Introduction to Probability and Statistics Using R

Third Edition

G. Jay Kerns

2017-02-05

Contents

Chapter 1

An Introduction to Probability and Statistics

This chapter has proved to be the hardest to write, by far. The trouble is that there is so much to say – and so many people have already said it so much better than I could. When I get something I like I will release it here.

In the meantime, there is a lot of information already available to a person with an Internet connection. I recommend to start at Wikipedia, which is not a flawless resource but it has the main ideas with links to reputable sources.

In my lectures I usually tell stories about Fisher, Galton, Gauss, Laplace, Quetelet, and the Chevalier de Mere.

1.1 Probability

The common folklore is that probability has been around for millennia but did not gain the attention of mathematicians until approximately 1654 when the Chevalier de Mere had a question regarding the fair division of a game's payoff to the two players, supposing the game had to end prematurely.

1.2 Statistics

Statistics concerns data; their collection, analysis, and interpretation. In this book we distinguish between two types of statistics: descriptive and inferential.

Descriptive statistics concerns the summarization of data. We have a data set and we would like to describe the data set in multiple ways. Usually this entails calculating numbers from the data, called descriptive measures, such as percentages, sums, averages, and so forth.

Inferential statistics does more. There is an inference associated with the data set, a conclusion drawn about the population from which the data originated.

There are two schools of thought regarding statistics: frequentist and bayesian. The difference between the schools is related to how the two groups interpret the underlying probability (see Section 4.3). The frequentist school gained a lot of ground among statisticians due in large part to the work of Fisher, Neyman, and Pearson in the early twentieth century. That dominance lasted until inexpensive computing power became widely available; nowadays the bayesian school is garnering more attention and at an increasing rate.

This book is devoted mostly to the frequentist viewpoint because that is how I was trained, with the conspicuous exception of Sections 4.8 and **??**.

Chapter 2

An Introduction to R

Every R book I have ever seen has had a section/chapter that is an introduction to R, and so does this one. The goal of this chapter is for a person to get up and running, ready for the material that follows. See Section 2.5 for links to other material which the reader may find useful.

What do I want them to know?

- Where to find R to install on a home computer, and a few comments to help with the usual hiccups that occur when installing something.
- Abbreviated remarks about the available options to interact with R.
- Basic operations (arithmetic, entering data, vectors) at the command prompt.
- How and where to find help when they get in trouble.
- Other little shortcuts I am usually asked when introducing R.

2.1 Downloading and Installing R

The instructions for obtaining R largely depend on the user's hardware and operating system. The R Project has written an R Installation and Administration manual with complete, precise instructions about what to do, together with all sorts of additional information. The following is just a primer to get a person started.

2.1.1 Installing R

Visit one of the links below to download the latest version of R for your operating system:

- Microsoft Windows: http://cran.r-project.org/bin/windows/base/
- MacOS: http://cran.r-project.org/bin/macosx/
- Linux: http://cran.r-project.org/bin/linux/

On Microsoft Windows, click the `R-x.y.z.exe` installer to start installation. When it asks for "Customized startup options", specify `Yes`. In the next window, be sure to select the SDI (single document interface) option; this is useful later when we discuss three dimensional plots with the `rgl` package [1].

Installing R on a USB drive (Windows)

With this option you can use R portably and without administrative privileges. There is an entry in the R for Windows FAQ about this. Here is the procedure I use:

1. Download the Windows installer above and start installation as usual. When it asks *where* to install, navigate to the top-level directory of the USB drive instead of the default `C` drive.
2. When it asks whether to modify the Windows registry, uncheck the box; we do NOT want to tamper with the registry.
3. After installation, change the name of the folder from `R-x.y.z` to just plain R. (Even quicker: do this in step 1.)
4. Download this shortcut and move it to the top-level directory of the USB drive, right beside the R folder, not inside the folder. Use the downloaded shortcut to run R.

Steps 3 and 4 are not required but save you the trouble of navigating to the `R-x.y.z/bin` directory to double-click `Rgui.exe` every time you want to run the program. It is useless to create your own shortcut to `Rgui.exe`. Windows does not allow shortcuts to have relative paths; they always have a drive letter associated with them. So if you make your own shortcut and plug your USB drive into some *other* machine that happens to assign your drive a different letter, then your shortcut will no longer be pointing to the right place.

2.1.2 Installing and Loading Add-on Packages

There are *base* packages (which come with R automatically), and *contributed* packages (which must be downloaded for installation). For example, on the version of R being used for this document the default base packages loaded at startup are

```
getOption("defaultPackages")
```

```
[1] "datasets"  "utils"     "grDevices" "graphics"
[5] "stats"     "methods"
```

The base packages are maintained by a select group of volunteers, called R Core. In addition to the base packages, there are over ten thousand additional contributed packages written by individuals all over the world. These are stored worldwide on mirrors of the Comprehensive R Archive Network, or `CRAN` for short. Given an active Internet connection, anybody is free to download and install these packages and even inspect the source code.

To install a package named `foo`, open up R and type `install.packages("foo")` . To install `foo` and additionally install all of the other packages on which `foo` depends, instead type `install.packages("foo", depends = TRUE)`.

The general command `install.packages()` will (on most operating systems) open a window containing a huge list of available packages; simply choose one or more to install.

No matter how many packages are installed onto the system, each one must first be loaded for use with the `library` function. For instance, the `foreign` package [105] contains all sorts of functions needed to import data sets into R from other software such as SPSS, SAS, *etc*. But none of those functions will be available until the command `library("foreign")` is issued.

Type `library()` at the command prompt (described below) to see a list of all available packages in your library.

For complete, precise information regarding installation of R and add-on packages, see the R Installation and Administration manual.

2.2 Communicating with R

2.2.1 One line at a time

This is the most basic method and is the first one that beginners will use.

- RGui (Microsoft ® Windows)
- RStudio
- Terminal
- Emacs/ESS, XEmacs

2.2.2 Multiple lines at a time

For longer programs (called *scripts*) there is too much code to write all at once at the command prompt. Furthermore, for longer scripts it is convenient to be able to only modify a certain piece of the script and run it again in R. Programs called *script editors* are specially designed to aid the communication and code writing process. They have all sorts of helpful features including R syntax highlighting, automatic code completion, delimiter matching, and dynamic help on the R functions as they are being written. Even more, they often have all of the text editing features of programs like Microsoft®Word. Lastly, most script editors are fully customizable in the sense that the user can customize the appearance of the interface to choose what colors to display, when to display them, and how to display them.

- **R Editor (Windows):** In Microsoft® Windows, R Gui has its own built-in script editor, