```

To create a copy of the variable that is numeric, we issue the following syntax:[2]

```
world$n.demo <- as.numeric(world$demo)
is.numeric(world$n.demo)

## [1] TRUE
```

To see how the variable is coded, let us display its summary statistics:

```
summary(world$n.demo)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    1.00    1.00    1.44    2.00    2.00      61
```

We see that the lowest value (Min.) is 1 and the highest value (Max.) is 2. Later, we shall see how one can dummy or 0/1 code the original variable.

3.3.2 Converting from Numeric to Factor Type

If we have relatively few numeric values, then the conversion to factor can proceed using the `factor` command. To illustrate the process, let us convert the newly created numeric democracy variable to a factor:

```
world$f.demo <- factor(world$n.demo,
  labels = c("Non-Democratic", "Democratic"))
summary(world$f.demo)

## Non-Democratic     Democratic           NA's
##             89             70             61
```

To ensure this is correct, let us compare the results to the original democracy variable:

```
summary(world$demo)

## Non-Democratic     Democratic           NA's
##             89             70             61
```

Since the distributions of `f.demo` and `demo` are identical, it looks like the conversion worked. To be absolutely sure, we want to ascertain that the value of

[2] We recommend making copies of extant variables that are converted or recoded, so that the original remains available.

each country is the same for the two variables. This can be achieved using the following syntax:

```
world$demo == world$f.demo
```

```
##   [1]   NA TRUE TRUE   NA   NA TRUE   NA   NA TRUE TRUE   NA
##  [12] TRUE TRUE TRUE   NA TRUE TRUE   NA TRUE TRUE   NA TRUE
##  [23]   NA TRUE TRUE   NA TRUE TRUE   NA TRUE TRUE TRUE TRUE
##  [34] TRUE TRUE TRUE   NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [45] TRUE   NA TRUE TRUE   NA TRUE TRUE TRUE TRUE TRUE   NA
##  [56] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE   NA   NA TRUE
##  [67] TRUE TRUE   NA   NA TRUE TRUE TRUE TRUE TRUE TRUE   NA
##  [78]   NA   NA TRUE TRUE TRUE TRUE   NA TRUE   NA TRUE   NA
##  [89] TRUE TRUE TRUE TRUE TRUE   NA TRUE TRUE TRUE TRUE   NA
## [100] TRUE TRUE TRUE   NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [111] TRUE TRUE   NA TRUE   NA   NA TRUE TRUE TRUE TRUE   NA
## [122] TRUE   NA   NA   NA TRUE TRUE TRUE TRUE   NA TRUE TRUE
## [133] TRUE TRUE TRUE TRUE   NA TRUE TRUE   NA TRUE TRUE TRUE
## [144] TRUE TRUE   NA TRUE TRUE TRUE   NA TRUE TRUE TRUE TRUE
## [155] TRUE TRUE TRUE   NA TRUE   NA TRUE TRUE TRUE   NA   NA
## [166]   NA   NA   NA   NA   NA TRUE TRUE TRUE   NA TRUE TRUE
## [177]   NA TRUE TRUE TRUE   NA TRUE TRUE   NA TRUE TRUE   NA
## [188] TRUE   NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [199] TRUE   NA TRUE TRUE TRUE TRUE   NA   NA TRUE TRUE TRUE
## [210] TRUE   NA TRUE TRUE TRUE   NA TRUE TRUE TRUE TRUE TRUE
```

We see that all levels of the factor variables are identical (TRUE), at least where the data are not missing (NA).

When there are many values of the numeric variable, then it will be necessary to group the values first. This can be accomplished by using R's `cut` command. We illustrate this using the human development index (HDI). The United Nations group the HDI into four groups depending: (1) a HDI greater than .80 ("very high human development"); (2) a HDI greater than .70 ("high human development"); (3) a HDI greater than .55 ("medium human development"); and (4) HDI scores between 0 and .55 ("low human development"). We can apply this coding scheme using the following syntax:

```
world$f.hdi <- cut(world$hdi,
  breaks=c(-Inf, 0.55, 0.70, 0.80, Inf),
  right = TRUE, labels = c("Low Dev.", "Medium Dev.",
    "High Human Dev.", "Very High Human Dev."))
table(world$f.hdi)
```

```
##
##             Low Dev.          Medium Dev.
##                   43                   43
```

```
##      High Human Dev. Very High Human Dev.
##                   52                   49
```

The option `right = TRUE` means that the interval is open on the left and closed on the right (e.g., $(0.70, 0.80]$). To reverse this, you should specify `right = FALSE`.

3.4 Recoding Variables

By recoding, we mean the acts of combining and reassigning values of a variable. This is quite difficult to do in R, but fortunately the `car` library offers a simple solution. We illustrate the process using the `polity` variable in the `world` data frame. The original variable ranges between -10 and 10, in steps of 1, although there are some values smaller than -10 to indicate special transitional states of countries. We want to recode the values -10 through -5 into 1, -4 through 4 into 2, and 5 through 10 into 3, while all else is coded as `NA`. The syntax for making this recode is as follows:

```
world$new.polity <- car::recode(world$polity,
  "-10:-5=1;-4:4=2;5:10=3;else=NA")
table(world$new.polity)

##
##   1   2   3
##  25  35 100
```

The `car` library can also be used to recode and convert formats at the same time. We show the process with the same example as before. [`recode` is also used from dplyr, for that we call it explicitly.]

```
world$f.polity <- car::recode(world$polity,
  "-10:-5='Low';-4:4='Medium';5:10='High';else=NA",
  as.factor.result=TRUE)
table(world$f.polity)

##
##   High    Low Medium
##    100     25     35
```

Similarly, `recode` can be used to convert factor to numeric variables:

```
world$n.demo <- car::recode(world$demo,
  "'Non-Democratic'=0;'Democratic'=1;else=NA",
  as.numeric.result=TRUE)
summary(world$n.demo)
```

```
##    0    1 NA's
##   89   70   61
```

Now we have created a **dummy** (i.e., 0/1) variable for democracy.

3.5 Creating New Variables

It is frequently necessary to create new numeric variables from existing ones. A lot of what we need to do this was discussed already under the rubric of functions in Chapter 1. In essence, the process is extremely simple, although it can get somewhat more complicated when we want to create variables that contain group means and other statistics.

3.5.1 Simple Computations

Let us start with a simple example. The data set `world indicators.dta` includes six governance indicators that have been collected by the World Bank:

polstab	political stability and absence of violence
goveff	government effectiveness
corrupt	control of corruption
law	rule of law
regul	regulatory quality
voice	voice and accountability

These indicators are coded such that higher scores indicate better governance. We would now like to create an index that combines all six indicators. We do this by summing those indicators and then dividing by 6.

```
attach(world)

## The following objects are masked _by_ .GlobalEnv:
##
##    country, polity

governance <- (1/6)*(polstab+goveff+corrupt+law+regul+voice)
summary(governance)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.1667 27.6700 48.5000 49.7600 73.6700 98.3300      11

detach(world)
```

We have now created a new variable `governance` in the data frame that contains the desired index. Note that we used the `attach` command so that we did not have to reference the data frame using the $ notation.

We can also use `dplyr`'s `mutate` command to add new variables. We illustrate this by creating a new variable containing the ranks on the human development index.

```
library(dplyr)
world <- mutate(world, hdi.rank = 187-rank(hdi, na.last = "keep",
  ties.method = "average"))
summary(world$hdi.rank)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0    46.5    92.5    93.0   139.5   186.0      33
```

```
world[1:10,c(1, 24)]
```

```
##                 country hdi.rank
## 1           Afghanistan    168.0
## 2               Albania     94.0
## 3               Algeria     92.5
## 4        American Samoa       NA
## 5               Andorra     36.5
## 6                Angola    148.0
## 7              Anguilla       NA
## 8  Antigua and Barbuda     60.0
## 9             Argentina     48.0
## 10              Armenia     86.0
```

We assign the new variable to the same data frame, i.e., `world`. The `rank` command ranks the HDI values from smallest to largest. We have added two options. First `na.last = "keep"` means that missing values appear as `NA` on the ranked variable. Second, `ties.method = "average"` is a method for addressing ties. A tie occurs when two countries have identical HDI scores. The average method means that the ranks of those two countries will be averaged. The ranking procedure gives the highest rank to the country with the highest HDI (Norway). However, we typically want this country to have a rank of 1, indicating that it is first in HDI. To accomplish this, we subtract the rank from 187, i.e., the number of countries with non-missing values on the HDI. We can construct this number as follows:

```
sum(!is.na(world$hdi))
```

```
## [1] 187
```

We summarize the results using the `summary` command. We also list the 1st and 24th columns of the data frame for the first 10 rows, so that we can check the plausibility of the results.[3]

[3] By using the `concatinate` function, we can select non-sequential columns from the data frame.

Finally, let us consider the use of `ifelse` statements. Imagine that we want to create two dummy variables, one for democracies (the `polity` score is at least 8) and one for autocracies (the `polity` score is at most -8). We can accomplish this using the `cut` command that we described earlier. However, we can also use the `ifelse` syntax:

```
world$democracy <- ifelse(world$polity>=8, 1, 0)
world$autocracy <- ifelse(world$polity<=-8 & world$polity>=-10, 1, 0)
world[1:20,c(1, 12, 25, 26)]
```

```
##                country polity democracy autocracy
## 1          Afghanistan    -66         0         0
## 2              Albania      9         1         0
## 3              Algeria      2         0         0
## 4       American Samoa     NA        NA        NA
## 5              Andorra     NA        NA        NA
## 6               Angola     -2         0         0
## 7             Anguilla     NA        NA        NA
## 8  Antigua and Barbuda     NA        NA        NA
## 9            Argentina      8         1         0
## 10             Armenia      5         0         0
## 11               Aruba     NA        NA        NA
## 12           Australia     10         1         0
## 13             Austria     10         1         0
## 14          Azerbaijan     -7         0         0
## 15             Bahamas     NA        NA        NA
## 16             Bahrain     -8         0         1
## 17          Bangladesh      5         0         0
## 18            Barbados     NA        NA        NA
## 19             Belarus     -7         0         0
## 20             Belgium      8         1         0
```

The first `ifelse` command assigns the value 1 to the new variable `democracy` if `polity` is greater or equal to 8; everywhere else, a value of 0 is assigned. The second `ifelse` command assigns the value 1 to the new variable `autocracy` if `polity` is no smaller than -10 and no larger than -8; everywhere else a value of 0 is assigned. This means that countries with `polity` scores below -10 and between -7 and 7 receive scores of 0 on both dummy variables. Those countries with a score of `NA` on `polity` automatically receive a score of `NA` on both dummy variables.

For the creation of new numeric variables, R contains numerous built-in

functions. Commonly used functions include the following:

`abs(x)`	absolute value of x	`round(x, k)`	round x to the kth digit
`ceiling(x)`	round up x	`sqrt(x)`	$\sqrt{x}$
`exp(x)`	e^x	`a+b`	$a + b$
`floor(x)`	round down x	`a*b`	$a \cdot b$
`log(x)`	$\ln(x)$	`a/b`	$\frac{a}{b}$
`log10(x)`	$\log(x)$	`x^a`	x^a

3.5.2 Computations Involving Grouping Variables

To conclude the discussion of computing new variables, let us consider group summaries as variables to a data frame. More specifically, we want to add a variable measuring the lowest level of the human development index on a sub-continent. This can be done using `dplyr`:

```
world2 <- group_by(world, subcontinent)
world2 <- mutate(world2, min.hdi = min(hdi, na.rm = TRUE))
world2[1:10,c(1, 3, 5, 27)]

## # A tibble: 10 x 4
## # Groups:   subcontinent [8]
##                 country    subcontinent   hdi min.hdi
##                   <chr>          <fctr> <dbl>   <dbl>
##  1         Afghanistan   Southern Asia 0.468   0.468
##  2             Albania Southern Europe 0.716   0.716
##  3             Algeria Northern Africa 0.717   0.473
##  4      American Samoa       Polynesia    NA   0.694
##  5             Andorra Southern Europe 0.830   0.716
##  6              Angola   Middle Africa 0.526   0.338
##  7            Anguilla       Caribbean    NA   0.471
##  8 Antigua and Barbuda       Caribbean 0.774   0.471
##  9           Argentina   South America 0.808   0.638
## 10             Armenia    Western Asia 0.730   0.500
```

The first command generates a data frame `world2` that is sorted by sub-continent. The second command then creates a new variable, `min.hdi`, which is the minimum HDI value in the sub-continent. The `mutate` command knows that the minimum should be created by sub-continent because this is how we grouped the data frame.

3.6 Collapsing and Reshaping Data

We finish the discussion of data management by discussing how to collapse and reshape data. The need to carry out these transformations does not arise frequently. It does become relevant, however, when you engage in advanced statistical analyses such as panel data analysis.

3.6.1 Collapsing a Data Frame

When we collapse a data frame, this means that we create a new data frame at a higher level of aggregation. Consider, for example, the following hypothetical data:

Person	*Class*	*Height*	*Weight*
1	1	1.07	22.7
2	1	1.12	25.3
3	1	1.00	21.1
4	2	1.22	30.0
5	2	1.30	28.2

We now want to collapse the data to the class level, for example by computing the highest age and weight in each class and presenting those class results. Hence, the collapsed data looks like this:

Class	*Min.Height*	*Min.Weight*
1	1.00	21.1
2	1.22	28.2

Such a presentation of the data is useful when you are interested in class-level instead of individual data.

In R, collapsing data is quite easy when you use `dplyr`'s `summarize` command. I illustrate this with the world economic, political, and social indicators data. Specifically, we create a data frame consisting of the sub-continents and the maximum values of the human development index and political stability in each sub-continent.

```
world3 <- group_by(world, subcontinent)
world3 <- summarize(world3, max.hdi=max(hdi, na.rm=TRUE),
  max.stab=max(polstab, na.rm=TRUE), count=n())
head(world3)

## # A tibble: 6 x 4
##                subcontinent max.hdi max.stab count
##                      <fctr>   <dbl>    <dbl> <int>
## 1 Australia and New Zealand   0.933       99     2
## 2                 Caribbean   0.815      100    23
## 3           Central America   0.765       67     8
## 4              Central Asia   0.757       54     5
## 5            Eastern Africa   0.771       78    19
## 6              Eastern Asia   0.891       82     8
```

We have already seen the first command, which causes the data in `world` to be grouped by sub-continent. The second command is new: it creates summaries, in this case maximums as well as the number of cases (`count`), by sub-continent. These are then stored in the data frame `world3`.

3.6.2 Changing between Long and Wide Formats

In political science, we frequently collect data over multiple years for different countries. We can structure these data in multiple ways. A first structure is that in **wide** format. Here, the years become columns. As an example, consider the Polity scores in 1970, 1990, and 2010 for South American countries:

```
southam.wide <- read.dta("south america wide.dta")
head(southam.wide)

##     country polity1970 polity1990 polity2010
## 1 Argentina         -9          7          8
## 2   Bolivia         -5          9          7
## 3    Brazil         -9          8          8
## 4     Chile          6          8         10
## 5  Colombia          7          8          7
## 6   Ecuador          0          9          5
```

We observe that each country receives one row. The Polity scores for the different years are shown in separate columns. A second structure is called the **long** format. Here, the years become sub-units for each country. For example, if we have three years, then each country occurs three times. As an example, consider again the Polity scores for South America:

```
southam.long <- read.dta("south america long.dta")
head(southam.long)

##     country year polity
## 1 Argentina 1970     -9
## 2 Argentina 1990      7
## 3 Argentina 2010      8
## 4   Bolivia 1970     -5
## 5   Bolivia 1990      9
## 6   Bolivia 2010      7
```

Now we observe that instead of having three Polity scores, we are left with only one variable. We also observe that each country occurs three times. Each time that the country occurs is marked by a year: 1970, 1990, or 2010. Whereas the wide structure requires only 12 rows (there are 12 countries), the long structure requires 36 rows (12 countries $\times$ 3 time points). The wide structure requires 4 columns, but the long structure requires only 3.

Some statistical programs would like the data to be structured long, whereas others prefer a wide structure. It is thus useful to know how one can shift between these formats. We prefer to do this using the package `tidyr`, as it greatly simplifies the task. To convert from wide to long format, we use `tidyr`'s `gather` function.

```
library(tidyr)
southam.wide$country <- factor(southam.wide$country)
southam1 <- gather(southam.wide, year, polity,
                   polity1970:polity2010)
head(southam1, n=15)

##       country       year polity
## 1   Argentina polity1970     -9
## 2     Bolivia polity1970     -5
## 3      Brazil polity1970     -9
## 4       Chile polity1970      6
## 5    Colombia polity1970      7
## 6     Ecuador polity1970      0
## 7      Guyana polity1970      1
## 8    Paraguay polity1970     -8
## 9        Peru polity1970     -7
## 10   Suriname polity1970     NA
## 11    Uruguay polity1970      8
## 12  Venezuela polity1970      9
## 13 Argentina polity1990      7
## 14    Bolivia polity1990      9
## 15     Brazil polity1990      8
```

It is important to turn the country variable into a factor variable. After doing that, we can apply the `gather` function. Its first argument is the name of the data frame in wide format. Its second argument is a new index for counting the measurement occasions, which we have labelled `year`. The third argument is the name of the new variable that should contain the Polity scores. The last argument gives the variables that should be rearranged from wide to long format.

The conversion from long to wide format proceeds via `tidyr`'s `spread` function. The first act is again to turn the country variable into a factor. Subsequently, the `spread` function can be applied.

```
library(tidyr)
southam.long$country <- factor(southam.long$country)
southam2 <- spread(southam.long, year, polity)
head(southam2, n=10)

##       country 1970 1990 2010
## 1   Argentina   -9    7    8
## 2     Bolivia   -5    9    7
## 3      Brazil   -9    8    8
## 4       Chile    6    8   10
## 5    Colombia    7    8    7
## 6     Ecuador    0    9    5
```

```
## 7     Guyana    1   -7    6
## 8   Paraguay   -8    2    8
## 9       Peru   -7    8    9
## 10  Suriname   NA    2    5
```

We see that `spread` takes two arguments: the year and the name of the variable that is to be spread. The function produces a new data structure with columns named `1970`, `1990`, and `2010`, which include the relevant Polity scores for each country in each year.

3.7 Conclusion

Data manipulation is one of the more difficult things to accomplish in R. Thanks to the contributions of an active user community, however, the task is becoming easier all of the time. We hope that the present chapter has shown that recoding existing variables and creating new ones is really not all that difficult in R. We also hope that you have discovered that R contains many useful functions for a variety of data manipulation tasks, functions that, to our mind, are sometimes more flexible than those found in other statistical software packages.

3.8 Exercises

(1) Open the data file `studidata.Rda` and sort the data by major field of study (`hf`), then the first minor (`nf_1`), and then the second minor (`nf_2`). Show the first ten records of the sorted data frame.

(2) Using the same data, create a data frame consisting only of those students who major in political science. Show the last ten records of the resulting data frame.

(3) Now draw a sample of 5 cases from the original data frame. Show the first three variables for these cases.

(4) The variable `smoke` is a dummy variable that takes on the value 0 if someone does not smoke and 1 if he/she smokes. Create a new factor that captures the same information but references the first group as "non-smoker" and the second group as "smoker."

(5) Use the `cut` function to create a new variable that distinguishes between four categories of smokers: (1) students who do not smoke; (2) students who smoke 15 or fewer cigarettes a week; (3) students who smoke more than 15 cigarettes a week but fewer than 100; and (4) students who smoke at least 100 cigarettes each week. Display the new variable as well as `cigs` for the first 15 students in the data frame.

(6) The data frame contains data about the estimated height (`estheight`) in centimeters and the estimated weight (`estweight`) in kilograms for each student. Based on these variables, construct a new variable measuring the BMI of each student. Show this variable for the first 5 students in the data frame.

(7) Group the data frame by sex (`man`). Now add three new variables to the data frame: the minimum BMI for each sex, the maximum BMI for each sex, and the maximum number of cigarettes consumed by each sex.

(8) Now create a new data frame of summary statistics by major. This data frame should show the minimum BMI and the maximum number of cigarettes consumed in each major. Take a look at the resulting data frame. Do you observe any pattern in BMI and cigarette consumption by major?

Part II

Descriptive Statistics

Chapter 4

Tabulating and Visualizing Frequencies

Now that we have seen the basics of R let us delve into some real data analysis. In this chapter, we discuss how one can tabulate and visualize single variables. Political attributes such as whether a country is a democracy are variables because not all units score identical on those attributes (e.g., not all countries are democracies but some are). From a descriptive perspective, political scientists want to know how often particular values arise in the data, for example, how many democracies there are. Frequency tables, bar charts, histograms, and density plots are perfect devices for conveying information about (relative) frequencies. R is capable of creating beautiful tables and graphics and this chapter shows how.

4.1 Required Packages

For the exercises in this chapter, you will need to install and activate the following library:

- `ggplot2`

4.2 Frequency Tables

4.2.1 Tabulating Factors

Consider again the data from the students of the University of Zurich. We want to know how the students are distributed across different majors. The easiest way to do this is to generate a frequency table. R contains several commands for creating frequency tables. As part of the `base` installation, one can use the `table` command.

```
load("studidata.Rda")
table(dat2$hf)
```

```
##
##                       BWL                  Englisch
##                         1                         1
## Erziehungswissenschaften               Ethnologie
##                         1                         1
##                Geschichte    Politikwissenschaften
##                        15                        36
##               Psychologie      Sozialanthropologie
##                         4                         3
##                Soziologie                       VWL
##                        15                         8
##           Zeitgeschichte
##                         1
```

As always, you can also assign the table to an object. This is useful if the goal is to transform the frequencies into relative frequencies:

```
mytable <- table(dat2$hf)
prop.table(mytable)
```

```
##
##                       BWL                  Englisch
##                0.01162791                0.01162791
## Erziehungswissenschaften               Ethnologie
##                0.01162791                0.01162791
##                Geschichte    Politikwissenschaften
##                0.17441860                0.41860465
##               Psychologie      Sozialanthropologie
##                0.04651163                0.03488372
##                Soziologie                       VWL
##                0.17441860                0.09302326
##           Zeitgeschichte
##                0.01162791
```

Personally, we would prefer to see percentages and would like them to be rounded to the first decimal value. This requires just a little bit of adjusting the `prop.table` command:

```
round(100*prop.table(mytable), 1)
```

```
##
##                       BWL                  Englisch
##                       1.2                       1.2
```

```
## Erziehungswissenschaften               Ethnologie
##                      1.2                      1.2
##                Geschichte   Politikwissenschaften
##                     17.4                     41.9
##              Psychologie     Sozialanthropologie
##                      4.7                      3.5
##               Soziologie                      VWL
##                     17.4                      9.3
##           Zeitgeschichte
##                      1.2
```

When you report percentages like this, you should always report the sample size. This can be obtained using a command that we already encountered in Chapter 3:

```
sum(!is.na(dat2$hf))

## [1] 86
```

By default, the frequency table generated by `table` is sorted in the same order as the values of the factor. If we would like to sort the table by the magnitude of the frequencies, then the following syntax will prove useful.[1]

```
mytable <- sort(table(dat2$hf), decreasing = TRUE)
mytable

##
##    Politikwissenschaften               Geschichte
##                       36                       15
##               Soziologie                      VWL
##                       15                        8
##              Psychologie     Sozialanthropologie
##                        4                        3
##                      BWL                  Englisch
##                        1                        1
## Erziehungswissenschaften               Ethnologie
##                        1                        1
##           Zeitgeschichte
##                        1
```

4.2.2 Tabulating Numeric Variables

In the examples so far, we have tabulated factors. We can tabulate numeric variables just as easily, but here it sometimes makes more sense to perform

[1] The `sort` function is an alternative for `dplyr`'s `arrange` command.

some form of grouping prior to the tabulation. Take, for example, the estimated height from the UZH student data. If we tabulate this without some form of grouping, we find that the result is not very informative.

```
table(dat2$estheight)

##
##              163              165              166
##                1                2                1
##              167              168              169
##                1                2                2
##           169.75              170              171
##                1               17                4
##              172              173              174
##                8                4                1
##              175              176              177
##               12                7                1
##              178              179              180
##                5                3                5
##              181              182 182.529998779297
##                4                2                1
##              183              185
##                1                1
```

This is not very informative because there are a lot of height values that occur only once. In this case, we may wish to convert the numeric variable to a factor using, for example, Sturges' method for computing the number of bins and bin widths. It is not difficult to do this, as the following syntax shows.

```
attach(dat2)
factor.height<-factor(cut(estheight,
                          breaks=nclass.Sturges(estheight)))
table(factor.height)

## factor.height
## (163,166] (166,168] (168,171] (171,174] (174,177] (177,180]
##         3         4        24        13        19         9
## (180,182] (182,185]
##        11         3

detach(dat2)
```

The `nclass.Sturges` option in the `cut` function takes care of the creation of the bins according to Sturges' formula, which makes life a lot easier for us. The results now clearly show that the most common height bracket is $(168, 171]$, i.e., heights greater than 1 meter 68 and up to 1 meter 71.

The categories of the grouped height variable constitute a rank ordering. For this kind of variable, it makes sense to compute both proportions and cumulative frequencies. A particularly elegant syntax for doing so is the following:

```
height.dat <- as.data.frame(table(factor.height))
height.dat <- transform(height.dat, cumFreq = cumsum(Freq),
                        relative = prop.table(Freq))
height.dat

##   factor.height Freq cumFreq   relative
## 1     (163,166]    3       3 0.03488372
## 2     (166,168]    4       7 0.04651163
## 3     (168,171]   24      31 0.27906977
## 4     (171,174]   13      44 0.15116279
## 5     (174,177]   19      63 0.22093023
## 6     (177,180]    9      72 0.10465116
## 7     (180,182]   11      83 0.12790698
## 8     (182,185]    3      86 0.03488372
```

This table now shows us that 44 of the students have heights of 1.74 meters or below.

4.3 Data Visualization

Graphics are one of R's great strengths. While the basic plots already look quite nice, you can truly unleash R's graphical powers when you use the `ggplot2` package. As you will see shortly, this is truly capable of producing some amazing looking graphics. Moreover, `ggplot2` has a very nice and simple architecture that is easily mastered.

4.3.1 Terminology

The name `ggplot` stands for grammar of graphics plot, which means that there is an actual grammar underlying the construction of graphics. This grammar has several elements, some of which are optional but most of which are mandatory.

Data Graphs in `ggplot2` should reference the data we want to visualize. These data have to be in the form of a R data frame.

Coordinate System We have to specify the coordinates of a 2-dimensional space in which the data will be mapped. For our purposes, these will usually be Cartesian coordinates.

Geoms Geoms are the geometric objects that are used to represent the data in the space. These can be points, lines, bars, and many other objects.

Aesthetics As the name implies, the aesthetics influence the aesthetic properties of the graph. This includes colors, line sizes, point sizes, etc.

Scales Scales determine how visual characteristics capture values in the data. For example, we can decide that values should be displayed in their original metric, but we can also opt for a log-scale.

Stats Stats reference statistics, more precisely, statistical transformations of the data that we would like to visualize. For example, we could opt for displaying means, medians, regression lines, etc. (Again, these concepts will be discussed in great detail later in this work book.)

Themes In `ggplot2`, a theme is an optional element that gives further control over the axes of the plot. This is sometimes useful, for example, to control the over-plotting of axis labels.

Facets In `ggplot2`, facets are used to visualize sub-groups in the data frame. This is often of great help when we seek to explore data, so that we encourage the use of facets in data visualization.

Typically, the elements of the `ggplot2` grammar are introduced in layers. For example, we first specify the data and the coordinates, only to add information about the geoms in a next step. These layers are connected through a plus symbol. The process will be illustrated here for bar charts, histograms, and density plots.

4.3.2 Creating Bar Charts

Bar charts are a perfect tool for visualizing nominal and ordinal variables. Creating them is quite easy using `ggplot2`. Here, we illustrate the grammar for this chart by visualizing the distribution of majors in our sample of UZH students. We start by specifying the data frame and the horizontal axis of the bar chart; the vertical axis, by default, is a frequency count.

```
load("studidata.Rda")
library(ggplot2)
p <- ggplot(dat2, aes(x = hf))
```

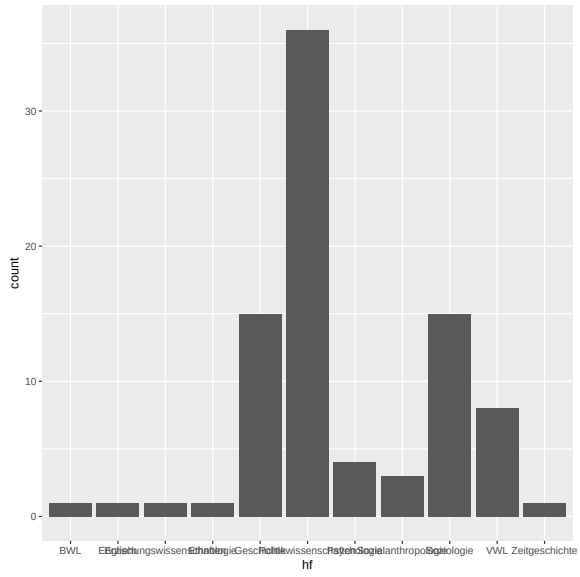
The object `p`, which is a list, now contains the basic space in which a geom can be placed. To place the geom, we issue the following syntax:

```
p1 <- p + geom_bar()
```

This is all we need to do to create a basic bar chart (see Figure 4.1).

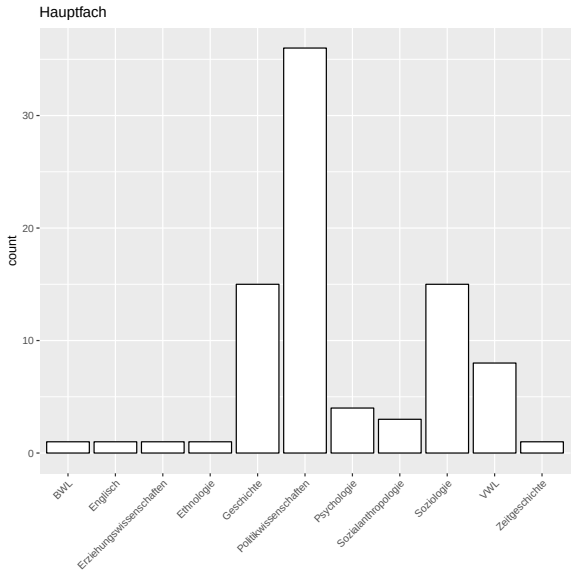
Figure 4.1 does not look very nice. There are two critical problems and one aesthetic problem. Critical is that the axis title (`hf`) is not particularly

Figure 4.1: A Basic Bar Chart



Note: This bar chart has not yet been overhauled to make it nicer. As a result, the bars are in basic black, whereas the axis labels overlap because they are too long.

Figure 4.2: An Edited Bar Chart



Note: The bar chart now clearly shows that it pertains to students' major fields of study. Those fields can be clearly identified on the horizontal axis.

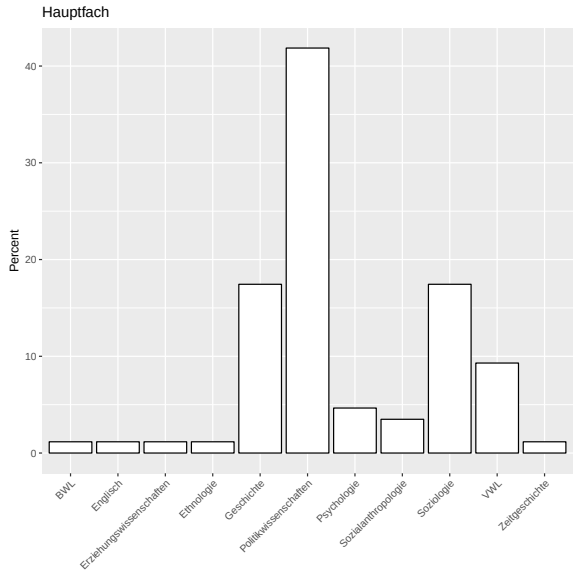
meaningful. Also critical is that the axis labels (the names of the major fields of study) overlap so that they are unreadable. An aesthetic problem, to our minds, is with the black colors of the bars. They sort of jump of the page and, should you want to print the graph, they will consume a lot of black ink. So let us fix these problems.

```
p2 <- p + geom_bar(col = "black", fill = "white") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
    axis.title.x = element_blank()) +
  ggtitle("Hauptfach")
```

The first line creates the bar chart with each bar demarcated by black lines and filled in with a white color. The second line places the axis labels at a 45 degree angle, preventing them from overlapping. The third line suppresses the title for the horizontal axis. Instead, the fourth line clarifies the meaning of the horizontal axis by adding the title "Hauptfach" to the graph. This is just a matter of taste: we find the title more attractive than the horizontal axis, which would appear below the labels and as such would be quite far down. The resulting graph is shown in Figure 4.2.

The bar chart that we just produced shows frequency counts. We might

Figure 4.3: A Bar Chart with Percentages



Note: The vertical axis of the bar chart is now in a percentage metric ($n = 86$).

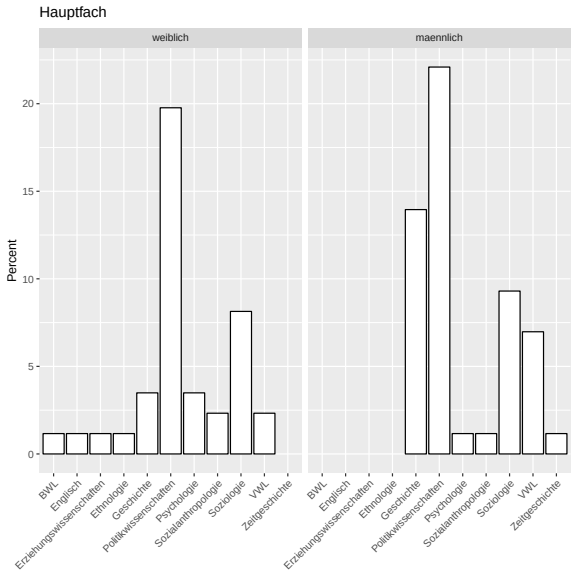
want to show percentages instead. This requires a bit of revision of the code.

```
p3 <- p + geom_bar(aes(y = 100*((..count..)/sum(..count..))),
             col = "black", fill = "white") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.title.x = element_blank()) +
  ylab("Percent") +
  ggtitle("Hauptfach")
```

The action is in the first line, where we take the frequency count for each category and the divide it by the sum of all of the frequency counts (i.e., n) to form a proportion. The multiplication by 100 converts this to a percentage. The result is shown in Figure 4.3. This graph clearly reveals that over 40 percent of the sample has political science as its major. When you present a graph with percentages, it is good practice to include information about the sample size, as we have done in Figure 4.3.

We conclude the discussion of bar charts by introducing faceting. Specifically, we are interested in the question of whether the distribution of majors is different for men and women. This requires only one additional argument to the code we just provided:

Figure 4.4: A Facetted Bar Chart



Note: Note the separate plots for men ($n = 48$) and women ($n = 38$).

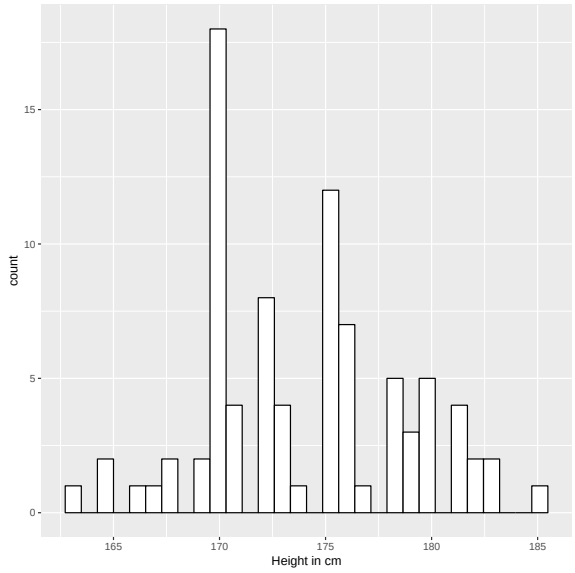
```
p4 <- p + geom_bar(aes(y = 100*((..count..)/sum(..count..))),
              col = "black", fill = "white") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.title.x = element_blank()) +
  ylab("Percent") +
  ggtitle("Hauptfach") +
  facet_wrap(~man)
```

The last line tells `ggplot2` to split the sample by the variable `man`. This results in separate bar charts for men and women, as is shown in Figure 4.4. The graph shows that the percentage of political science majors among men is higher than that among women. We also see that a higher percentage of men have history and economics majors, and that the diversity of majors is greater among female students (e.g., BWL, English, etc.).

4.3.3 Creating Histograms

Histograms are useful visualizations for interval and ratio scales. Their construction with `ggplot2` is quite easy, as we shall show in this section. Our example concerns the estimated heights of the UZH students. The following

Figure 4.5: A Histogram with Many Bins



Note: Histogram using `ggplot2`'s default bin width of range/30.

syntax creates a histogram with a color scheme similar to Figure 4.2:

```
p <- ggplot(dat2, aes(x = estheight))
p1 <- p + geom_histogram(col = "black", fill = "white") +
  xlab("Height in cm")
```

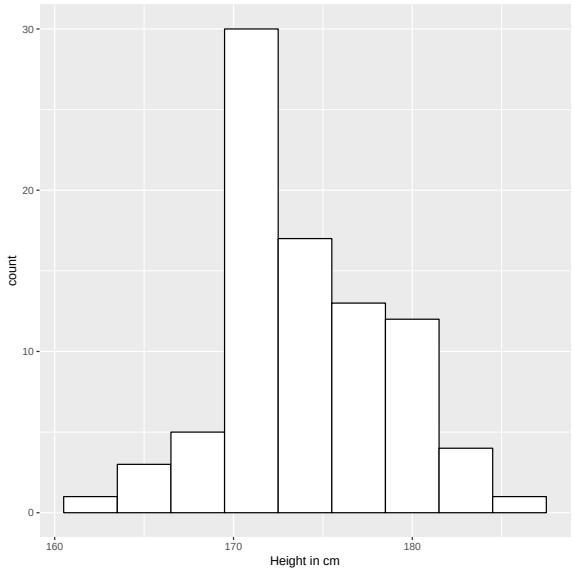
This graph utilizes the default of 30 bins, which results in gaps in the histogram (see Figure 4.5). As you can see, the geom is that of histogram. Also note that the horizontal axis label clearly indicates the metric (centimeters) of the variable. It is important that you always indicate this.

If we want to control the number of bins, then `ggplot2` requires that we do so by setting the bin width. To set the bin width according to Sturges, we can issue the following syntax:

```
wide <- ceiling((max(dat2$estheight) - min(dat2$estheight))/
                nclass.Sturges(dat2$estheight))
```

We then issue the following syntax for the geom:

Figure 4.6: A Histogram with Expanded Bin Widths



Note: Histogram using Sturges' procedure for setting the number of bins.

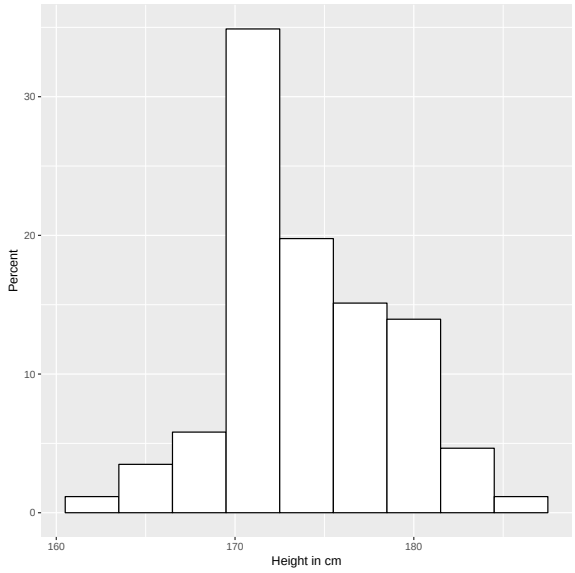
```
p2 <- p + geom_histogram(col = "black", fill = "white",
                         binwidth = wide) +
  xlab("Height in cm")
```

The result is the histogram shown in Figure 4.6, which contains eight bins. Parenthetically, the reason that `ggplot2` sets the number of bins to a relatively large number by default is that it encourages you to consider a reasonable amount of detail in the data. It considers this to be better statistical practice than the application of a somewhat arbitrary binning scheme like that of Sturges, although the latter approach characterizes R's default program for creating histograms, to wit `hist`.

The procedures shown so far produce histograms stated in terms of frequency counts. To generate histograms of percentages, we modify the command in the same manner as we did for bar charts:

```
p3 <- p + geom_histogram(aes(y = 100*((..count..)/sum(..count..))),
                         binwidth = wide, col = "black",
                         fill = "white") +
  xlab("Height in cm") +
  ylab("Percent")
```

Figure 4.7: A Histogram with Expanded Bin Widths and Percentages



Note: Histogram using Sturges' procedure for setting the number of bins. The vertical axis is stated in a percentage metric ($n = 86$).

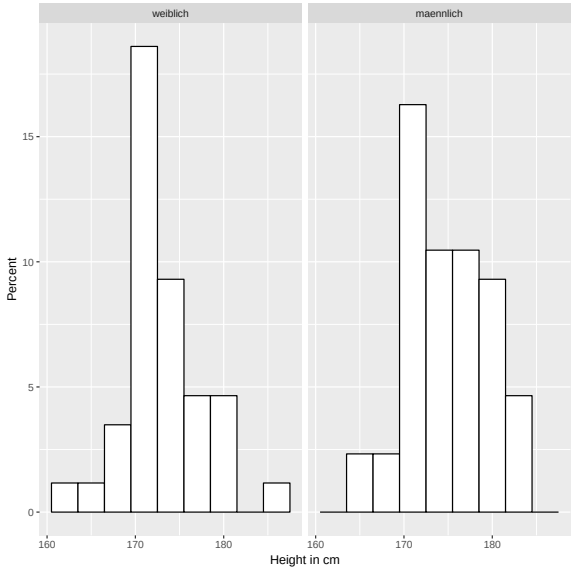
The result can be seen in Figure 4.7.

Finally, it is again possible to apply facets. In Figure 4.8, we show separate histograms for men and women. We created this graph using the following syntax:

```
p4 <- p + geom_histogram(aes(y = 100*((..count..)/sum(..count..))),
                  binwidth = wide, col = "black",
                  fill = "white") +
  xlab("Height in cm") +
  ylab("Percent") +
  facet_wrap(~man)
```

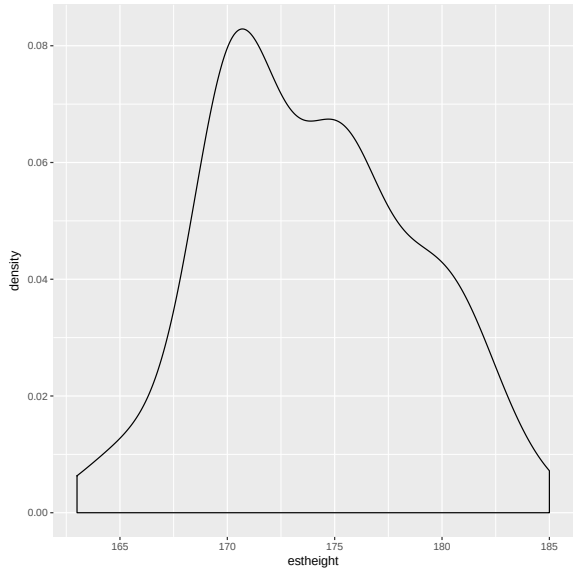
The graph clearly shows that men tend to be taller than women. As one indication of this pattern one could focus on the tallest bar, which corresponds to the most common height in the sample. For women, this tallest bar is lower than for men.

Figure 4.8: A Facetted Histogram



Note: Note the separate plots for men ($n = 48$) and women ($n = 38$).

Figure 4.9: A Density Plot



Note: This plot was created using the default settings in `ggplot2`.

4.3.4 Density Plots

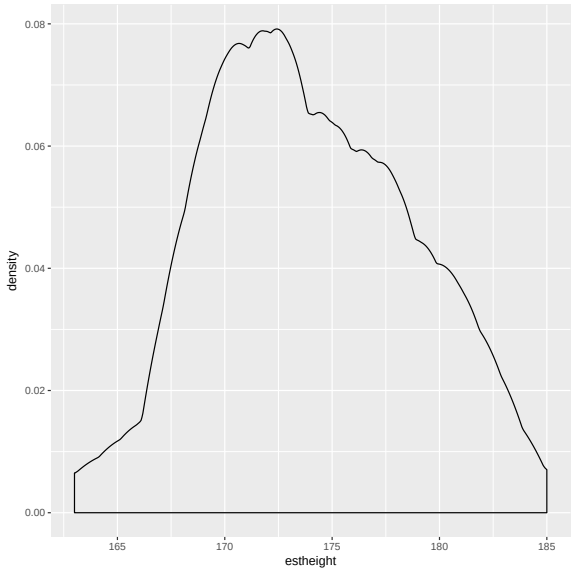
The final use of `ggplot2` that we want to illustrate in this chapter concerns the construction of density plots. We use students' heights again as an example. The geom for creating a density is `geom_density`. In its simplest form, the syntax for a density plot is:

```
library(ggplot2)
p1 <- ggplot(dat2, aes(x = estheight)) + geom_density()
```

This yields Figure 4.9, which is a continuous approximation of the distribution of height using the default settings in `ggplot2`.

If we want to control the appearance of the density plot, then two parameters are of particular interest. The first is the kernel that is being used. The default in `ggplot2` is a Gaussian kernel but many other options are available. Figure 4.10 illustrates the effect of the kernel on the shape of the density plot. Our personal favorites are Epanechnikov and Gaussian kernels, since they produce smooth curves while preserving detail, but you may find that another kernel works better for you. The key here is that you should choose a kernel that allows you to understand the empirical distribution of the variable of interest. To change the kernel to, for example, Epanechnikov, you can change the previous

Figure 4.10: The Impact of Kernel Selection



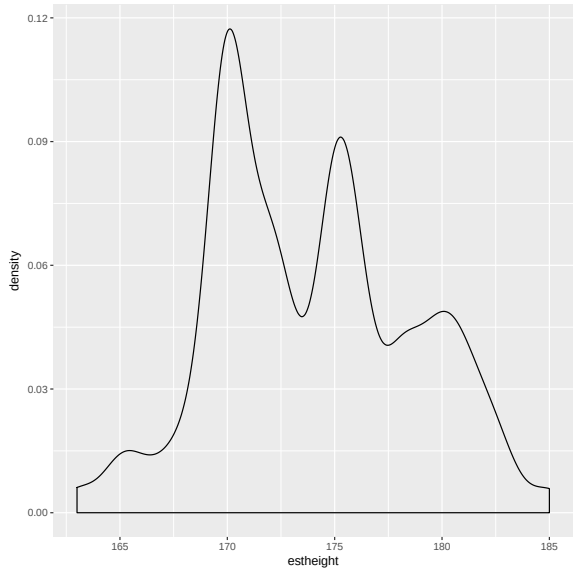
Note: The plot shows the effect of different kernel choices on the shape of the density plot.

syntax in the following manner:

```
library(ggplot2)
p2 <- ggplot(dat2, aes(x = estheight)) +
  geom_density(kernel="epanechnikov")
```

The second parameter that influences the shape of the density plot is the bandwidth, which determines how many points in the vicinity of the target point are considered when computing the density at the target point. Figure 4.11 shows the effect of different bandwidths on the density plot. When the bandwidth is set very low, for example, at one hundredth of the standard deviation (SD) of the kernel, then the resulting plot is extremely course and not very helpful in clarifying the shape of the distribution. However, too large of a bandwidth, for example, 3 standard deviations of the kernel, means that important details are lost. Again, the result is that the plot becomes useless. It is best to play around with different bandwidths and to decide on one that reveals both the overall shape of the distribution and important details such as spikes. In `ggplot2`, the bandwidth is controlled by an adjustment parameter. For example,

Figure 4.11: The Impact of Bandwidth



Note: The plot shows the effect of different bandwidth choices on the shape of the density plot. All plots use the Gaussian kernel.

```
library(ggplot2)
p3 <- ggplot(dat2, aes(x = estheight)) +
  geom_density(adjust=.5)
```

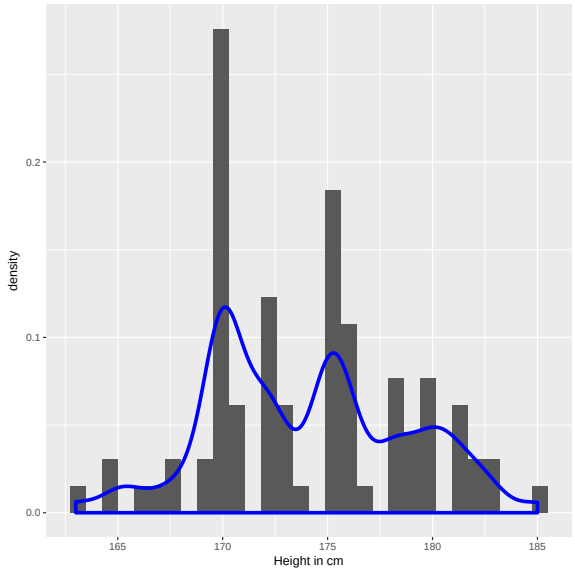
sets the adjustment to .5 times the bandwidth, which is equal to the standard deviation of the selected kernel.

Frequently, histograms and density plots are combined to give two different perspectives on the distribution of a variable. It is easy to do this, as long as you ensure that the y-axis for the histogram captures density (i.e., relative frequency) information and not raw frequencies.

```
library(ggplot2)
p4 <- ggplot(dat2, aes(x = estheight)) +
  geom_histogram(aes(y = ..density..)) +
  geom_density(adjust = .5, col = "blue", size = 1.5) +
  xlab("Height in cm")
```

The resulting graph is displayed in Figure 4.12. With a bandwidth of half of a standard deviation, we see that most of the spikes in the histogram are nicely

Figure 4.12: A Combined Histogram and Density Plot



Note: The density plot uses the Gaussian Kernel and a standard deviation of .5.

reflected in the density plot. Note that the size parameter causes the line of the density plot to be drawn thicker so that it is better visible.

We conclude the discussion of density plots by considering once more the differences between men and women. As we have done in the past, we can create separate density plots for men and women using the `facet_wrap` function:

```
library(ggplot2)
p5 <- ggplot(dat2, aes(x = estheight)) +
  geom_density(adjust = .5) +
  xlab("Height in cm") +
  facet_wrap(~man)
```

The result is shown in the top panel of Figure 4.13. However, we can also do the facetting within a single figure by drawing the density plots for men and women in different colors:

```
library(ggplot2)
library(gridExtra)

##
## Attaching package:  'gridExtra'
```

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

p6 <- ggplot(dat2, aes(x = estheight)) +
  geom_density(aes(col = factor(man)), adjust = .5) +
  xlab("Height in cm")
```

The result of this specification is shown in the bottom panel of Figure 4.13. Parenthetically, the use of colors to discriminate between men and women is also available for bar charts and histograms. We did not illustrate this option in those contexts because the end result is typically a graph that is too busy to be interpreted easily. Even with density plots, we would not use colors for more than three sub-groups, as this would also make for a graph where too much is going on all at once.

If you choose to work with colors, then you can easily control the choice of colors and the layout of the legend. This is shown in Figure 4.14. Panel (a) omits the legend title and is created by using the following syntax:

```
library(ggplot2)
p <- ggplot(dat2, aes(x = estheight)) +
  geom_density(aes(col = factor(man)), adjust = .5) +
  xlab("Height in cm")
p1 <- p + theme(legend.title = element_blank()) +
  ggtitle("Panel (a)")
```

Panel (b) changes the legend title and is obtained using:

```
library(ggplot2)
p2 <- p + scale_color_discrete(name = "Geschlecht:") +
  ggtitle("Panel (b)")
```

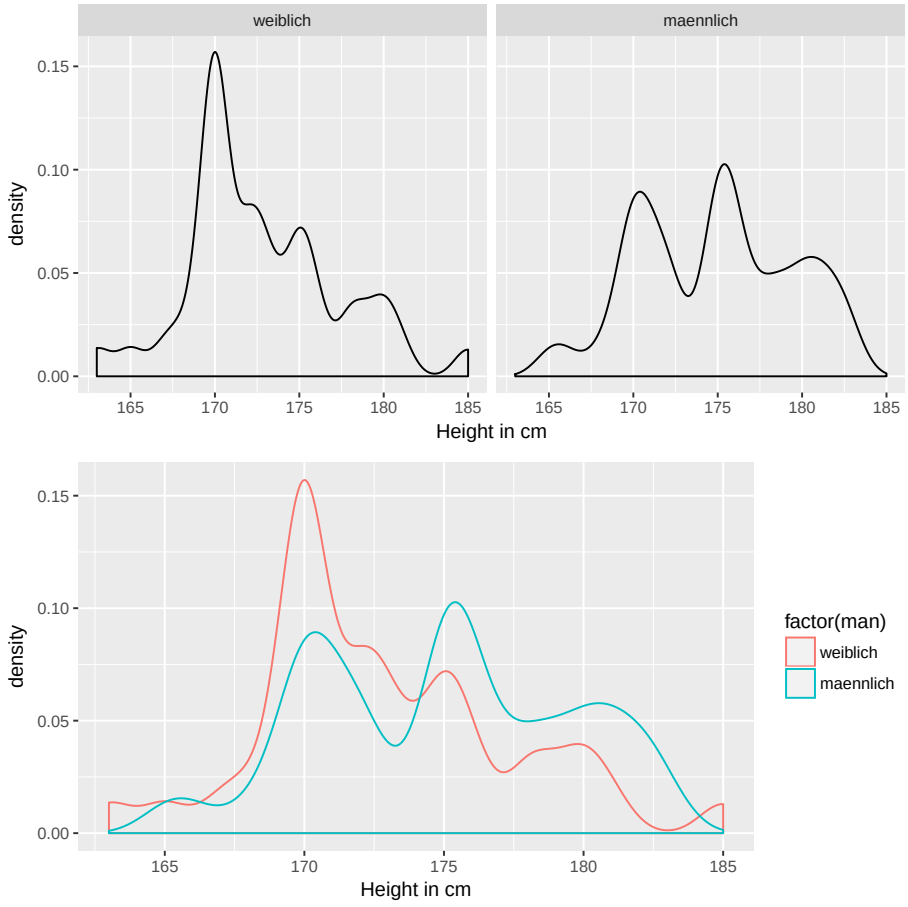
Panel (c) changes the legend title and labels and requires the following syntax:

```
library(ggplot2)
p3 <- p + scale_color_discrete(name = "Sex:",
                               labels = c("Female", "Male")) +
  ggtitle("Panel (c)")
```

Finally, panel (d) changes the legend title, labels, and colors:

```
p4 <- p + scale_color_manual(values = c("#E69F00", "#56B4E9"),
                             name = "Sex:",
                             labels = c("Female", "male")) +
  ggtitle("Panel (d)")
```

Figure 4.13: Facetted Density Plots



Note: The top panel shows separate density plots for men and women. The bottom panel differentiates between men and women by using colors.

Codes such as #E69F00 reference the color palette as a hexadecimal red-green-blue (RGB) triplet of intensities of those colors. Details can be found in the Cookbook for R.

4.4 Conclusions

In this chapter, we have taken a look at tabular and visual displays of frequency distributions. We introduced the `table` command for creating frequency tables and the `ggplot2` library for generating bar charts, histograms, and density plots. I hope you agree that R is capable of generating very attractive graphics. This is one of R's many strengths. Well designed graphs and tables convey important information and play a central role in the presentation of statistical results. As a result, this will certainly not be the last time we will use the commands and package introduced in this chapter. Hopefully, you now have a basic appreciation of how graphics and tables are built in R.

4.5 Exercises

(1) Load `world indicators.dta`. Now create a frequency table displaying the democracy indicator. How many democracies are there in the data? And how many non-democracies are there?

(2) Redo the table from (1) so that it shows the percentages of democracies and non-democracies.

(3) Take a look at the variable `polstab`, which measures political stability. Tabulate the frequency distribution for this variable after you have created bins using Sturges' procedure.

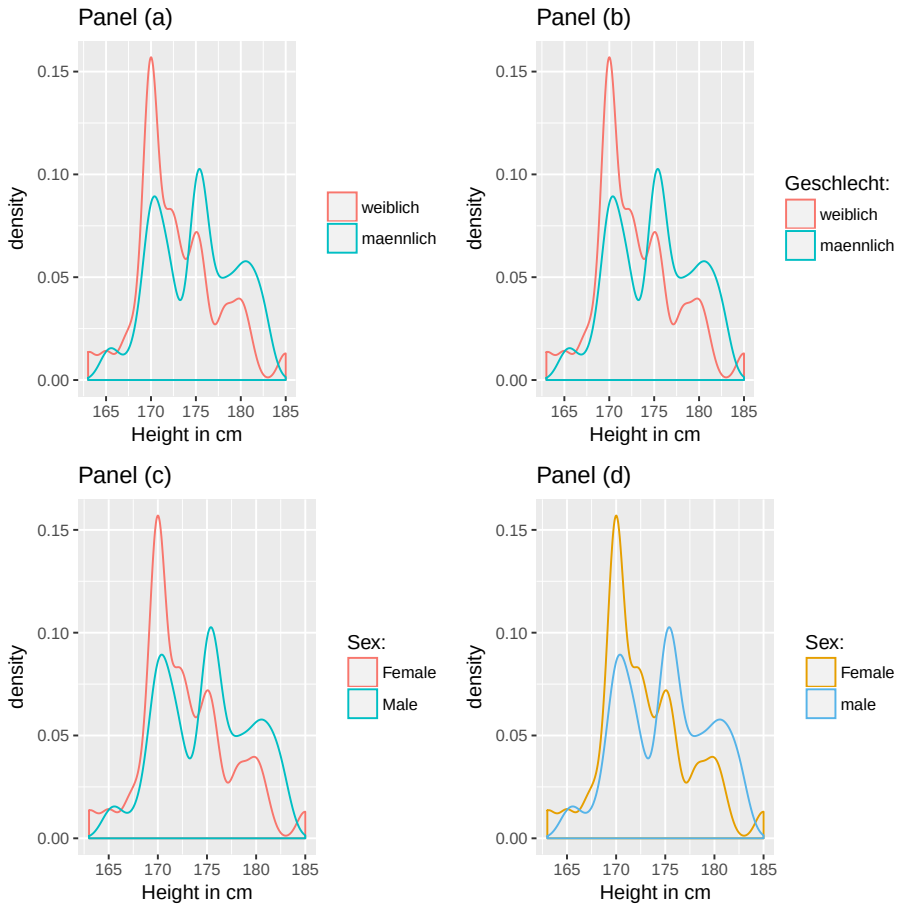
(4) For the binned version of political stability, generate a data frame showing the intervals, frequencies, proportions, and cumulative frequencies. Which interval is most common? What is the cumulative frequency for this interval and how do you interpret this? What percentage of cases fall into this interval?

(5) Create a bar chart of the democracy indicator where the vertical axis shows percentages.

(6) Create a histogram of political stability using Sturges' binning rule and letting the vertical axis show percentages.

(7) Create a facetted version of the political stability histogram where the sub-groups correspond to democracies and non-democracies respectively.

Figure 4.14: Legends and Colors in Density Plots



Note: The panels show different colors and legends that can be created using `ggplot2`.

(8) Create a density plot using an Epanechnikov kernel and an appropriate bandwidth for the corruption variable (`corrupt`). Justify why you chose the bandwidth.

(9) Repeat the exercise in (8) but now overlaying the density function on top off a histogram. How well do the spikes of the density function capture the spikes in the histogram? Could this be improved by choosing a different bandwidth?

(10) Within the same plot, show the density of corruption for democracies and for non-democracies. What does the graph tell you?

Chapter 5

Summary Statistics

In Chapter 2, we introduced the `summary` command. This produced a number of summary statistics that, at the time, made little sense. Now that we have discussed summary statistics in the lecture, it becomes possible to work with them. The current chapter describes a number of methods in R for obtaining summary statistics. In addition, we discuss how those statistics should be interpreted.

5.1 Required Packages

For the exercises in this chapter, you will need to install and activate the following libraries:

- `ggplot2`
- `moments`
- `psych`

5.2 Measures of Central Tendency

5.2.1 The Mode

R does not have a built-in function for computing the mode, but one can be programmed quite easily:[1]

```
Mode <- function(x, na.rm=FALSE) {
  if(na.rm){
    x <- na.omit(x)
  }
  ux <- unique(x)
```

[1]The function shown here was proposed by Ken Williams at Stackoverflow.

```
  tab <- tabulate(match(x, ux)); ux[tab == max(tab)]
}
```

We apply this function to the world political and economic indicators (`world indicators.dta`), specifically to the Polity variable:

```
library(foreign)
world <- read.dta("world indicators.dta")
Mode(world$polity)

## [1] NA
```

We see that the mode is `NA`, which is actually correct because the Polity indicator is missing for most of the countries in the data frame. Still, this result is probably not what we are looking for. We would much rather obtain the mode for the subset of countries where a Polity score is available. To accomplish this, we use the `na.rm=TRUE` to remove the texttttNAs from the sample, before calculating the mode:

```
Mode(world$polity,na.rm=TRUE)

## [1] 10
```

This shows that the most frequent score on the Polity variable is 10 when the score is actually available. This is easily verified using the `table` function:

```
table(na.omit(world$polity))

##
## -88 -77 -66 -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1
##   1   2   2   2   4   3  11   3   2   5   4   7   3   2   2
##   2   3   4   5   6   7   8   9  10
##   3   4   5   6  12  12  21  15  34
```

5.2.2 The Median

R has a built-in function for the median. To use this function, it is again important to understand how to handle missing data. If a variable contains missing values and we apply the function `median` without any options, then R returns `NA`. To prevent this from happening, you should add the option `na.rm = TRUE`, which removes the missing values. For example,

```
median(world$hdi, na.rm = TRUE)

## [1] 0.717
```

produces the median value of the human development index (HDI), in this case 0.717. We can interpret this as follows: half of the HDI values fall below or at 0.717 and half of the values exceed or are at 0.717.

5.2.3 Means

Arithmetic Mean In R, computation of the arithmetic mean follows the same logic as the median. For example, to compute the mean of the HDI we should issue the following syntax:

```
mean(world$hdi, na.rm = TRUE)

## [1] 0.685631
```

Again, the option of `na.rm = TRUE` is mandatory, as R would otherwise return `NA` if there are any missing data.

Geometric Mean The geometric mean is useful in the context of proportional growth. The variable `growth` in the data set `world indicators` gives the population growth in each country in 2013 (relative to 2012). The geometric mean requires that the values are positive, so that we need to deselect those countries for which growth was flat or negative. We do this with the `dplyr` library. For the geometric mean itself, we use the `psych` package.

```
library(dplyr)
library(psych)

##
## Attaching package:  'psych'
## The following objects are masked from 'package:ggplot2':
##
##    %+%, alpha

growth <- filter(world, growth > 0)
geometric.mean(growth$growth, na.rm = TRUE)

## [1] 1.201933
```

As another example, consider the following financial application. A person invests 1,000 Swiss Francs. After the first year, her stocks are worth 1,200 Francs. After two years, they are worth 1,250 Francs, and after the third year the worth is CHF1,400. The growth rates are $(1200-1000)/1000 = .200$ for year 1, $(1250-1200)/1200 = .042$ for year 2, and .120 for year 3. Or, we multiply the original investment by 1.200 in the first year, by 1.042 in the second year, and by 1.120 in the third year. We enter the data and compute the geometric mean as:

```
library(psych)
invest <- c(1.200, 1.042, 1.120)
geometric.mean(invest)

## [1] 1.118808
```

(We do not need to add `na.rm = TRUE` because there are no missing values.) This is the constant growth rate that would yield the same final amount: in year 1, we get 1119; in year 2, we get 1252; and, in year 3, we get 1400. Note that the amount produced for the third year exactly replicates the worth of the investment in that year. This would not have happened had we computed the arithmetic mean. It is easily verified that the arithmetic mean is 1.121. This is too high, however, as it would suggest an investment worth of 1407 in year 3, more than was actually realized.

Harmonic Mean The harmonic mean is used most often when the variable is a rate and/or when the effect of atypical data points should be minimized. The world economic, political, and social indicator data include the variable `homicide`, which gives the number of homicides per 100,000 inhabitants in various countries. To compute the harmonic mean, we proceed as follows.

```
library(psych)
harmonic.mean(world$homicide, na.rm = TRUE)

## [1] 0
```

This is equal to the reciprocal of the arithmetic mean of the reciprocal homicide rates. On the average, we would say that the homicide rate is 0. We see that the harmonic mean is weighted toward the lower values of the homicide rates. This is one of the properties of harmonic means.

To understand the idea of harmonic means a bit better, let us look at another, non-political example. Imagine you are a cook and you are expected to deliver a number of chicken pot pies. You can prepare 12 pot pies per hour. With your oven capacity, you can cook 4 pies per hour. What is the overall production rate? The average is 6 pies per hour *for each stage* of the cooking process. If we could replace the two phases with a process that yields 6 pies an hour in each phase, we would get the same result. Again, note that the harmonic mean is weighted down toward the slower of the two production processes.

5.3 Quantiles

Consider the human development index in `world indicators.dta`. We would like to identify the quartiles for this variable.[2] As it turns out, this is quite simple in R:

[2]The United Nations use the quartile values of the component indicators of the HDI to group countries. Here, we are interested in the quartiles of the HDI itself.

```
quantile(world$hdi, probs = c(0.25, 0.50, 0.75), na.rm = TRUE, type = 6)

##   25%   50%   75%
## 0.561 0.717 0.812
```

The `probs` argument is critical in specifying the quantiles one is looking for: Here, we have specified values of 0.25, 0.50, and 0.75 to capture the 1st, 2nd, and 3rd quartiles. As we have seen before, `na.rm = TRUE` removes the missing values prior to computing the quartiles. The final argument is `type = 6`, which influences the method for computing the quantiles. Here, we have specified method 6, which corresponds to what we discussed in class. Substantively, the results imply that 25 percent of the countries are at or below an HDI value of 0.561, that 50 percent of the countries are at or below an HDI value of 0.717, and that 75 percent of the countries are at or below an HDI value of 0.812.

If instead we would have liked to compute quintiles, then the only argument that would have to be adjusted is the `probs` argument. Specifically,

```
quantile(world$hdi, probs = c(0.2, 0.4, 0.6, 0.8, 1.0), na.rm = TRUE,
         type = 6)

##    20%    40%    60%    80%   100%
## 0.5168 0.6684 0.7486 0.8300 0.9440
```

provides the 1st, 2nd, 3rd, 4th, and 5th HDI quintiles.

5.4 Dispersion

5.4.1 The Inter-Quartile Range

In R, the inter-quartile range (IQR) can be computed using the `IQR` command. The following syntax applies the command to the human development index:

```
IQR(world$hdi, na.rm = TRUE, type = 6)

## [1] 0.251
```

Here `type` again controls the method of computing the 1st and 3rd quartiles that guide the IQR.

5.4.2 The Range

R has a range command but it does not actually produce the range in the classical sense of the term. Instead, it shows the minimum and maximum values of a variable. For example, applying

```
range(world$hdi, na.rm = TRUE)

## [1] 0.337 0.944
```

shows that the minimum HDI in the data is 0.337 and that the maximum is 0.944. The typical definition of the range is the difference between the maximum and minimum of a variable. It is easy to adjust the `range` command to obtain this result:

```
range(world$hdi, na.rm = TRUE)[2] -
  range(world$hdi, na.rm = TRUE)[1]

## [1] 0.607
```

Here `range()[2]` references the maximum and `range()[1]` the minimum. The difference between those values is the range. In this case, the range is 0.607, which is considerable given that the maximum conceivable range on the HDI is 1.

5.4.3 The Variance and Standard Deviation

To obtain the variance, we issue the following syntax:

```
var(world$hdi, na.rm = TRUE)

## [1] 0.02435892
```

This gives us the variance of HDI in the world economic, political, and social indicators data. To obtain the standard deviation, the following syntax suffices:

```
sd(world$hdi, na.rm = TRUE)

## [1] 0.1560735
```

The maximum possible standard deviation on a 0-1 variable such as the HDI is .5. Our standard deviation is 31 percent of this maximum, which may be considered a moderate degree of dispersion.

5.5 Skewness and Kurtosis

Sticking with the human development index, let's consider its skewness and kurtosis. To compute these statistics, we need the `moments` library. The skewness of the HDI can be computed using

```
library(moments)
skewness(world$hdi, na.rm = TRUE)

## [1] -0.4296318
```

We observe that the HDI has a negative skew.

The kurtosis of the HDI is obtained through the following code:

```
library(moments)
kurtosis(world$hdi, na.rm = TRUE)

## [1] 2.197094
```

To obtain the excess kurtosis we need to subtract 3. This is done very easily:

```
library(moments)
kurtosis(world$hdi, na.rm = TRUE) - 3

## [1] -0.8029064
```

This shows that the HDI distribution is slightly platykurtic.

5.6 Box Plots

We conclude this chapter by discussing the construction of box plots. For this we need to use the `ggplot2` library. The relevant geom is `geom_boxplot`. By default, this geom assumes that you want to apply facetting. We can use this to generate a box plot of HDI for each of the continents in the data:

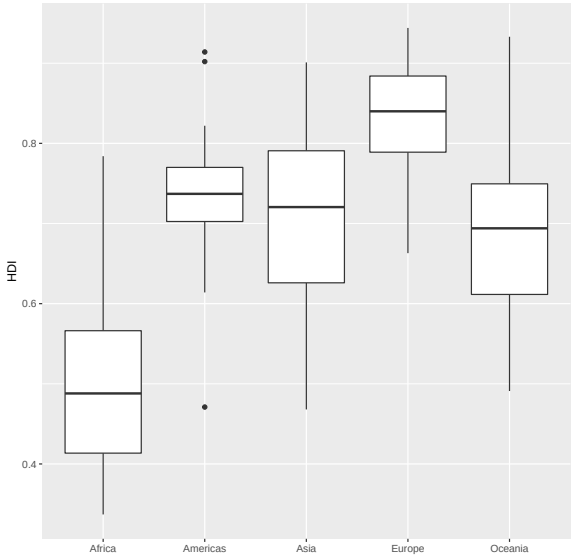
```
library(ggplot2)
p1 <- ggplot(world, aes(factor(continent), hdi)) +
  geom_boxplot() + xlab("") + ylab("HDI")
```

The resulting graph is shown in Figure 5.1. This figure clearly shows, for example, that the human development index has a comparatively high median and small spread—the box is narrow—in Europe. It also shows that there are several outliers in the Americas. The outliers with an unusually high HDI (for the Americas) are Canada and the USA, whereas the outlier with an unusually low HDI is Haiti.

Box plots are at their most useful when we compare distributions like we have done in Figure 5.1. However, you can trick `ggplot` into creating a box plot of a variable without applying facetting:

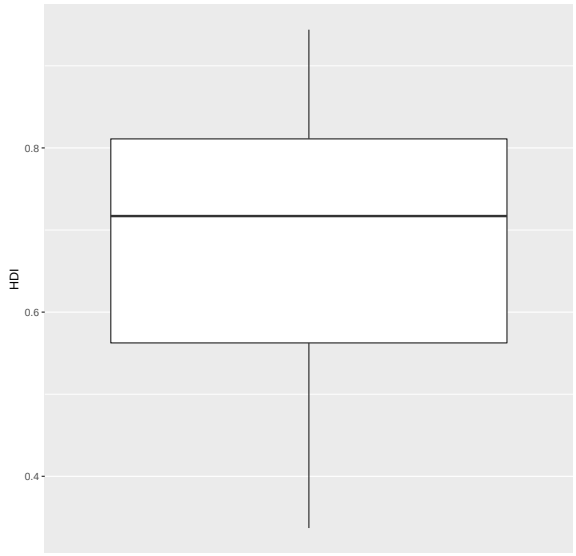
```
p2 <- ggplot(world, aes(factor(1), hdi)) +
  geom_boxplot() + xlab("") + ylab("HDI") +
  scale_x_discrete(breaks=NULL)
```

Figure 5.1: Box Plot of Human Development by Continent



Note: The plot shows clear differences in the central tendency and spread of the HDI across continents.

Figure 5.2: Box Plot of Human Development



Note: No facetting has been applied.

The trick is the specification of the dummy factor 1. Since this would display an unsightly 1 on the horizontal axis, we add the scale option to suppress it. The result is shown in Figure 5.2. Looking at the location of the first and third quartiles relative to the median, as well as the length of the whiskers, it is clear that there is a tail in the direction of lower values of the HDI. This is consistent with the negative skewness coefficient we observed earlier.

5.7 Conclusions

In this chapter, we showed how R can be used to obtain any number of summary statistics about single variables. In the next two chapters, we shall focus on methods for describing relationships between two variables at a time. Before going there, however, you will have an opportunity to practice what you learnt in this chapter.

5.8 Exercises

(1) Load `world indicators.dta`. This data contains a variable `demindx`, which is the democracy indicator of the Economist Intelligence Unit. It ranges

from 0 to 10, with higher values indicating a better functioning democracy. Compute the mode of the democracy index, ignoring those countries that have missing values on the index. Interpret the result.

(2) Now compute the median of the democracy index, again ignoring countries with missing values on the index. Interpret the result.

(3) Compute the arithmetic mean of the democracy index, ignoring once more countries with missing values. Interpret the result.

(4) Compute the quartiles of the democracy index and interpret the result.

(5) What is the inter-quartile range of the democracy index?

(6) What is the variance of the democracy index? Is this large or small compared to the maximum possible variance?

(7) Compute the skewness of the democracy index and interpret the result.

(8) Compute the excess kurtosis of the democracy index and interpret the result.

(9) Limit your analysis to the European continent. Now create a box plot of the democracy index where you facet by sub-continent. Interpret the resulting graph.

(10) The data set `world indicators.dta` includes two indicators for democracy in 2013, to wit, the aforementioned democracy index and the Polity democracy score (`politydemo`). Write a function that computes the geometric mean of these two indicators. What value of the geometric mean do you obtain for Switzerland?

Chapter 6

Contingency Tables

In this chapter, we consider the construction of contingency tables, as well as the computation of measures of association for contingency tables. When your variables are nominal or ordinal in nature, contingency tables often are the best way to display and analyze relationships. R has some basic capabilities for the creation and analysis of contingency tables. Through various add-on libraries, these capabilities can be expanded so that they cover everything we have discussed in the course.

6.1 Required Packages

For the exercises in this chapter, you will need to install and activate the following libraries:

- `epitools`
- `gmodels`
- `PResiduals`
- `rapport`
- `vcd`

6.2 Creating a Contingency Table

6.2.1 Twoway Tables

For the creation of a two-way contingency table—i.e., a table of two variables—one could use R's built-in `table` command. However, we like to use the `gmodels` library because it automates a number of basic functions, such as marginals and percentages. To illustrate the use of this library, let us consider the democracy indicator in `world indicators.dta` and tabulate it against continent.

```
library(foreign)
library(gmodels)
world <- read.dta("world indicators.dta")
CrossTable(world$continent, world$demo, digits = 3,
  expected = FALSE, prop.r = FALSE, prop.c = FALSE,
  prop.t = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                  | world$demo
## world$continent | Non-Democratic |     Democratic |      Row Total |
## ----------------|----------------|----------------|----------------|
##          Africa |             41 |              8 |             49 |
## ----------------|----------------|----------------|----------------|
##        Americas |              7 |             17 |             24 |
## ----------------|----------------|----------------|----------------|
##            Asia |             36 |              9 |             45 |
## ----------------|----------------|----------------|----------------|
##          Europe |              3 |             33 |             36 |
## ----------------|----------------|----------------|----------------|
##         Oceania |              2 |              3 |              5 |
## ----------------|----------------|----------------|----------------|
##    Column Total |             89 |             70 |            159 |
## ----------------|----------------|----------------|----------------|
##
##
```

Notice that the row variable is declared first and the column variable is defined second. We observe that there are 41 non-democratic African countries, 8 democratic African countries, 7 non-democratic American countries, 17 democratic American countries, etc.[1]

The table we just created is barebones, since it only reports frequencies. We can easily add proportions, however. To add proportions that take n as their base, we change `prop.t = FALSE` to `prop.t = TRUE`.

[1] The option `digits = 3` has the effect that any results reported in the cells have a precision of three digits. Obviously, one can change this number. The option may also be left out altogether.

```
library(gmodels)
CrossTable(world$continent, world$demo, digits = 3,
  expected = FALSE, prop.r = FALSE, prop.c = FALSE,
  prop.t = TRUE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                  | world$demo
## world$continent | Non-Democratic |     Democratic |      Row Total |
## ----------------|----------------|----------------|----------------|
##          Africa |             41 |              8 |             49 |
##                 |          0.258 |          0.050 |                |
## ----------------|----------------|----------------|----------------|
##        Americas |              7 |             17 |             24 |
##                 |          0.044 |          0.107 |                |
## ----------------|----------------|----------------|----------------|
##            Asia |             36 |              9 |             45 |
##                 |          0.226 |          0.057 |                |
## ----------------|----------------|----------------|----------------|
##          Europe |              3 |             33 |             36 |
##                 |          0.019 |          0.208 |                |
## ----------------|----------------|----------------|----------------|
##         Oceania |              2 |              3 |              5 |
##                 |          0.013 |          0.019 |                |
## ----------------|----------------|----------------|----------------|
##    Column Total |             89 |             70 |            159 |
## ----------------|----------------|----------------|----------------|
##
##
```

We now observe, for example, that the proportion of non-democratic countries that reside in Oceania is 0.013.

To add proportions that take the column totals as their base, we change `prop.c = FALSE` to `prop.c = TRUE`.

```
library(gmodels)
CrossTable(world$continent, world$demo, digits = 3,
  expected = FALSE, prop.r = FALSE, prop.c = TRUE,
  prop.t = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                  | world$demo
## world$continent | Non-Democratic |     Democratic |      Row Total |
## ----------------|----------------|----------------|----------------|
##          Africa |             41 |              8 |             49 |
##                 |          0.461 |          0.114 |                |
## ----------------|----------------|----------------|----------------|
##        Americas |              7 |             17 |             24 |
##                 |          0.079 |          0.243 |                |
## ----------------|----------------|----------------|----------------|
##            Asia |             36 |              9 |             45 |
##                 |          0.404 |          0.129 |                |
## ----------------|----------------|----------------|----------------|
##          Europe |              3 |             33 |             36 |
##                 |          0.034 |          0.471 |                |
## ----------------|----------------|----------------|----------------|
##         Oceania |              2 |              3 |              5 |
##                 |          0.022 |          0.043 |                |
## ----------------|----------------|----------------|----------------|
##    Column Total |             89 |             70 |            159 |
##                 |          0.560 |          0.440 |                |
## ----------------|----------------|----------------|----------------|
##
##
```

We now observe that 46.1 percent of the world's non-democracies reside in Africa and that 47.1 percent of the world's democracies reside in Europe.

To add proportions that take the row totals as their base, we change `prop.r = FALSE` to `prop.r = TRUE`.

```
library(gmodels)
CrossTable(world$continent, world$demo, digits = 3,
  expected = FALSE, prop.r = TRUE, prop.c = FALSE,
  prop.t = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
##
##
## Total Observations in Table:  159
##
##
##                  | world$demo
## world$continent | Non-Democratic |     Democratic |      Row Total |
## ----------------|----------------|----------------|----------------|
##          Africa |             41 |              8 |             49 |
##                 |          0.837 |          0.163 |          0.308 |
## ----------------|----------------|----------------|----------------|
##        Americas |              7 |             17 |             24 |
##                 |          0.292 |          0.708 |          0.151 |
## ----------------|----------------|----------------|----------------|
##            Asia |             36 |              9 |             45 |
##                 |          0.800 |          0.200 |          0.283 |
## ----------------|----------------|----------------|----------------|
##          Europe |              3 |             33 |             36 |
##                 |          0.083 |          0.917 |          0.226 |
## ----------------|----------------|----------------|----------------|
##         Oceania |              2 |              3 |              5 |
##                 |          0.400 |          0.600 |          0.031 |
## ----------------|----------------|----------------|----------------|
##    Column Total |             89 |             70 |            159 |
## ----------------|----------------|----------------|----------------|
##
##
```

We now see, for example, that 83.7 percent of the African countries are non-democratic and that 91.7 percent of Europe is democratic.

6.2.2 Multiway Tables

On occasion, we may want to tabulate multipl variables. For this purpose, we recommend using the `xtabs` command. Imagine that we want to explore the relationship between human development, democracy, and political stability. We dichotomize human development using the median and political stability at the 50 percentile mark. Let's start by performing these operations.

```
lbl <- c("Low", "High")
world$dihdi <- cut(world$hdi,
  breaks = c(0, median(world$hdi, na.rm = TRUE), 1),
  labels = lbl, right = FALSE)
table(world$dihdi)

##
##  Low High
##   93   94

world$distab <- cut(world$polstab, breaks = c(0, 50, 101),
  labels = lbl, right = FALSE)
table(world$distab)

##
##  Low High
##  105  106
```

Having created the variables, we can now set up the table:

```
xtabs(~distab + demo + dihdi, data = world)

## , , dihdi = Low
##
##       demo
## distab Non-Democratic Democratic
##   Low              52         13
##   High              8          7
##
## , , dihdi = High
##
##       demo
## distab Non-Democratic Democratic
##   Low              21         10
##   High              7         38
```

We see that we obtain two 2×2 sub-tables, one for low and one for high levels of HDI. We see, for example, that 52 non-democratic, low HDI countries suffer from low political stability. Thirty-eight democratic, high HDI countries have high

political stability, etc. We can also present the frequencies a bit differently—and many would say better—by using the `ftable` command:

```
mytable <- xtabs(~distab + demo + dihdi, data = world)
ftable(mytable)

##                         dihdi Low High
## distab demo
## Low    Non-Democratic        52   21
##        Democratic            13   10
## High   Non-Democratic         8    7
##        Democratic            7   38
```

This provides exactly the same information, albeit in a single table.

One might want to compute proportions. For example, one might be interested in determining what portion of non-democratic, low HDI countries suffer from low political stability. Here, R's `prop.table` command comes in handy. We want to apply this to each of the sub-tables created by `xtabs`. Specifically,

```
prop.table(mytable[,,1], 2)

##       demo
## distab Non-Democratic Democratic
##   Low       0.8666667  0.6500000
##   High      0.1333333  0.3500000

prop.table(mytable[,,2], 2)

##       demo
## distab Non-Democratic Democratic
##   Low       0.7500000  0.2083333
##   High      0.2500000  0.7916667
```

The command turns an existing table into a table of proportions. By adding `[,,1]` we select the first sub-table (the one for low HDI); by adding `[,,2]` we select the second sub-table (the one for high HDI). The `prop.table` command can use various bases of proportions. By specifying 2 at the end of the command, we ensure that column totals are used as the base. Had we specified 1, then the row totals are used as the base. Finally, no specification means that n serves as the base of the proportions.

Substantively, the results show that roughly 87 percent of the non-democratic, low HDI countries suffer from low political stability. For democratic, low HDI countries, this percentage is 65. The results also show that 75 percent of the non-democratic, high HDI countries suffer from low political stability. Finally, only 21 percent of democratic, high HDI countries suffer from low political sta-

bility. These results suggest that the combination of democracy and high HDI seems to be the best guarantee for political stability.

6.3 Relative Risk and the Odds Ratio

6.3.1 The Relative Risk Ratio

Imagine we want to infer the risk of political instability from the status of a country qua democracy. More precisely, we want to know if non-democracies have a higher risk of being unstable than democracies. To answer this question, we need to compute the relative risk ratio. Although it is not hard to do this by hand, it turns out that the `epitools` library will do the computation for us. The syntax is as follows:

```
library(epitools)
epitab(world$demo, world$distab, method = "riskratio",
       rev = "both")

## $tab
##                 Outcome
## Predictor        High        p0 Low        p1 riskratio
##   Democratic       46 0.6571429  24 0.3428571  1.000000
##   Non-Democratic   15 0.1685393  74 0.8314607  2.425094
##                 Outcome
## Predictor           lower    upper      p.value
##   Democratic           NA       NA           NA
##   Non-Democratic 1.730364 3.398752 4.153456e-10
##
## $measure
## [1] "wald"
##
## $conf.level
## [1] 0.95
##
## $pvalue
## [1] "fisher.exact"
```

Before interpreting the results, let us dissect the syntax. The `epitab` command generates a contingency table, in this case of `demo` as the row variable representing the risk factor, and `distab` as the column variable representing the outcome. The `method` option allows us to select the relative risk versus the odds ratio, among other things. Finally, the `rev` option allows us to change around the rows and/or columns of the table. Here we do both. If we ask R to construct a table of the two variables, we get[2]

[2]Here you see the use of the `table` command for creating contingency tables.

```
table(world$demo, world$distab)

##
##                   Low High
##   Non-Democratic   74   15
##   Democratic       24   46
```

The `epitools` library would now define the outcome of interest in terms of the second column, which is high political stability. However, we are interested in *low* stability as the outcome, which requires that we reverse the order of the columns. In terms of the row variable, `epitools` would define the relative risk ratio comparing the second to the first row. But this means we would compare democracies to non-democracies, whereas we would like to reverse the comparison. To accomplish this, we reverse the order of the rows as well.

Looking at the `epitable` output, for now only columns 1-6 are of interest.[3] Column 1 shows the levels of the risk factor. Column 2 shows the number of cases where low stability is absent, i.e., where political stability is high, for both levels of the risk factor. Column 3 shows the proportion of democracies and non-democracies, respectively, where the outcome is negative, i.e., low stability is absent. Column 4 shows the number of cases where low stability is present, doing this for both levels of the risk factor. Column 5 shows the proportion of democracies and non-democracies, respectively, where the outcome is positive, i.e, stability is low. Finally, column 6 shows the relative risk ratio. To ascertain the relative risk ratio, we look only at the second row, the one corresponding to non-democracies. We see that the relative risk ratio is approximately 2.425, which is equal to the ratio of `p1` for non-democracies and democracies. We interpret this as follows: compared to democracies, non-democracies have a roughly 2.425 higher risk of having low political stability.[4]

6.3.2 The Odds Ratio

The `epitable` command can also be used to compute the odds ratio. The only change in the syntax that is required is to replace `method = "riskratio"` with `method = "oddsratio"`. Thus, to compute the odds ratio for low political stability we issue the following syntax:

```
library(epitools)
epitab(world$demo, world$distab, method = "oddsratio",
       rev = "both")

## $tab
##                   Outcome
```

[3] The remaining columns are relevant for testing the statistical significance of the relative risk ratio, a topic we shall discuss later in this course.

[4] The relative risk ratio shown in the first row is uninteresting because it is based on a comparison of democracies with themselves.

```
## Predictor        High        p0 Low       p1 oddsratio
##   Democratic       46 0.7540984  24 0.244898  1.000000
##   Non-Democratic   15 0.2459016  74 0.755102  9.455556
##                 Outcome
## Predictor           lower   upper      p.value
##   Democratic           NA      NA           NA
##   Non-Democratic 4.499285 19.8715 4.153456e-10
##
## $measure
## [1] "wald"
##
## $conf.level
## [1] 0.95
##
## $pvalue
## [1] "fisher.exact"
```

Again, only columns 1-6 from the output are relevant for now. Key to the interpretation is column 6, which shows the odds ratio to be roughly 9.456. Thus, the odds of low compared to high political stability are almost 9.5 times higher for non-democracies than they are for democracies.[5]

6.4 Measures of Association for Nominal Variables

6.4.1 Cramér's V

Earlier, we cross-tabulated continent with democracy. In this section, we shall compute measures of association between these two variables. We begin by treating the two variables symmetrically, so that we do not make a distinction between a dependent and an independent variable. For this purpose Cramér's V is an appropriate choice. In R, V can be computed using the `vcd` library.

```
library(vcd)
```

```
## Loading required package:  grid
##
## Attaching package:  'vcd'
## The following object is masked from 'package:epitools':
##
##    oddsratio
```

```
mytable <- table(world$continent, world$demo)
assocstats(mytable)
```

[5] Compared to the relative risk ratio, `p0` and `p1` are now not computed within rows (using row totals as the base) but within columns (using column totals as the base).

```
##                      X^2 df   P(> X^2)
## Likelihood Ratio 73.137  4 4.8850e-15
## Pearson          66.470  4 1.2612e-13
##
## Phi-Coefficient   : NA
## Contingency Coeff.: 0.543
## Cramer's V        : 0.647
```

We first create a table—`mytable`—and then apply the `assocstats` function to it. This generates a large number of statistics, including the Pearson chi-squared statistic that serves as the basis of V. The statistic that interests us, however, is the last one. We see that Cramér's V is 0.647, which can be considered evidence of a strong association between the two variables.

6.4.2 Goodman and Kruskal's λ

Now let us analyze the relationship between continent and democracy from a slightly different perspective. Specifically, we want to predict the status qua democracy of a country based on the continent in which it is located. For this purpose, we can use Goodman and Kruskal's λ. The computation of this statistic can be carried out using the `rapportools` library.

```
library(rapportools)
```

```
## Loading required package:  reshape
##
## Attaching package:  'reshape'
## The following objects are masked from 'package:tidyr':
##
##    expand, smiths
## The following object is masked from 'package:dplyr':
##
##    rename
##
## Attaching package:  'rapportools'
## The following objects are masked from 'package:moments':
##
##    kurtosis, skewness
## The following object is masked from 'package:dplyr':
##
##    n
## The following objects are masked from 'package:stats':
##
##    IQR, median, sd, var
```

```
## The following objects are masked from 'package:base':
##
##    max, mean, min, range, sum

mytable <- table(world$continent, world$demo)
lambda.test(mytable, direction = 1)

## [1] 0.5857143
```

The option `direction` declares the independent variable. By specifying 1, the row variable is designated as the independent variable. By designating 2, the column variable is designated as the independent variable. Finally, by designating 0, both versions of λ are computed. Here, we have chosen the row variable as the independent variable, as we want to use continent to predict democracy. The result shows that knowing a country's continent improves the prediction of its democratic status by 58.6 percent relative to not knowing this information. This suggests a strong relationship between democracy and continent. This is the proportional reduction in error (PRE).

If we had wanted to predict a country's continent from whether or not it is a democracy, then we would have issued the following syntax:

```
library(rapport)
lambda.test(mytable, direction = 2)

## [1] 0.2272727
```

This result is a bit weaker, with the PRE being only 0.227.

6.5 Measures of Association for Ordinal Variables

6.5.1 Goodman and Kruskal's γ

Imagine we are interested in the relationship between human development and democracy. We employ the categorization of the HDI proposed by the United Nations Development Programme (UNDP) and the categorization of the democracy index used by the Economist Intelligence Unit, which turns both variables into ordinal scales. As we discussed in Chapter 3, the UNDP distinguishes between low, medium, high, and very high human development. We can generate these categories as follows.

```
world$f.hdi <- cut(world$hdi,
  breaks=c(-Inf, 0.55, 0.70, 0.80, Inf), right = TRUE,
  labels = c("Low Dev.", "Medium Dev.",
             "High Human Dev.", "Very High Human Dev."))
```

The Economist Intelligence Unit distinguishes between full democracies, flawed democracies, hybrid regimes, and authoritarian regimes, using cut-offs on `demindx` of 8, 6, and 4, respectively. We can generate these categories using the following syntax.

```
world$f.demindx <- cut(world$demindx,
  breaks=c(-Inf, 4, 6, 8, Inf), right = FALSE,
  labels = c("Authoritarian", "Hybrid",
             "Flawed Dem.", "Full Dem."))
```

The contingency table of the two variables can be obtained in the usual manner.

```
mytable <- table(world$f.hdi, world$f.demindx)
mytable
```

```
##
##                         Authoritarian Hybrid Flawed Dem.
##   Low Dev.                         24     14           4
##   Medium Dev.                      10     11          16
##   High Human Dev.                  10     13          14
##   Very High Human Dev.              6      1          17
##
##                         Full Dem.
##   Low Dev.                      0
##   Medium Dev.                   0
##   High Human Dev.               3
##   Very High Human Dev.         22
```

We now proceed to compute Goodman and Kruskal's γ, in order to measure the association between the two variables. For this, we use the `PResiduals` library, which contains the `GKGamma` function.[6]

```
library(PResiduals)
GKGamma(mytable)
```

```
## $scon
## [1] 6367
##
## $sdis
## [1] 1472
##
## $gamma
## [1] 0.6244419
```

[6]The libraries `MESS` and `vcdExtra` also contain functions for computing Goodman and Kruskal's γ. These are good when the goal is to engage in hypothesis testing. The advantage of the current function is that it shows the concordant and discordant pairs automatically.

We observe that γ is equal to 0.624 for the data at hand. This suggests a strong positive correlation between human development and the level of democracy.

6.5.2 Spearmans's ρ

With ordinal dependent variables, one can also calculate Spearman's ρ, the rank correlation coefficient proposed by Spearman. Returning to the original measurements of human development and the democracy index. Doing so is extremely easy, except that we have to use a slightly different syntax to remove missing values.

```
cor(world$hdi, world$demindx, use = "complete.obs",
    method = "spearman")

## [1] 0.6432357
```

Here, `use = "complete.obs"` means that we only use those country cases for which we have data on the democracy index and the HDI. We see that Spearman's ρ is 0.643, which indicates a strong positive association between the level of human development and the level of democracy.

6.6 Conclusions

R offers a number of very nice features for analyzing contingency tables. In the exercises you will be able to practice those features yourself. Remember that tabular analysis is often a useful starting point for analyzing relationships between variables. So you should feel free to tabulate early and often.

6.7 Exercises

(1) Consider the world economic, political, and social indicator data. Create a variable containing the quintiles on political stability. You could label these as "very low," "low," "moderate," "high," and "very high." Then create a twoway table with continent as the column variable and the stability quintiles as the row variable. The table should display the frequencies, as well as the column proportions. Interpret the results.

(2) Replicate the analysis from Section 6.2.2, now substituting a dichotomous version of corruption in lieu of the dichotomous version of political stability. Interpret the results.

(3) Continuing with the dichotomous corruption measure, let us turn this into the outcome variable and let us designate the dichotomous version of HDI as the risk factor. What is the relative risk of low-HDI countries scoring high

on corruption compared to high-HDI countries? How would you interpret this result?

(4) What is odds ratio for high versus low corruption when we compare high-HDI to low-HDI countries? How would you interpret this result?

(5) Compute and interpret Cramér's V for the twoway table between high and low HDI and high and low corruption.

(6) Compute and interpret Goodman and Kruskal's γ for a table that uses the 4-category HDI measure of the United Nations Development Programme and the democracy indicator derived from Polity.

(7) Take the 4-category HDI measure and tabulate it against the political stability quintiles generated in question (1). Now compute and interpret Goodman and Kruska's γ.

(8) Compute and interpret Spearman's ρ for HDI by corruption.

Chapter 7

Correlation and Regression

In this chapter, we shall take a look at association for interval and ratio scales. We begin by discussing scatter plots as a tool for visualizing relationships between variables. We then move onto a discussion of covariance and correlation. We conclude by discussing simple regression analysis in R.

7.1 Required Packages

For the exercises in this chapter, you will need to install and activate the following library:

- `GGally`
- `ggExtra`
- `ggplot2`

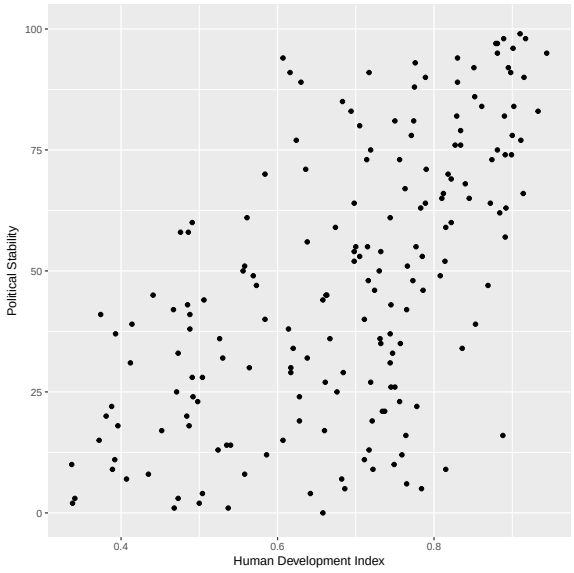
7.2 Scatter Plots

One of the best ways to explore the association between interval/ratio variables is to create a scatter plot. In its most basic form, a scatter plot shows the combined values of two variables in a Cartesian coordinate system. We can add various bells and whistles to the basic scatter plot, as we shall show in this section. Later in the chapter, we shall show as well how one can add information about correlation or a regression line to the scatter plot.

7.2.1 Basic Scatter Plots

Let us consider the relationship between the human development index (HDI) and political stability. We can visualize this relationship by creating a scatter plot using `ggplot2`.

Figure 7.1: Scatter Plot of Human Development and Political Stability



Note: Each point represents the value on HDI and political stability for a particular country.

```
library(foreign)
library(ggplot2)
world <- read.dta("world indicators.dta")
p1 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  xlab("Human Development Index") + ylab("Political Stability")
```

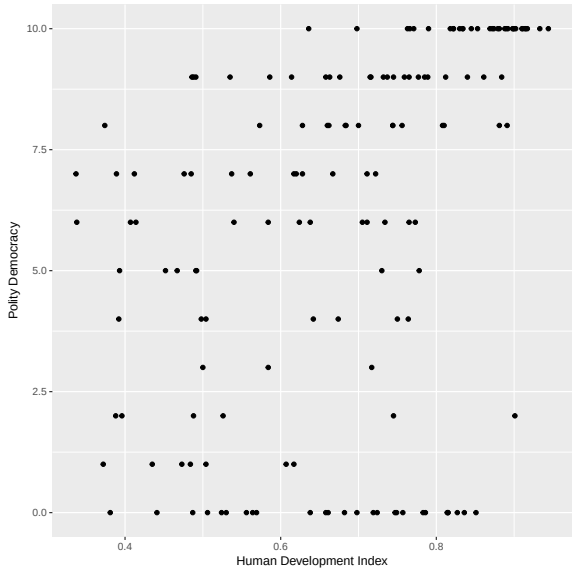
This produces Figure 7.1. We observe that low values of HDI tend to go together with low values of political stability, and that high values of HDI tend to occur with high values of stability. This is indicative of a positive relationship, although it is far from perfect. The last conclusion follows from the fact that the points do not neatly fall on a straight line but, rather, are scattered about such an, as of yet imaginary, line.

7.2.2 Jittered Scatter Plots

Now let us use a similar procedure to plot HDI against `politydem`, the Polity democracy score. The result is shown in Figure 7.2.[1] This graph, however, does not look as nice as Figure 7.1. The reason is that there are only 11 unique

[1] We do not repeat the syntax, since it is essentially the same as that used for Figure 7.1.

Figure 7.2: Scatter Plot of Human Development and the Polity Democracy Score



Note: This graph suffers from over-plotting.

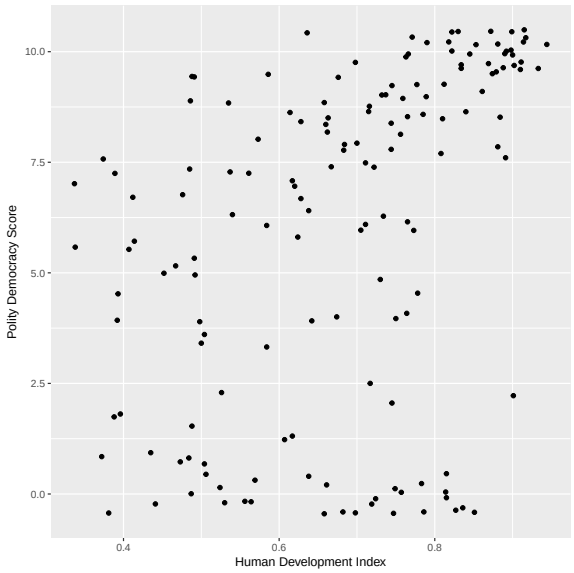
democracy scores, causing the points on the vertical axis to be aligned in a peculiar way and producing a problem of over-plotting. Over-plotting arises when dots are put (nearly) on top of each other, so that it becomes difficult to tell how many cases (e.g., countries) are situated at a specific location.

The solution is to add a small amount of random noise—so-called jitter—to each unit, so that the points can be more easily distinguished. One can add jitter along the horizontal axis, the vertical axis, or both. In our case, the problem is limited to the vertical axis, and we shall add jitter only along this dimension.

```
library(ggplot2)
p3 <- ggplot(world, aes(x = hdi, y = politydem)) +
  geom_point(position = position_jitter(w = 0, h = .5)) +
  xlab("Human Development Index") +
  ylab("Polity Democracy Score")
```

The result is shown in Figure 7.3. Before interpreting the plot, we should comment briefly on the syntax. The `position_jitter` option allows us to specify the nature and the degree of jittering. It has two arguments: width (w) and height (h). Width influences the degree of jittering along the horizontal axis and

Figure 7.3: Jittered Scatter Plot of Human Development and the Polity Democracy Score



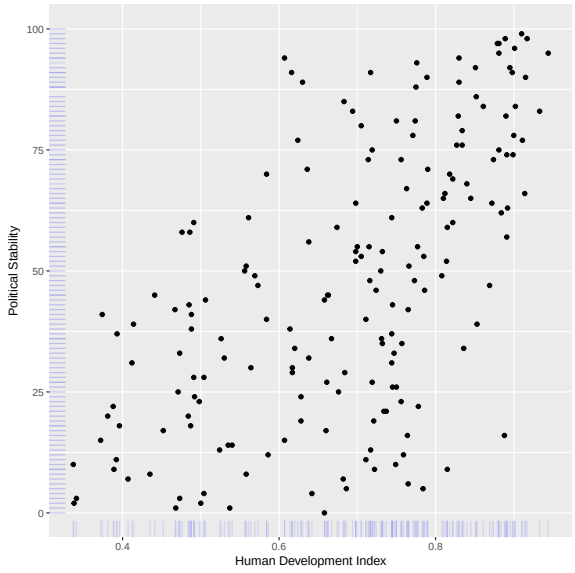
Note: Jittering has been applied to the vertical axis of the plot.

has been set to 0 here. Height influences the degree of jittering along the vertical axis and has been set to .5. Both height and width are stated as a proportion of the resolution of the plot, which roughly corresponds to the smallest distance between any two elements in a vector. Both the horizontal and vertical axis correspond to vectors of values of a variable and the smallest distance between any two units on a given vector is taken to determine the amount of jittering that is applied. The default is 40 percent on both axes. The effect of jittering is that Figure 7.3 now looks a lot more like Figure 7.1 than did Figure 7.2. In substantive terms, it appears that there is something of a positive relationship between human development and democracy. However, the relationship does not appear to be particularly strong due to the considerable scatter we observe in Figure 7.3.

7.2.3 Adding Marginal Distributions

Useful as (jittered) scatter plots are, it would be nice to add information about the marginal distributions of the variables in the plot. There are several ways of doing this. First, one can apply so-called rugging. A rug is a set of tickmarks that appear on the axis of a plot to indicate the distribution of a variable. We illustrate it here for the earlier scatter plot of human development and political

Figure 7.4: Rugged Scatter Plot of Human Development and Political Stability



Note: The tickmarks at the margins reflect the distribution of the human development index and political stability.

stability.

```
library(ggplot2)
p4 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  geom_rug(col = "blue", alpha = .1) +
  xlab("Human Development Index") +
  ylab("Political Stability")
```

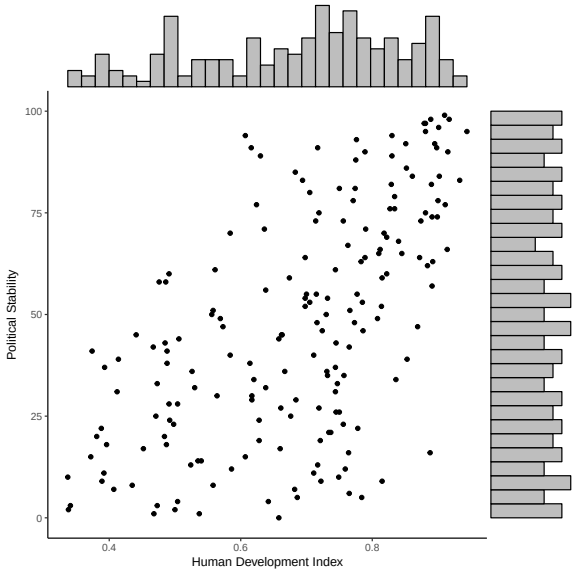
The result is shown in Figure 7.4.[2]

An alternative approach is to provide complete histograms of the marginal distributions. For this, we like to use the `ggExtra` package.

```
library(ggExtra)
p <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  xlab("Human Development Index") +
```

[2]The parameter `alpha` controls the transparency of the rug. It is set to a low value so as not to distract from the main act on the plot, which is the scatter itself.

Figure 7.5: Rugged Scatter Plot of Human Development and Political Stability with Histograms



Note: The histograms in the margins show the distributions of human development and political stability.

```
  ylab("Political Stability") +
  theme_classic()
p5 <- suppressWarnings(ggExtra::ggMarginal(p, type = "histogram"))
```

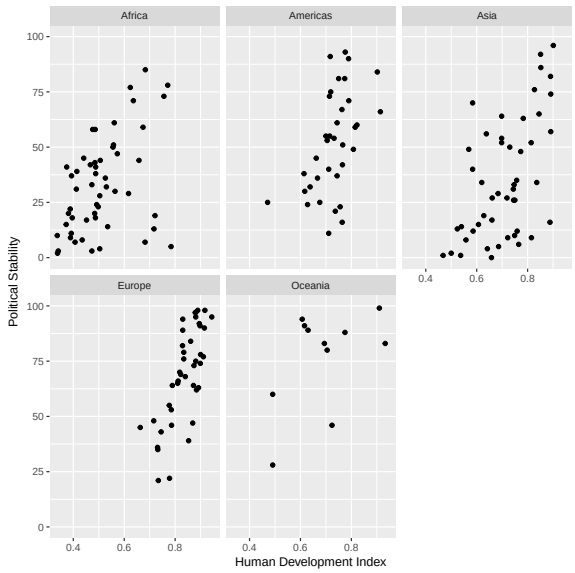
The result is shown in Figure 7.5.[3] By specifying `type = "density"`, you obtain density plots in the margins in lieu of histograms. Parenthetically, a similar kind of display can be obtained using the `GGally` library, which we shall illustrate later in this chapter.

7.2.4 Facetted Scatter Plots

As we did with density plots, it is possible to apply facetting in a scatter plot. We illustrate this here for the relationship between human development and political stability. The facetting that we apply is at the level of the continents. If we want to create separate graphs for each continent.

[3]The inclusion of `theme_classic` means that the background is white and the gridlines have been removed. This argument is optional.

Figure 7.6: Facetted Scatter Plot of Human Development and Political Stability with Histograms



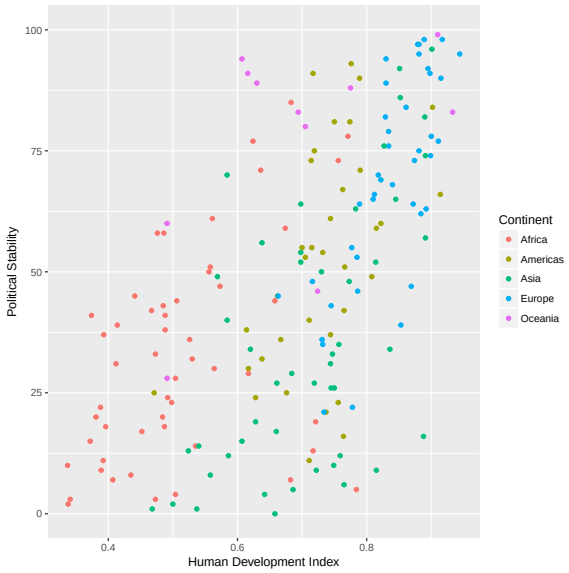
Note: Facetting is done by continent.

```
p6 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  facet_wrap(~continent) +
  xlab("Human Development Index") +
  ylab("Political Stability")
```

We see that we have again used the `facet_wrap` option, which we introduced in Chapter 4. The result is shown in Figure 7.6. From a substantive vantage point, we see that the relationship between human development and political stability is positive on every continent. However, the degree of scatter varies greatly. The clustering of points is quite tight in Europe and considerably less so in Africa, which suggests a much stronger relationship in Europe than in Africa. The other continents are situated in between these extremes.

We strongly prefer facetting to demonstrate group differences such as those between continents. However, it is also possible to show differences of this nature using color coding within a single scatter plot. This can be accomplished using the following syntax:

Figure 7.7: Scatter Plot of Human Development and Political Stability with Histograms Using Different Colors for Each Continent



Note: Colors correspond to continents.

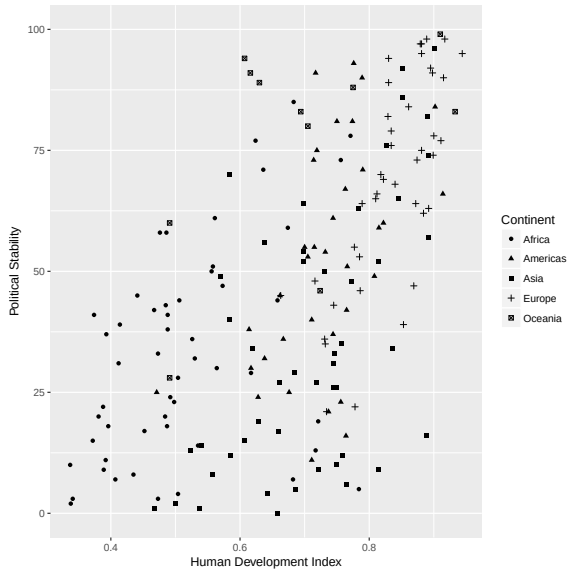
```
p7 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point(aes(color = factor(continent))) +
  xlab("Human Development Index") +
  ylab("Political Stability") +
  scale_color_discrete(name = "Continent")
```

The result is shown in Figure 7.7. Although the colors make for an artistic scatter plot, I hope you will agree that its much harder to read than Figure 7.6. The same is true of a variant that uses different shapes for different continents:

```
p8 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point(aes(shape = factor(continent))) +
  xlab("Human Development Index") +
  ylab("Political Stability") +
  scale_shape_discrete(name = "Continent")
```

This rendition is shown in Figure 7.8.

Figure 7.8: Scatter Plot of Human Development and Political Stability with Histograms Using Different Shapes for Each Continent



Note: Shapes correspond to continents.

7.3 Measures of Association

7.3.1 Covariance

To obtain the covariance between two interval/ratio scales, we can utilize the `cov` function. If we want to remove missing values, then the syntax is:

```
cov(world$hdi, world$polstab, use = "complete.obs")

## [1] 2.651434
```

This gives the covariance between the human development index and political stability. The argument `complete.obs` means that we only use those units for which we have data on both variables. We see that the covariance is positive, suggesting a positive linear association, just as we discussed when considering Figure 7.1. This means that increases in the human development index tend to be associated with increases in political stability.

The `cov` function can be expanded to consider more than two variables. In this case, it will produce a matrix of all pairwise covariances. For example,

```
world.sub <- select(world, hdi, polstab, politydem)
S <- cov(world.sub, use = "complete.obs")
S

##                  hdi    polstab  politydem
## hdi       0.02580682   2.529529  0.2404437
## polstab   2.52952851 667.051084 41.5083849
## politydem 0.24044372  41.508385 13.8992088
```

generates a 3×3 matrix (see below) of covariances between the human development index, political stability, and the Polity democracy score. Those variables have first been selected from the original data frame `world` and placed into their own data frame `world.sub` (see Chapter 3).

A matrix is a rectangular array consisting of rows and columns. In our case, we have three rows and columns, corresponding to the number of variables we are looking at. Where a row and a column meet, we have a cell. In our case, the cell contents are numbers corresponding to covariances. In all, we have $3 \times 3 = 9$ cells/covariances. Their meaning is as follows.

- The cell in the first row and first column contains the value 0.026 (approximately). This is the covariance of the human development index with itself, which is the same as the variance of the human development index. (You can verify this by evaluating the `var` function.)

- The cell in the second row and the first column contains the value 2.53 (approximately). This is the covariance between the human development index and political stability. Note that this is the same value as that

Table 7.1: Structure of a Covariance Matrix with K Variables

	X_1	X_2	X_3	$\cdots$	X_K
X_1	$\text{Var}(X_1)$	$\text{Cov}(X_1, X_2)$	$\text{Cov}(X_1, X_3)$	$\cdots$	$\text{Cov}(X_1, X_K)$
X_2	$\text{Cov}(X_1, X_2)$	$\text{Var}(X_2)$	$\text{Cov}(X_2, X_3)$	$\cdots$	$\text{Cov}(X_2, X_K)$
X_3	$\text{Cov}(X_3, X_1)$	$\text{Cov}(X_2, X_3)$	$\text{Var}(X_3)$	$\cdots$	$\text{Cov}(X_3, X_K)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
X_K	$\text{Cov}(X_K, X_1)$	$\text{Cov}(X_K, X_2)$	$\text{Cov}(X_K, X_3)$	$\cdots$	$\text{Var}(X_K)$

found in the first row and second column. We observe that the covariance is positive, as we already saw.

- The cell in the second row and the second column contains the value 667.051n(approximately). This is the covariance of political stability with itself, which is the same as the variance of political stability. We see that the relationship between the two variables is positive.

- The cell in the third row and the first column contains the value 0.24 (approximately), which is the covariance between the Polity democracy score and the human development index. This is the same value as that found in the first row and third column. It suggests a positive relationship between the two variables, such that high values of human development tend to go together with high values of democracy (and vice versa).

- The cell in the third row and the second column contains the value 41.508 (approximately). This is the covariance between political stability and the Polity democracy score. This is the same value as that found in the second row and third column. Since the covariance is positive, we know that high democracy scores tend to occur along with political stability (and vice versa).

- The cell in the third row and third column contains the value 13.899 (approximately). This is the covariance of the Polity democracy score with itself, i.e., the variance.

In general, covariance matrices produced by `cov` take the form shown in Table 7.1. We see that the diagonal elements contain variances and that the off-diagonal elements are covariances.

In the previous analysis, we asked the covariances to be computed on complete cases only. In the context of an analysis of three variables, this means that a country cannot have missing values on any of the three variables; we call this **listwise deletion**. Thus, when we compute the covariance between the human development index and political stability, there cannot be missingness on those variables but—and this is important—also not on the Polity democracy score. This is the default approach in political science, as it ensures that the same number of observations is used throughout.

It is possible, however, to relax this requirement. Specifically, we can issue the following syntax:

```
cov(world.sub, use = "pairwise.complete.obs")

##                  hdi    polstab  politydem
## hdi       0.02435892   2.651434  0.2404437
## polstab   2.65143369 850.104446 41.7455087
## politydem 0.24044372  41.745509 13.9076433
```

Now, the covariance between the human development index and political stability is based on cases that have no missing values on those variables. However, it does not matter if those cases are missing on the Polity democracy score. We call this **pairwise deletion**. Missingness on that variable becomes relevant only if we want to obtain its variance or covariance with other variables. The method has two major drawbacks. First, in R it is extremely difficult to uncover the sample size on which each variance/covariance is based. Second, and more important, when covariances are based on different samples, it becomes dangerous to compare them. If the goal is to compare different covariances, we recommend always that you use the `complete.obs` option.

7.3.2 Correlation

To compute the Pearson product-moment correlation, one can proceed in nearly the same way as with the covariance. For example, to compute the correlation between the human development index and political stability, the following syntax suffices.

```
cor(world$hdi, world$polstab, use = "complete.obs")

## [1] 0.601534
```

This is the same syntax that we used to compute Spearman's ρ in the previous chapter. Only there, we specified `method = "spearman"`, whereas we do not bother to specify a method here. The reason is that the `cor` function computes the Pearson product-moment correlation by default; one is required to use the `method` option only if one wishes to compute another type of correlation.

Just as with covariances, we can compute a correlation matrix of many variables. For example,

```
R <- cor(world.sub, use = "complete.obs")
R

##                 hdi   polstab politydem
## hdi       1.0000000 0.6096672 0.4014682
## polstab   0.6096672 1.0000000 0.4310833
## politydem 0.4014682 0.4310833 1.0000000
```

gives the correlation matrix for human development, political stability, and democracy using listwise deletion. The correlation matrix under pairwise deletion is given by

```
R <- cor(world.sub, use = "pairwise.complete.obs")
R

##                 hdi   polstab politydem
## hdi       1.0000000 0.6015340 0.4014682
## polstab   0.6015340 1.0000000 0.4313486
## politydem 0.4014682 0.4313486 1.0000000
```

As before, this is a 3×3 matrix. The interpretation of the elements of this matrix is as follows.

- The element that makes up the first row and first column is the correlation of the Human Development Index with itself, which is by definition 1.
- The element that makes up the second row and the first column is the correlation of the Human Development Index with political stability. This correlation is 0.602 and can be considered a strong positive correlation. The same correlation can be found in the first row and second column of the correlation matrix.
- The element that makes up the second row and the second column is the correlation of political stability with itself, which is by definition 1.
- The element that makes up the third row and the first column is the correlation between the Human Development Index and the Polity democracy score. This correlation is 0.401 and can be considered a moderate positive correlation. The same correlation can be found in the first row and the third column of the correlation matrix.
- The element that makes up the third row and the second column is the correlation between political stability and the Polity democracy score. This correlation is 0.431 and can again be considered a moderate positive correlation. The same correlation can be found in the second row and the third column of the correlation matrix.
- The element that makes up the third row and the third column is the correlation of the Polity democracy score with itself. As always, this correlation is 1 by definition.

In general, we see that the correlation matrix has a structure very similar to Table 7.1, with the peculiarity that the diagonal elements are all 1 and that the off-diagonal elements represent correlations instead of covariances.

7.3.3 Showing Correlations in a Scatter Plot

It would be nice to show the correlation along with a scatter plot of the data. One of the easiest ways to do this is to use the newly developed `GGally` library. For example, to create Figure 7.9, the following syntax can be used.

```
library(dplyr)
library(GGally)

##
## Attaching package:  'GGally'
## The following object is masked from 'package:dplyr':
##
##    nasa

world.sub <- select(world, hdi, polstab)
lbl <- c("HDI", "Political Stability")
p1 <- ggpairs(world.sub, columns = 1:2, columnLabels = lbl)
```

This figure clearly shows that the correlation between the human development index and political stability. In addition, it provides the scatter plot between those variables, as well as their densities. Hence, a plot of this nature provides a great deal of information.

The `ggpairs` function does not limit us to creating plots with just two variables. If we issue the following syntax

```
library(dplyr)
library(GGally)
world.sub <- select(world, hdi, polstab, politydem)
lbl <- c("HDI", "Political Stability", "Democracy")
p2 <- ggpairs(world.sub, columns = 1:3, columnLabels = lbl)
```

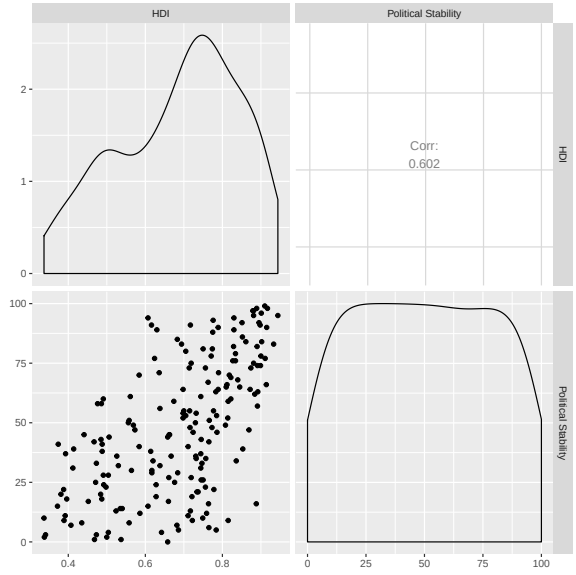
then we obtain Figure 7.10. This kind of figure is known as a **matrix scatter plot** and is an extremely useful tool for exploring relationships between variables that theoretically should hang together.

7.4 Simple Regression

7.4.1 Fitting a Simple Regression Model

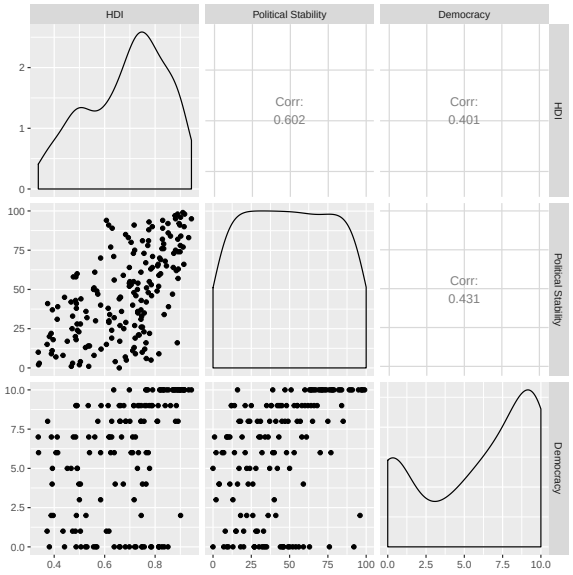
Imagine that we are interested in predicting the level of political stability in a country based on the level of human development in that country. Since political stability is measured on a ratio scale, we can perform a regression analysis. This amounts to obtaining the line that best fits the scatter plot between the two variables. Fitting a regression is extremely easy in R:

Figure 7.9: Human Development and Political Stability



Note: The upper-left quadrant shows the density of the human development index (HDI). The upper-right quadrant shows the correlation between the HDI and political stability. The lower-left quadrant gives the scatter plot of the HDI and political stability. The lower-right quadrant shows the density of political stability.

Figure 7.10: Human Development, Political Stability, and Democracy around the World



Note: The plot contains the following elements: (1) top row, left is the density of the human development index (HDI); (2) top row, center is the correlation between the HDI and political stability; (3) top row, right is the correlation between the HDI and democracy; (4) middle row, left is the scatter plot between the HDI and political stability; (5) middle row, center is the density if political stability; (6) middle row, right is the correlation between political stability and democracy; (7) bottom row, left is the scatter plot between the HDI and democracy; (8) bottom row, center is the scatter plot between political stability and democracy; and (9) bottom row, right is the density for democracy.

```
fit <- lm(polstab ~ hdi, world)
summary(fit)

##
## Call:
## lm(formula = polstab ~ hdi, data = world)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.429 -16.547   1.539  17.073  55.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -27.229      7.472  -3.644 0.000349 ***
## hdi          108.849     10.628  10.242  < 2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.62 on 185 degrees of freedom
##   (33 observations deleted due to missingness)
## Multiple R-squared:  0.3618,Adjusted R-squared:  0.3584
## F-statistic: 104.9 on 1 and 185 DF,  p-value: < 2.2e-16
```

Here `lm` stands for linear model and the dependent and independent variable are separated via a tilde.

Many of the results shown in the output are still too advanced to discuss at this stage. However, we are equipped to interpret two elements of the output: (1) the column labeled "Estimate," and (2) the "multiple R-squared." The estimate labeled "(Intercept)" is -27.229; this is the intercept of the linear regression equation. The estimate labeled "hdi" is 108.849; this is the slope of the linear regression equation. The regression equation may thus be written as

$$\widehat{\text{Stable}} \quad = \quad -27.229 + 108.849 \times \text{HDI}$$

This means that a unit increase in the human development index is expected to increase political stability by 108.849 units. You should keep in mind—this is where descriptive statistics are invaluable—that the HDI has an empirical range that falls well short of a full unit. Perhaps, then, it is better to look at the effect of an increase of a tenth of a unit in human development. This is expected to increase political stability by one tenth times 108.849, or 10.885. Given that the political stability scale ranges between 0 and 100, this can be considered a sizable effect.

The intercept of the regression equation is -27.229. This is the expected level of political stability when the human development index is 0. It is not particularly useful to offer this interpretation, however, because the human development index is nowhere 0.

Turning to the multiple R-squared, we see that it is approximately equal to 0.362. This means that human development explains about 36.2 percent of the variance in political stability. This is quite reasonable, given that we know political stability to depend on a lot more than human development alone.

7.4.2 Graphing the Regression Line

It is possible to visualize the regression line in a scatter plot using `ggplot2`. The syntax is as follows.

```
library(ggplot2)
p1 <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, size = 1.5) +
  xlab("HDI") + ylab("Political Stability")
```

The result is shown in Figure 7.11. The key function that was used to create the regression line is `geom_smooth`, which receives three arguments. First, we set the method to `lm`, which we encountered before. Second, `se` is set to `FALSE` so that only the regression line is being displayed. Finally, `size = 1.5` draws the regression line at a width of 1.5 times the standard line size, so that it stands out a bit more.

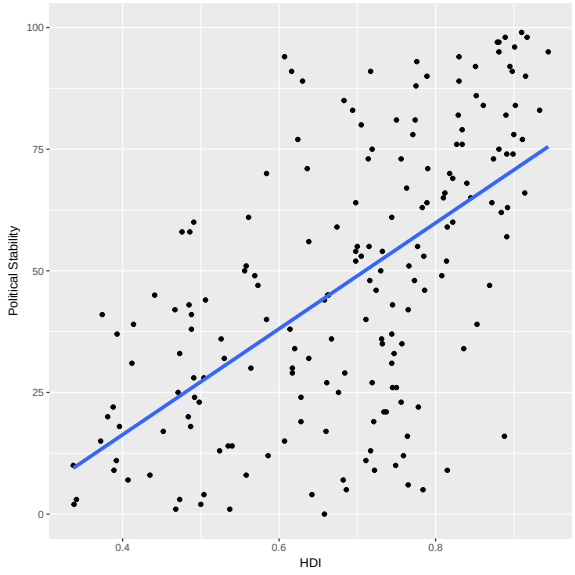
We can add some annotation to the scatter plot, so that the equation for the regression line and the R-squared are depicted as well. This requires that you write the following function:

```
lm_eqn = function(world) {
  m = lm(polstab ~ hdi, world);
  eq <- substitute(italic(Stability) == a + b %.%
                     italic(HDI)*","~~
                     italic(R)^2~"="~r2,
                   list(a = format(coef(m)[1], digits = 3),
                        b = format(coef(m)[2], digits = 4),
                        r2 = format(summary(m)$r.squared,
                                    digits = 3)))
  as.character(as.expression(eq));
}
```

In a next step, we now create the plot:

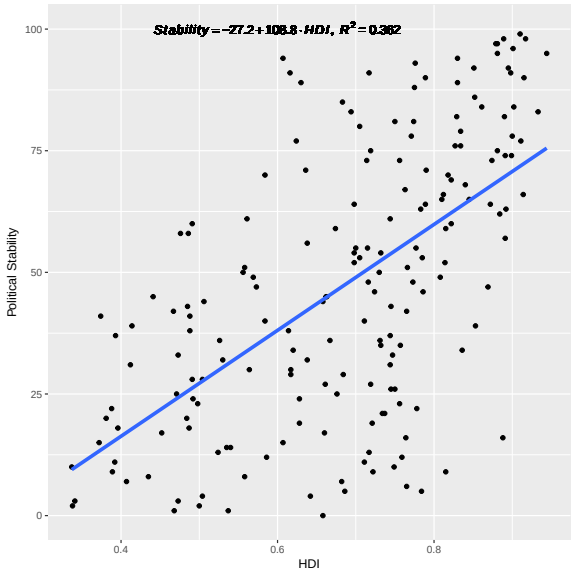
```
library(ggplot2)
p <- ggplot(world, aes(x = hdi, y = polstab)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, size = 1.5) +
  xlab("HDI") + ylab("Political Stability")
p2 <- p + geom_text(x = 0.60, y = 100, label =
```

Figure 7.11: Regressing Political Stability onto Human Development



Note: The blue line is the linear regression line.

Figure 7.12: Regressing Political Stability onto Human Development Once More



Note: The blue line is the linear regression line. The text shows the regression equation and multiple R-squared.

```
lm_eqn(world), parse = TRUE)
```

The resulting graph is shown in Figure 7.12.

7.5 Exercises

(1) Consider the data in `studidata.Rda`. This data set contains the variables `height` and `weight`, which capture students' self-reported heights and weights, respectively. Construct a scatter plot of the two variables. Comment on whether the pattern reflects weak or strong association and in which direction this goes. If there is a relationship, does it appear to be linear?

(2) Redo the scatter plot from problem (1), now facetting on a student's sex (contained in the variable `man`). Are there any obvious differences in the relationship between height and weight between men and women?

(3) Compute the covariance and correlation between height and weight. Interpret the results.

(4) Repeat the computation of the correlation separately for men and women. Does it look like the relationship between the two variables differs between men and women?

(5) Run a regression of weight on height. Interpret the slope coefficient.

(6) Does it make sense to interpret the intercept of the regression you just ran? Why does it (or not) make sense?

(7) How much variance does height explain? Would you consider this a lot or a little?

(8) Run separate regressions for men and women. What differences do you observe?

Part III

Probability Theory

Chapter 8

Working With Probability Distributions

Probabilities are the glue that connect statistical inference and descriptive statistics. As such, it is of great importance that one learns how to work with them. In this chapter, we shall see how R can be enticed to serve as a probability calculator and random number generator. We start by illustrating this for the uniform distribution. Next, we introduce a number of useful statistical distributions and their relevant R syntax.

8.1 Required Packages

For this chapter, you need to download and install two packages:

- `actuar`
- `moments`

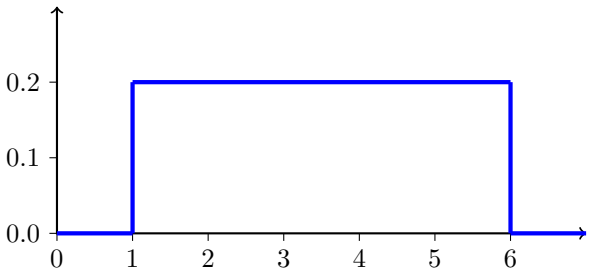
8.2 The Uniform Distribution

8.2.1 The Continuous Uniform Distribution

Imagine that the random variable X is distributed normally over the interval $[1, 6]$; we write $X \sim \mathcal{U}(1, 6)$. Figure 8.1 shows the probability density function (PDF) of the random variable. The question before us is how one can tell R to work with this distribution.

Declaring the Probability Density Function To declare the PDF in R, one first has to specify its support. Recall that the support is that range of values of the random variable for which the PDF is non-zero. In our case, this is the range between 1 and 6. We can specify this using

Figure 8.1: Uniform Probability Density Function



Note: The graph shows the continuous uniform distribution for a random variable that takes on values between 1 and 6.

```
x <- seq(1, 6, length = 100)
```

This creates a sequence of 100 numbers between 1 and 100. To obtain the PDF, we now use the `dunif` function:

```
f <- dunif(x, min = 1, max = 6)
f
```

```
##   [1] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [14] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [27] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [40] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [53] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [66] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [79] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
##  [92] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
```

We see that at each value of x, the PDF takes on the value .2, just as we saw in Figure 8.1. Keep in mind that, since X is continuous, the values of the PDF may not be interpreted as probabilities.

Cumulative Distribution Function To establish the probability that X takes on values between a and b, it is easiest to work with the cumulative distribution function (CDF). For the continuous uniform distribution, cumulative probabilities can be obtained using the `punif` function. For example, if we want to know $\Pr(1 \leq x \leq 2)$, then the following syntax suffices:

```
punif(2, min = 1, max = 6)
```

```
## [1] 0.2
```

The probability is .2, just as it should be:

$$\begin{aligned} \Pr(1 \leq x \leq 2) &= \int_1^2 f(x)dx \\ &= \underbrace{1}_{\text{base}} \cdot \underbrace{.2}_{\text{height}} \\ &= .2 \end{aligned}$$

Here, the base times height principle works for the computation of the integral because the distribution is rectangula (see Figure 8.1).

We can use the `punif` function to compute any probability. Imagine, for instance, that we want to ascertain the probability that the random variable takes on values between 3 and 5. Figure 8.2 shows the principle of how this is done. We begin by evaluating the cumulative probability at 5. This gives the probability of scoring 5 or less, which in our setup is identical to the probability that the random variable takes on values between 1 and 5. Clearly, this is not the probability we are looking for. However, when we subtract the cumulative probability at 3, then we do obtain $\Pr(3 \leq x \leq 5)$. In R this means that we issue the following command:

```
punif(5, min = 1, max = 6) - punif(3, min = 1, max = 6)

## [1] 0.4
```

It is easily verified this is the correct answer:

$$\begin{aligned} \Pr(3 \leq x \leq 5) &= \int_3^5 f(x)dx \\ &= \underbrace{(5-3)}_{\text{base}} \cdot \underbrace{.2}_{\text{height}} \\ &= .4 \end{aligned}$$

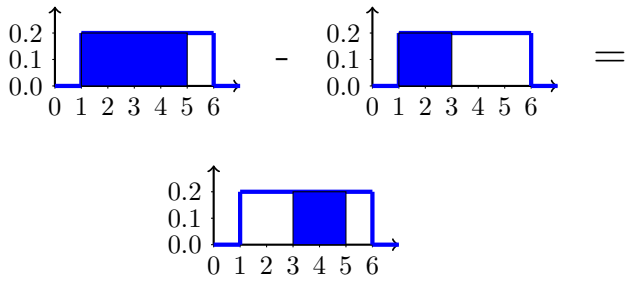
The `punif` function allows us to take some value of a random variable and determine the cumulative probability. We can also reverse the process by taking a cumulative probability and asking to what value of the random variable this corresponds. For this we use the `qunif` function. Imagine that we want to know for what value of X the cumulative probability is .8. We can now issue the following syntax.

```
qunif(.8, min = 1, max = 6)

## [1] 5
```

Thus, $F(X = 5) = .8$.

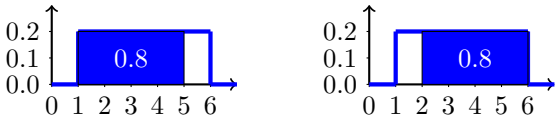
The `qunif` command is a bit more flexible than we just demonstrated because we can add the option `lower.tail`. Compare the following two commands:

Figure 8.2: Using Cumulative Probabilities to Obtain Other Probabilities



Note: $\Pr(3 \leq x \leq 5) = \Pr(1 \leq x \leq 5) - \Pr(1 \leq x \leq 3)$.

Figure 8.3: Different Variants of the `qunif` Function



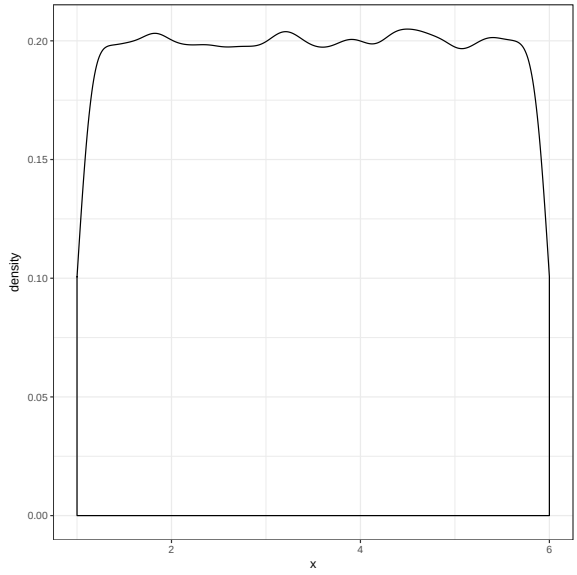
Note: The left panel takes the option `lower.tail = TRUE` and finds a value x such that $\Pr(X \leq x) = 0.8$. The right panel takes the option `lower.tail = FALSE` and finds a value x such that $\Pr(X \geq x) = 0.8$.

```
qunif(.8, min = 1, max = 6, lower.tail = TRUE)

## [1] 5

qunif(.8, min = 1, max = 6, lower.tail = FALSE)

## [1] 2
```

The first command asks for a value x such that $\Pr(X \leq x) = .8$. The second command asks for a value of x such that $\Pr(X > x) = .8$. The former corresponds to evaluating the cumulative distribution function, $F(x)$, whereas the latter corresponds to evaluating the survival function, $S(x)$. In general, the survival function is a function $S(x) = 1 - F(x)$. The differences are shown in Figure 8.3.

Generating Random Numbers On occasion, we may wish to sample from a particular distribution. This is helpful, for example, when we want to generate some artificial data to simulate the behavior of some phenomenon. In the case of the continuous uniform distribution, the corresponding command is

Figure 8.4: Random Numbers from a Continuous Uniform Distribution



Note: 100'000 draws generated with `runif`.

```
set.seed(1234)
x <- runif(100000, min = 1, max = 6)
```

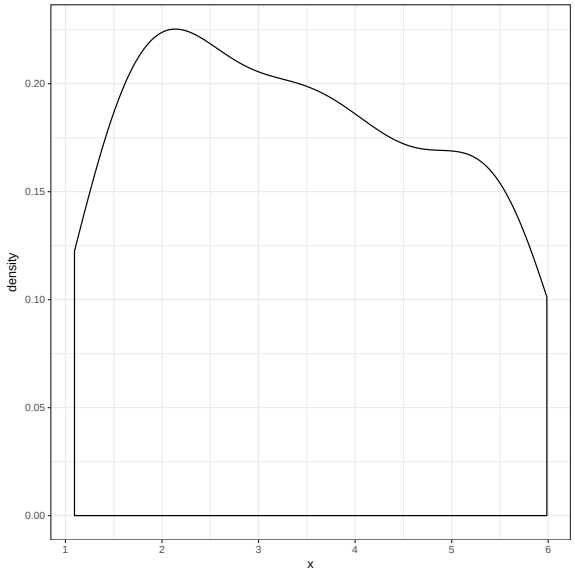
A graphical display of the distribution is shown in Figure 8.4. This reveals that the distribution is more or less uniform over the interval $[1, 6]$.

Getting into the syntax, the first line sets the seed for the random number generator. This is something that a random number generator needs to start the task of producing (pseudo-) random numbers. If you want to ensure that you always generate the same set of random numbers, so that others can replicate your work, it is essential that you set the seed. The number 1234 is arbitrary, but anyone using it will get the same random numbers. When you omit the seed, then the random number generator will use the state of the computer system (e.g., clock time). Since this will be different at different times and for different users, the random numbers that are generated will not be the same.

On the second line, we specify the `runif` function. The first argument of this function is the sample size, n, which tells R how many random numbers should be generated. The remaining arguments are familiar; they specify the support of the continuous uniform distribution.

With random number generation, one principle is of the utmost importance: the larger the sample size, the more closely the set of random numbers approx-

Figure 8.5: The Effect of a Small n on Random Number Generators



Note: 100 draws generated with `runif`.

imates the desired distribution. In Figure 8.5, we reduce the sample size to $n = 100$. We see that the distribution does not look very uniform at all. Instead of a density that is flat over the support, we observe two peaks. This can happen when we generate a small sample and is a normal result of sampling fluctuation, a result that we shall discuss later in the course.

Naming Conventions We have now seen all of the built-in R commands for the continuous uniform distribution. Perhaps you are seeing a pattern to the naming of those functions. They always end with `unif` and the letter that is added as a prefix determines the result that the function produces. Specifically, the prefixes have the following effects:

- `d` results in the evaluation of the probability density function
- `p` yields the cumulative probability p associated with a value x of the random variable
- `q` yields the value x of the random variable that is associated with a cumulative probability of p
- `r` generates random numbers

The nice thing in R is that these same naming conventions apply to all distributions. You will never have to guess as to what a particular command like `dunif` does; the first letter says it all.

8.2.2 Discrete Uniform Distribution

R does not have built-in functions for all distributions. A case in point is the discrete uniform distribution. However, using the knowledge from Chapter 1, it is often quite easy to write your own equivalents of the `d`, `p`, `q`, and `r` functions. We illustrate this here for the discrete uniform probability mass function.

Probability Mass Function Imagine a discrete random variable that takes on values between 1 and k. To make things specific, let $k = 5$ so that $X = 1, 2, 3, 4, 5$. We declare the support using the following syntax:

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

The `seq` command creates a sequence of numbers. The starting point is indicated by `from`, which in this case is 1. The end point is indicated by `to`, which for us is k, which we have initialized as 5. The `by` command indicates the step size, i.e., the amount by which the numbers increase between the starting and end points. This is set here to 1, so that we go from 1 to 2, 2 to 3, etc. Had we specified 2 instead, then we would have obtained a sequence of the numbers 1, 3, and 5. By initialzing a different value of k, we can create any sequence of numbers between 1 and k.

The probability mass function for $X = 1, 2, \cdots, k$ is $f(x) = 1/k$. This is easily declared as a function:

```
ddu <- function(x, k) ifelse(x>=1 & x <=k, 1/k, 0)
```

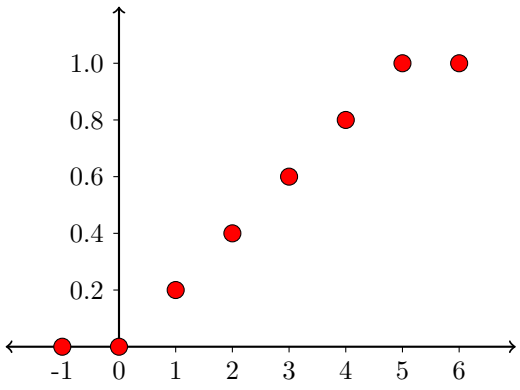
The function states that $f(x) = 0$ for values outside the interval $[1, k]$. Inside this interval, $f(x) = 1/k$. For example,

```
ddu(2, k)

## [1] 0.2
```

This function call computes $f(2)$, which is the probability that X takes on the value 2, in this case .2.

Cumulative Distribution Function The CDF is equal to $F(x) = x/k$ over the interval $x \in [1, k]$. Prior to $X = 1$, the CDF is 0; past k, it stays at a value of 1 (see Figure 8.6). We can program this in the following manner:

Figure 8.6: Discrete Uniform Cumulative Distribution Function



Note: $X = 1, 2, 3, 4, 5$.

```
pdu <- function(x, k) ifelse(x<1, 0, ifelse(x>=1 & x<=k, x/k, 1))
```

We can now apply this function to determine, for example, the cumulative probability at $X = 4$:

```
pdu(4, k)

## [1] 0.8
```

We can also write a function to determine the value of X that is associated with a particular cumulative probability.

```
qdu <- function(p,k) ifelse(p>=0 & p<=1, round(p*k), NA)
```

For example, the value of X for which the cumulative probability is .6 is given by

```
qdu(.6, k)

## [1] 3
```

Random Number Generation Finally, consider the idea of generating random numbers from a discrete uniform distribution. Here, we can use R's built-in sample function.

```
rdu <- function(n, k) sample(1:k, n, replace = TRUE)
```

We can now draw, for instance, 10 numbers from the distribution:

```
set.seed(8050)
rdu(10, 5)

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

8.3 Common Statistical Distributions

8.3.1 The Bernoulli Distribution

Description Consider a discrete random variable X that can take on values 0 and 1 with probabilities $1 - \pi$ and π, respectively. We say that X follows the Bernoulli distribution, which has one parameter, to wit π. We write this as $X \sim \mathcal{B}(\pi)$. The probability mass function is

$$f(x) \quad = \quad \pi^y \cdot (1 - \pi)^{1-y}$$

The Bernoulli distribution is of great relevance to political science, since many of our variables are dichotomous (e.g., peace versus war or vote versus abstain).

R R treats the Bernoulli distribution as a special case of the binomial distribution, which arises when the number of trials is equal to 1.

8.3.2 The Binomial Distribution

Description Consider a series of n independent Bernoulli trials, each with identical π. Each trial yields an outcome of 0 or 1, where 1 is often referred to as a "success." Now let X be a count of the number of 1s that are realized in n trials. Hence, $X = 0, 1, \cdots, n$. Then X follows the binomial distribution: $X \sim \mathcal{BN}(n, \pi)$. The probability mass function is given by

$$f(x) \quad = \quad \binom{n}{x} \pi^x \cdot (1 - \pi)^{n-x}$$

Here $\binom{n}{x}$ is the binomial coefficient, which captures the number of sequences of 0s and 1s that result in a value of x on the random variable.[1]

In the natural sciences, including genetics, the binomial distribution plays an important role. In the social sciences, the $\mathcal{BN}(1, \pi)$ distribution is highly

[1] The binomial coefficient evaluates to

$$\binom{n}{x} \quad = \quad \frac{n!}{x! \cdot (n - x)!}$$

Here $q! = q \cdot (q - 1) \cdot (q - 2) \cdots 2 \cdot 1$.

relevant, since it is identical to the Bernoulli distribution. For $n > 1$, the social scientific relevance is often less clear because the underlying assumptions are highly stringent. For example, one might be inclined to use the distribution to describe the number of yay votes in a legislature. Each legislator is either a yay (1) or a nay (0), and the count of 1s may be viewed as a measure of legislative support for a bill. While this would appear to be a perfect case for the binomial distribution, its validity is questionable. First, legislators act in political parties, and not independently from one another. Second, the probability of voting for the bill may vary greatly across legislators, depending on their ideological leanings.

R Imagine we are interested in obtaining the probability of 4 successes (1s) in 10 trials. Imagine the probability of success on a given trial is .4. Then the probability can be computed using

```
dbinom(4, 10, .4)

## [1] 0.2508227
```

Note that the first argument is the value x, the second argument is the number of trials, n, and the third argument is π.

Now lets consider a slightly different question: what is the probability of obtaining at most 4 successes. This can be computed as follows:

```
pbinom(4, 10, .4)

## [1] 0.6331033
```

We can, of course, also ask what value of X is associated with a cumulative probability of say 0.75:

```
qbinom(.75, 10, .4)

## [1] 5
```

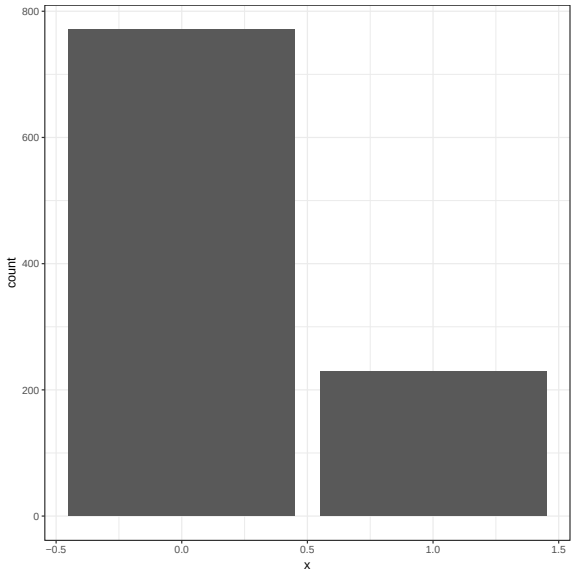
The `qbinom` contains the `lower.tail` option we encountered for the continuous uniform distribution.

Finally, imagine that we would like to generate a sample of 1000 draws from a Bernoulli distribution with $\pi = .2$. Then we issue

```
set.seed(1230)
x <- rbinom(1000, 1, .2)
```

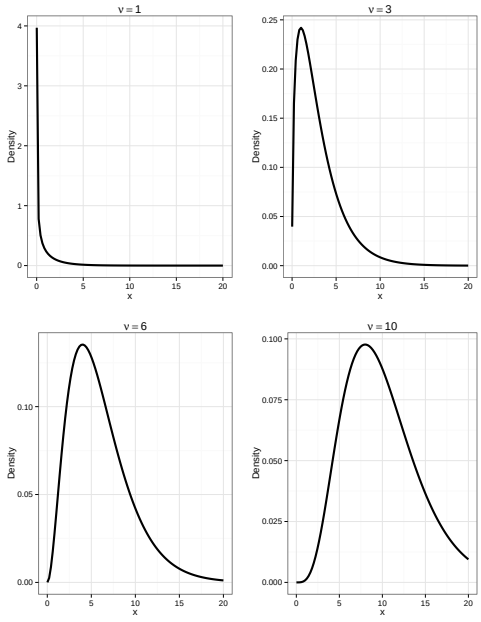
Note that the second argument, corresponding to the number of trials, has been set to 1. The histogram for this sample is shown in Figure 8.7.

Figure 8.7: Random Numbers from a Bernoulli Distribution



Note: 1000 draws from $\mathcal{B}(.2)$.

Figure 8.8: The Chi-Squared Distribution



Note: The chi-squared distribution with 1, 3, 6, and 10 degrees of freedom.

8.3.3 Chi-Squared Distribution

Description The chi-squared distribution plays an important role in hypothesis testing, in particular tests of the statistical independence between two variables. Imagine that we have a variable X that takes on non-negative values. One possible distribution for such a variable is the chi-squared distribution: $X \sim \chi^2_\nu$, where $\nu > 0$ are the so-called degrees of freedom. The probability density function is complex and can be found on Wikipedia. Figure 8.8 shows the chi-squared distribution for four different values of ν.

R The chi-squared probability density function can be evaluated using `dchisq`. For example, to evaluate the density at $X = 4$ when $\nu = 3$, we issue the following syntax:

```
dchisq(4, 3)

## [1] 0.1079819
```

The cumulative probability at $X = 4$ with $\nu = 3$ can be evaluated via

```
pchisq(4, 3)

## [1] 0.7385359
```

This is equal to $\Pr(X \leq 4)$. To evaluate the value at which a particular cumulative probability is reached, we issue the `qchisq` command. For example,

```
qchisq(.9, 3)

## [1] 6.251389
```

shows that $\Pr(X \leq 6.251) = .9$. The `qchisq` command has the option `lower.tail`.

Random numbers can be drawn using `rchisq`. For example,

```
rchisq(10, 3)

##  [1] 3.2602094 2.1408397 2.1199618 0.6719882 4.8660137
##  [6] 0.6352690 2.8129698 1.2885902 2.7700754 1.1617218
```

produces a sample of 10 observations drawn from a χ^2_3-distribution.

8.3.4 F Distribution

Description The F distribution, also known as Snedecor's distribuion or the Fisher-Snedecor distribution, is another staple of hypothesis testing. It is used extensively in tests of variances as well as analysis of variance, and will play a crucial role in the second part of this course. Lke the chi-squared distribution, the F distribution has a non-negative support. The probability density function is again quite complex and can be found on Wikipedia. Important to know is that the distribution is characterized by two degrees of freedom, which influence the shape of the distribution. If a random variable X follows the F distribution, we often write $X \sim \mathcal{F}(\nu_1, \nu_2)$, where ν_1 and ν_2 are the degrees of freedom. Examples of the distribution are shown in the four panels of Figure 8.9.

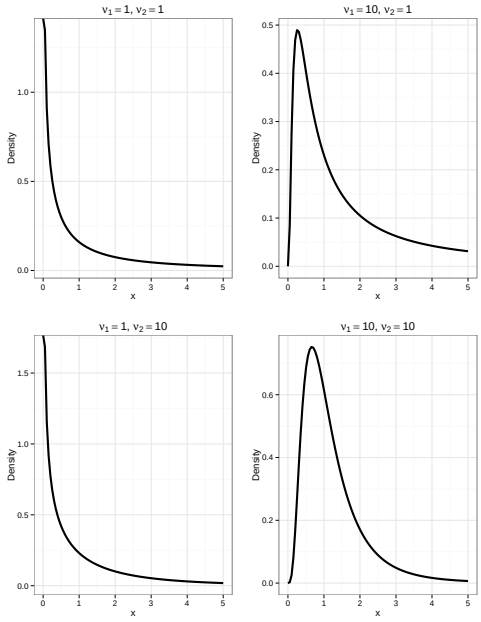
R To evaluate the value of the probability density function, we use the `df` function. For example,

```
df(2, 1, 10)

## [1] 0.1009389
```

The first argument is the value x, the second argument is ν_1, and the third arguments is ν_2.

The cumulative probability can be evaluated using `pf`. For example,

Figure 8.9: The F Distribution



Note: The F distribution with various degrees of freedom.

```
pf(2, 1, 10)

## [1] 0.8123301
```

gives the cumulative probability at $X = 2$ for a F distribution with 1 and 10 degrees of freedom. If we have the cumulative probability and want to work backwards to the value x, then we should use `qf`. For instance,

```
qf(.5, 5, 2)

## [1] 1.251925
```

states that, in a F distribution with $\nu_1 = 5$ and $\nu_2 = 2$, $\Pr(X \leq 1.252) = .5$. The `lower.tail` option is again available.

Random numbers can be generated using `rf`. For example,

```
rf(5, 10, 10)

## [1] 0.2745938 0.5705225 0.5449951 0.7770402 0.6661139
```

generates 5 random numbers drawn from $\mathcal{F}(10, 10)$.

8.3.5 Normal Distribution

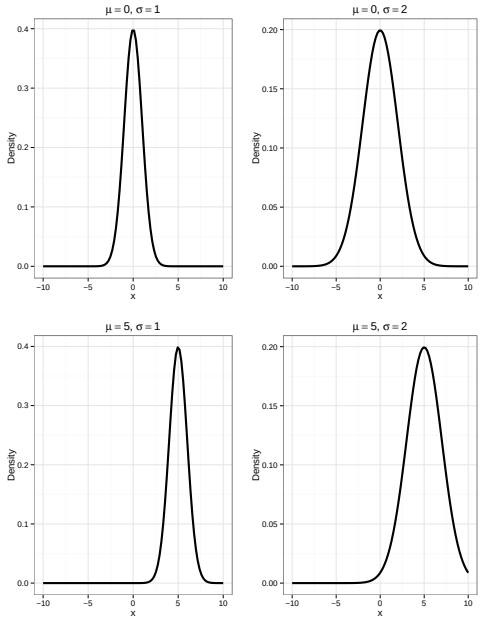
Description Of all the statistical distributions, the normal distribution without a doubt takes the center stage in social science. Although there are not that many social and political phenomena that follow the normal distribution exactly, it is often a useful starting point. The distribution is relatively simple and thi certainly contributes to its appeal. However, there are also plenty of theoretic reasons to opt for a normal distribution. Imagine, for example, that public opinion polls on an issue tends to some mean. However, there is random fluctuation around this mean due to different methodologies. If we assume that deviations above the mean are just as likely than deviations below the mean and that large deviations are improbable, then the normal distribution suggests itself. In statistics, too, the normal distribution emerges frequently, for example, as the sampling distribution of the sample mean (more about this in a couple of weeks).

The normal distribution can be used when the random variable takes on values in the interval $\pm\infty$. The probability density function is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Here, $\pi \approx 3.14159$, μ is the mean, and $\sigma > 0$ is the standard deviation. When X follows the normal distribution, we typically write $X \sim \mathcal{N}(\mu, \sigma)$. Figure 8.10 shows a variety of normal distributions.

Figure 8.10: The F Distribution



Note: The normal distribution with various means and standard deviations.

The distribution in the top-left panel is known as the **standard normal distribution**. It is written as $\mathcal{N}(0, 1)$ and is characterized by a mean of 0 and a standard deviation of 1. Notice that the apex of this distribution is situated at the mean; this is always the case with the normal distribution. The top-right panel shows what happens when we keep the mean at 0 but increase the standard deviation. The apex remains at 0, but the values are now spread out more due to the greater standard deviation. In the bottom-left panel, we observe the effect of changing the mean. Compared to the top-left panel, the spread in the distribution remains the same: in both cases, $\sigma = 1$. However, the whole curve is now shifted to the right by 5 units, so that the apex is now at $\mu = 5$. Finally, the bottom-right panel shows the combined effect of changing the mean and the standard deviation.

R In R, values of the standard normal probability density function can be obtained through

```
dnorm(x, mean =, sd = )
```

The first argument pertains to the value at which the probability density function needs to be evaluated. The second argument specifies the mean of the normal distribution; by default, this is 0. The final argument specifies the standard deviation of the normal distribution; by default, this is 1. The default settings mean that

```
dnorm(0)
```

evaluates the standard normal distribution at 0. If instead, we weant to evaluate $\mathcal{N}(2, 2)$ at $X = 0$, then we issue the following command:

```
dnorm(0, mean = 2, sd = 2)

## [1] 0.1209854
```

The cumulative probability is obtained using the `pnorm` function, which has the same arguments and defaults as `dnorm`. Given $\mathcal{N}(-2, 3)$, for example, the probability that X takes on a value of 0 or less is given by

```
pnorm(0, mean = -2, sd = 3)

## [1] 0.7475075
```

Reversing the process, `qnorm` finds the value of X that is associated with a cumulative probability of p. For example, to discover for which value of the standard normal distribution the cumulative probability is .95, we issue the following command:

```
qnorm(.95, mean = 0, sd = 1)

## [1] 1.644854
```

For this command, too, we can choose whether we want to evaluate the upper or the lower tail of the distribution.

Finally, random numbers can be drawn from a normal distribution by using `rnorm`. For example, to draw 15 observations from $\mathcal{N}(2, 4)$, the following syntax suffices:

```
rnorm(15, mean = 2, sd = 4)

##  [1]  1.2813207  0.7995045 -2.2780135  0.1071075  3.4154548
##  [6]  5.3287114 -1.4347779  3.3659262  6.2251219  5.7542229
## [11]  4.9496537  3.9265557  7.5719458  0.2038080 -4.1441288
```

8.3.6 Poisson Distribution

Description The Poisson distribution applies to discrete random variables that can take on values $0, 1, 2, \cdots$. A politically relevant example of such a variable is a protest count over a particular time period. For example, it could be that in 1910, a grand total of 11 industrial strikes took place in a particular country. Then 11 is the realized value of a discrete random variable. The protest variable is discrete because it makes little sense to speak, for example, of 10.25 strikes.

While discrete variables with values $0, 1, 2, \cdots$ can be captured through a variety of distribution, a popular choice has always been the Poisson distribution. This has a probability mass function

$$f(x) \quad = \quad \frac{\lambda^x}{x!} e^{-\lambda}$$

where $\lambda > 0$ is the sole parameter. The shorthand is $X \sim \mathcal{P}(\lambda)$.

R Imagine that we want to know the probability that 0 strikes took place. When we know that $\lambda = 2$, the probability can be computed using

```
dpois(0, lambda = 2)

## [1] 0.1353353
```

Imagine that we want to know the probability of 2 strikes or fewer when $\lambda = 2$. This probability can be obtained as follows:

```
ppois(2, lambda = 2)

## [1] 0.6766764
```

Reversing the process, if we know that the cumulative probability is .95, the value of X can be computed as

```
qpois(.95, lambda = 2)

## [1] 5
```

Here, we can again change the value of `lower.tail` to `TRUE`—the scenario that we just computed—or `FALSE`.

Finally,

```
rpois(10, lambda = 2)

##  [1] 1 3 0 2 4 2 0 2 0 1
```

draws 10 random numbers from $\mathcal{P}(2)$.

8.3.7 Student's t-Distribution

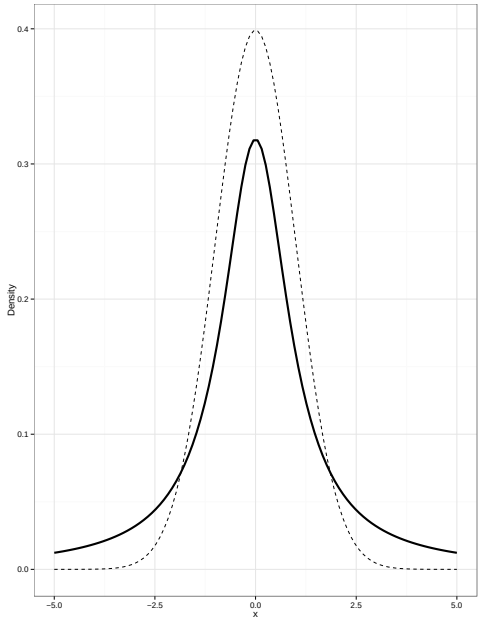
Description Student's t-distribution is a bell-shaped and symmetrical distribution whose shape depends on a single parameter, the degrees of freedom, $\nu > 0$. The probability density function is complex and may be found on Wikipedia. More important for present purposes is that you learn to recognize the shape of the distribution, which is shown—along with the standard normal distribution—in Figure 8.11. From this figure, it is clear that the t-distribution looks a lot like a standard normal distribution, with the exception that it has much heavier tails. By this, we mean that there is more probability mass in the tails of the t-distribution than in the tales of the standard normal distribution. However, as ν increases, then this distinction disappears. Asymptotically, then, as $\nu \to \infty$, student's t and the standard normal distributions become indistinuishable. In practice, the differences are already very small when $\nu = 30$.

When a random variable follows the t-distribution, we write $X \sim \mathcal{T}(\nu)$. In the social sciences, not all that many phenomena follow a t-distribution. However, one class of random variables, so-called test statistics, frequently follows this distribution. We shall see this when we come to the topic of hypothesis testing.

R Imagine $X \sim \mathcal{T}(10)$. The following now evaluates the probability density function at $X = 2$:

```
dt(2, df = 10)

## [1] 0.06114577
```

Figure 8.11: Student's t-Distribution



Note: The t-distribution is shown here for $\nu = 1$. The dashed line shows the graph of the standard normal distribution.

To evaluate the cumulative probability at $X = 2$, we issue

```
pt(2, df = 10)

## [1] 0.963306
```

Now imagine we want to know for which x, $\Pr(X \geq x) = .80$. This can be done as follows:

```
qt(.8, df = 10, lower.tail = FALSE)

## [1] -0.8790578
```

Note that we specify `lower.tail = FALSE` because we want to explore the upper tail of the t-distribution, as indicated by the $\geq$ symbol in the probability expression. Finally,

```
rt(5, df = 10)

## [1] -1.54278976  0.20702798  0.03392175 -0.19208024
## [5] -2.17059802
```

gives us 5 random draws from $\mathcal{T}(10)$.

8.3.8 Other Distributions

In the social sciences and statistics, there are many more distributions that are relevant. Many of these have also been implemented in R. CRAN shows a list of everything that is available in R, which is a lot.

8.4 Moments

To obtain the moments of the distributions that we have discussed, one can proceed in two ways. First, we can approach empirically and use the `moments` library. In this case, we draw a very large sample (e.g., $n = 1'000'000$) from a distribution and compute the moments based on this sample. Next, the `actuar` library contains the theoretical moments for a limited number of distributions. Here, we can obtain the moments without first having to generate data. This section illustrates both libraries for a limited number of distributions.

8.4.1 Working with `moments`

Let us return to the continuous uniform distribution. For $\mathcal{U}(a, b)$, we know that the first moment about 0 is equal to $\frac{1}{2}(a + b)$ (see, for example, Wikipedia). We also know that the second central moment, i.e., the variance, is equal to $\frac{1}{12}(b - a)^2$. Finally, the third and fourth standardized moments are 0 and $-\frac{6}{5}$.

We now evaluate these moments for a particular uniform distribution, to wit $\mathcal{U}(0, 10)$. We generate a sample of 1'000'000 observations from this distribution:

```
set.seed(1234)
x <- runif(1000000, 0, 10)
```

We now activate the `moments` library, which we used in Chapter 5 for computing skewness and kurtosis. We then ask for the first raw moment:

```
library(moments)
moment(x, order = 1, central = FALSE)

## [1] 4.998109
```

When we round this we obtain 5, which is the correct value. We can also ask for the second central moment:

```
moment(x, order = 2, central = TRUE)

## [1] 8.342678
```

This is very close to the actual value, which is 8.33. The third standardized moment is obtained via the `skewness` function:

```
skewness(x)

## [1] 0.001430739
```

When we round this, we get 0, which is the correct value. Finally, the fourth standardized moment is computed as follows:

```
kurtosis(x)

## [1] -1.201462
```

This, too, is very close to the actual value of -1.2 of the fourth standardized moments.

We can apply these functions to any and all distributions. The key is that we draw a large sample. Even with 1'000'000 observations, we see that the computed values deviate just the slightest amount from the true moments. In small samples, this will happen much more frequently, due to the greater sampling fluctuation.

8.4.2 Working with `actuar`

For a limited number of distributions, the `actuar` package will provide the moments without us having to generate a sample. A downside is that only raw

moments are computed. This means that central and standardized moments will have to be derived with some tricks.

Let us show the process for the chi-squared distribution. To obtain the first raw moment of a χ^2_5-distribution, we can do the following:

```
library(actuar)

##
## Attaching package:  'actuar'
## The following object is masked from 'package:grDevices':
##
##     cm

mchisq(1, 5)

## [1] 5
```

We see that the first argument is the order of the moment and the second argument is the degrees of freedom. Since the first raw moment is equal to the mean, we now know that the expectation of a chi-squared variate is the degrees of freedom. Consulting, for example, Wikipedia, it is easily verified that the variance of a chi-squared variate is 2μ. This is the second central moment, which is equal to

$$E[(X - E[X])^2] = \underbrace{E[X^2]}_{\text{2nd raw moment}} - \underbrace{(E[X])^2}_{\text{1st raw moment squared}}$$

We can implement this using

```
mchisq(2, 5) - (mchisq(1, 5))^2

## [1] 10
```

It is easily verified that the result is correct. For the skewness and kurtosis, I would suggest using the `moments` package, since the tricks needed to compute these are quite complex.

8.4.3 Exercises

(1) Imagine that X measures the number of newspaper articles dedicated to a particular political party in a particular year. We assume that $X \sim \mathcal{P}(\lambda)$ and that the average number of newspaper articles dedicated to the party is $\lambda = 10$. What is the probability that, in a given year, the party receives coverage in 20 articles?

(2) And what is the probability that the party receives coverage in 10 articles or more?

(3) Consider $X \sim \mathcal{T}(2)$. What is $\Pr(0 \leq x \leq 2)$?

(4) Consider an F distribution with 5 and 100 degrees of freedom. Simulate 10'000'000 observations. What are the mean, variance, skewness, and kurtosis?

(5) Consider the standard normal distribution. For what value of X is the cumulative probability equal to .975?

(6) Imagine we are interested in the number of days that a coalition government survives after it is put in place. Imagine that survival time follows a chi-squared distribution. What would be the probability that the government stays in power for more than one year (365 days) when the average survival time of the government is 460 days?

Part IV

Statistical Inference

Chapter 9

Hypothesis Tests for Single Variables

In this chapter, we discuss a number of procedures that are commonly used to test hypotheses about a single variable. We begin by showing how one can use R's probability functions to perform hypothesis tests. Next, we show tests for proportions, means, variances, and skewness/kurtosis.

9.1 Required Packages

The two required packages for these exercises are `EnvStats` and `fBasics`. These contain the test routines that we shall use to test hypotheses about the variance, skewness, and kurtosis.

9.2 Testing Hypotheses by Hand

Although R contains many automated procedures for hypothesis testing, it is often quite simple to do all the work by hand, using the software only as a calculator. We show how this is done in the Fisher and Neyman-Pearson approaches.

9.2.1 Implementing Fisher's Approach to Test a Hypothesis about a Proportion

Imagine, we are interested in forecasting a referendum. We formulate the null hypothesis that voters are undecided: $H_0 : \pi = .5$, where π is the probability that a person will vote in favor of the proposal. We randomly draw a sample of 10 individuals and find that 9 of them plan to vote for the proposal. Should we reject the null hypothesis at a significance level of .05?

To answer this question, we begin by computing the probability that 9 out of 10 individuals vote in favor if the null hypothesis is true. This can be done

by relying on the binomial distribution:

```
dbinom(9, 10, .5)

## [1] 0.009765625
```

We need to combine this probability with the probability of outcomes that are just as unlikely or even less likely. There are three such outcomes: 0, 1, and 10. Thus, the p-value is

```
dbinom(0, 10, .5) + dbinom(1, 10, .5) + dbinom(9, 10, .5) +
  dbinom(10, 10, .5)

## [1] 0.02148438
```

Since this value falls below the significance level, we reject the null hypothesis.

9.2.2 Implementing the Neyman-Pearson Approach to Test a Hypothesis about a Proportion

Someone believes that the referendum will fail and argues that the probability that a citizen votes for a proposal is only .4. We call this H_a. The null hypothesis, as before, is that citizens are undecided: $H_0 : \pi = .5$. We draw a bigger sample of $n = 400$ and find that 180 plan to vote for the proposal. The Type-I error rate is .05.

Let X denote the number of yes votes. We invoke the central limit theorem and assume that $X \sim \mathcal{N}(n \cdot \pi, \sqrt{n \cdot \pi \cdot (1 - \pi)})$. Under the null hypothesis $\pi_0 = .5$. We want to find the critical value x_c such that

$$\Pr\left(Z \leq \frac{X_c - n \cdot \pi_o}{\sqrt{n \cdot \pi_0 \cdot (1 - \pi_0)}}\right) = \alpha,$$

where Z is a standard normal variate. The reason we are looking in the left tail of the standard normal distribution, as indicated by the less-then-or-equal sign, is that the hypothesized value under H_a is less than under H_0. To find the value of the standard normal variate corresponding to .05, we issue

```
z <- qnorm(.05, mean = 0, sd = 1, lower.tail = TRUE)
z

## [1] -1.644854
```

We now obtain X_c as follows:

```
n <- 400
pi0 <- 0.5
x.crit <- z*sqrt(n*pi0*(1-pi0))+n*pi0
x.crit
```

```
## [1] 183.5515
```

where `pi0` is π_0. If the observed value of X is less then or equal to 183.551, then we reject the null hypothesis. That is the case here:

```
isTRUE(180<=x.crit)

## [1] TRUE
```

We reject the null hypothesis in favor of the alternative hypothesis.

It is also easy to compute the statistical power. The statistical power is the probability of rejecting H_0 assuming that H_A is true. This is computed as follows:

```
pi1<-0.4
w <- (x.crit - n*pi1)/sqrt(n*pi1*(1-pi1))
power <- pnorm(w, mean = 0, sd = 1, lower.tail = TRUE)
power

## [1] 0.9918852
```

Hence, we have plenty of power in our test.

9.3 Testing Proportions

We just performed a test of proprtions by hand. However, we can also let R perform this test for us. We illustrate the process using the `demo` variable in the `world indicators.dta` data. As our research hypothesis, we formulate the notion that, in 2013, democracies where still less common than non-democracies around the globe. If it is equally likely to observe a democracy and a non-democracy, then $\pi = \Pr(Democracy) = .5 = \Pr(Non - Democracy)$. Thus, our research hypothesis is

$$H_A : \quad \pi < .5$$

The null hypothesis negates the research hypothesis and, as such, can be formulated as

$$H_0 : \pi \geq .5$$

This hypothesis will now be tested. We do this at a significance level of .10.

To perform the test, we use the `prop.test` function. In its simplest form, this function requires a couple of arguments: (1) a count of the number of democracies (or, more generally, a count of the 1s); (2) the sample size; (3) a specification of π_0; and (4) an indication of the test direction. To obtain the first two ingredients, we start by creating a table:

```
library(foreign)
world <- read.dta("world indicators.dta")
table(world$demo)

##
## Non-Democratic     Democratic
##             89             70
```

We observe that the number of democracies is 70 and that the sample size is 159. We can now issue the following syntax:

```
prop.test(70, 159, p = .5, alternative = "less")

##
##  1-sample proportions test with continuity correction
##
## data:  70 out of 159, null probability 0.5
## X-squared = 2.0377, df = 1, p-value = 0.07672
## alternative hypothesis: true p is less than 0.5
## 95 percent confidence interval:
##  0.0000000 0.5086092
## sample estimates:
##         p
## 0.4402516
```

Looking at the results, we see that the null probability has been set to 0.5 and the alternative hypothesis is that the "true p is less than 0.5." This is conform the specification of H_A. We also see that the estimated probability of encountering a democracy is 0.44. This clearly is in the direction of H_A. The question is whether we can reject H_0. The answer to that question can be found in the line starting with `X-squared`. Here, we observe a test statistic of 2.0377. Using a χ^2_1 sampling distribution for the test statistic, we obtain a p-value of .07672. This is smaller than the significance level, so we reject H_0 in favor of H_A.

A couple of comments are in order. The way I have presented the test, I have mixed elements from the Fisher and Neyman-Pearson approaches. From Fisher, I have borrowed the ideas of a significance level and a p-value. From Neyman and Pearson, I have borrowed the notion that there is an alternative hypothesis. In practice, social scientists often engage in this kind of mixing of the approaches. As long as one remains aware of their philosophical differences, then there should be no problem in doing so.

Second, the test deviates a bit from the one we did by hand. Instead of relying on the normal approximation, what the current test does is to compare the observed distribution to the distribution that would have arisen if π were 0.5. This test logic is described in the notes on applied hypothesis testing. The χ-squared test statistic, however, is practically the same as when we would

compute a standard normal test statistic, as we did earlier, and square it.[1]

9.4 Testing Means

Testing Hypotheses about a Single Mean

The 2012 American National Election Study (ANES) asked respondents to indicate their feelings toward liberals on a so-called feeling thermometer. This measure runs from 0 (very cool feelings) to 100 (very warm feelings), and traditionally 50 has been treated as the neutral catgeory. A researcher formulates the hypothesis that Americans generally feel negatively about liberals: $H_A : \mu < 50$. Instead of testing this hypothesis, we test the complementary null hypothesis: $H_0 : \mu \geq 50$. How do we perform this test in R?

The answer is that we use the `t.test` function. This has three central arguments: (1) an indication of the hypothesized value; (2) a specification of the direction of the test; and (3) a reference to the variable in the data. Our data are contained in the file `feeltherm.dta`. The relevant variable is `feel_liberal`:

```
feel.dat <- read.dta("feeltherm.dta")
summary(feel.dat$feel_liberal)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   30.00   50.00   48.31   65.00  100.00     541
```

To test the null hypothesis, we no issue the following syntax:

```
t.test(feel.dat$feel_liberal, mu = 50, alternative = "less")

##
##  One Sample t-test
##
## data:  feel.dat$feel_liberal
## t = -4.8105, df = 5372, p-value = 7.733e-07
## alternative hypothesis: true mean is less than 50
## 95 percent confidence interval:
##      -Inf 48.88519
## sample estimates:
## mean of x
##  48.30579
```

What can we conclude from the test? First of all, our sample mean is 48.31, which is in the direction of H_A. Second, the test statistic is -4.81. When referred to a Student's t-distribution with 5372 degrees of freedom, we find $p = 0.000$

[1] Note that the square of a standard normal variate is χ-squared distributed with 1 degree of freedom.

(the rounded value of 7.733e-07). We would reject H_0 at any conventional significance level. The conclusion, then, is that the we side with the researcher, at least until new data come along.

Testing Hypotheses about Means in Paired Samples

The same data set also contains feeling thermometer ratings for conservatives. The same researcher now hypothesizes that Americans feel warmer toward conservatives than liberals: $H_A : \mu_C > \mu_L$ or, alternatively, $\mu_C - \mu_L > 0$. The null hypothesis that we test negates the research hypothesis: $H_0 : \mu_C - \mu_L \leq 0$.

Feelings toward conservatives and liberals are measured within the same individuals, so this is a classical example of a paired samples t-test. What we do is to generate a new variable that measures the difference in feelings toward conservatives and liberals in each group. From the algebra of expectations, we know that $\mu_{C-L} = \mu_C - \mu_L$. Hence, by formulating the null hypothesis as $\mu_{C-L} \leq 0$ we capture the original null hypothesis $\mu_C - \mu_L \leq 0$.

We start the procedure by generating the new variable:

```
library(dplyr)
feel.dat <- mutate(feel.dat, diffeel =
              feel_conservative - feel_liberal)
summary(feel.dat$diffeel)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -100.000  -20.000    0.000    5.016   30.000  100.000
##     NA's
##      572
```

We now perform the t-test on the new variable:

```
t.test(feel.dat$diffeel, mu = 0, alternative = "greater")

##
##  One Sample t-test
##
## data:  feel.dat$diffeel
## t = 8.7368, df = 5341, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  4.071262      Inf
## sample estimates:
## mean of x
##  5.015724
```

We observe that respondents like conservatives better than liberals by a margin of roughly 5 points (the sample mean on `diffeel`). This difference favors H_A. The test-statistic is 8.74. When referred to a t-distribution with 5341 degrees

of freedom, the p-value is 0.000. We reject the null hypothesis in favor of the research hypothesis.

9.5 Testing Variances

Our researcher is extremely prolific and generates yet another hypothesis: feelings toward liberals are above 50 percent of maximum polarization. While it may not look like it, this can be construed as a hypothesis about the variance. Under maximum polarization, there is a chance of .5 that someone gives liberals a score of 0 and an equal chance that she gives them a score of 100. In this case, the variance is equal to 2500. Our researcher thus argues that the variance in feelings toward liberals is greater than 1250. The corresponding null hypothesis is that the variance is less or equal to 1250: $H_0 : \sigma^2 \leq 1250$.

We can test the null hypothesis using a χ^2-test, which is implemented in the `EnvStats` package. The `varTest` command in this package takes four arguments: (1) a reference to the data; (2) a specification of the hypothesized value; (3) an indication of the direction of the test; and (4) a significance level. The following syntax performs the test of interest:

```
library(EnvStats)

##
## Attaching package:  'EnvStats'
## The following objects are masked from 'package:actuar':
##
##    dpareto, ppareto, qpareto, rpareto
## The following objects are masked from 'package:rapportools':
##
##    iqr, kurtosis, skewness
## The following objects are masked from 'package:moments':
##
##    kurtosis, skewness
## The following objects are masked from 'package:stats':
##
##    predict, predict.lm
## The following object is masked from 'package:base':
##
##    print.default

suppressWarnings(varTest(feel.dat$feel_liberal,
  sigma.squared = 1250, alternative = "greater",
  conf.level = 0.95))

##
## Results of Hypothesis Test
## --------------------------
```

```
##
## Null Hypothesis:                 variance = 1250
##
## Alternative Hypothesis:          True variance is greater than 1250
##
## Test Name:                       Chi-Squared Test on Variance
##
## Estimated Parameter(s):          variance = 666.4517
##
## Data:                            feel.dat$feel_liberal
##
## Test Statistic:                  Chi-Squared = 2864.143
##
## Test Statistic Parameter:        df = 5372
##
## P-value:                         1
##
## 95% Confidence Interval:         LCL = 645.8192
##                                  UCL =      Inf
```

We observe that the sample variance is way smaller than 1250. Consequently, it should not come as a surprise that we fail to reject the null hypothesis. The χ^2 test statistic is huge at 2864.143. When referred to a χ^2-distribution with $n - 1 = 5372$ degrees of freedom, the p-value is 1. Clearly, there is no reason to reject the null hypothesis in favor of the researcher's claim.

9.6 Testing Normality

The final test we shall discuss is the D'Agostino's test of normality, which simultaneously considers skewness and excess kurtosis. It is well known that the normal distribution has no skewness or excess kurtosis, so that the null hypothesis is that both of these moments are 0. The test assesses the actual degree of skewness and excess kurtosis in the data and determines if this is consistent with the null hypothesis. It also allows us to detemine where the problem lies: with skewness, kurtosis, or both.

The version of the test that we like the best is implemented in the `fBasics` library. It is the `dagoTest` command and, in its basic form, it takes only one argument: a data vector. We apply it to feelings about liberal groups.

```
detach("package:EnvStats", unload=TRUE)
library(fBasics)

## Loading required package:  timeDate
##
## Attaching package:  'timeDate'
```

```
## The following objects are masked from 'package:rapportools':
##
##    kurtosis, skewness
## The following objects are masked from 'package:moments':
##
##    kurtosis, skewness
## Loading required package:  timeSeries
##
## Attaching package:  'timeSeries'
## The following object is masked from 'package:psych':
##
##    outlier
##
## Rmetrics Package fBasics
## Analysing Markets and calculating Basic Statistics
## Copyright (C) 2005-2014 Rmetrics Association Zurich
## Educational Software for Financial Engineering and Computational
Science
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
## https://www.rmetrics.org --- Mail to:  info@rmetrics.org
##
## Attaching package:  'fBasics'
## The following object is masked from 'package:psych':
##
##    tr
```

```
x <- na.omit(feel.dat$feel_liberal)
dagoTest(x)
```

```
##
## Title:
##  D'Agostino Normality Test
##
## Test Results:
##   STATISTIC:
##     Chi2 | Omnibus: 150.7298
##     Z3  | Skewness: -8.4747
##     Z4  | Kurtosis: -8.8831
##   P VALUE:
##     Omnibus  Test: < 2.2e-16
##     Skewness Test: < 2.2e-16
##     Kurtosis Test: < 2.2e-16
##
## Description:
##  Mon Oct 16 10:35:24 2017 by user: binding
```

(Make sure to unload `EnvStats` using `detach`; otherwise `dagoTest` does not yield any output.) If we are looking to test skewness and excess kurtosis jointly, then we should look at the omnibus test. The test statistic here is roughly 150.73. When referred to a χ^2-distribution with 2 degrees of freedom, this yields $p = .000$, which is suficient evidence to reject the null hypothesis. The problem is with both skewness and kurtosis. The test statistic for skewness is -8.47. When referred to the standard normal distribution, we obtain $p = .000$, so that we reject the null hypothesis that the skewness is 0. The test statistic for excess kurtosis is -8.88. Again using the standard normal distribution, the p-value is .000. Thus, we reject the hypothesis of no excess kurtosis as well.

9.7 Exercises

(1) Imagine a scholar who argues that the fraction of Democratic party identifiers to date is .3. We take this as the null hypothesis. We conduct a survey among 2229 respondents and find that 802 of them identify as Democrats. Use R to test the null hypothesis, setting the Type-I error rate to .05. Should it be rejected or not? Motivate your answer, i.e., show the test statistic and p-value.

(2) A scholar argues that feeliings toward big business are, on the average, negative in the aftermath of the economic crisis. The 2012 American National Election Studies (ANES) contain a feeling thermometer rating for big business. The measure is included in the `feel.dta` data. Formulate the null hypothesis that goes along with the research hypothesis. Then test this hypothesis. What do you conclude and why?

(3) Using again feelings toward big business, can we say that this variable is distributed normally in the population? Motivate your answer.

(4) Consider the research hypothesis that the variance in the population in feelings toward big business is at most 500. Formulate and test the relevant null hypothesis. What do you conclude and why?

(5) The 2012 ANES respondents also asks respondents about their feelings toward unions. Imagine your research hypothesis is that, on the average, Americans like big business better than unions. Formulate and test the relevant null hypothesis. What do you conclude and why?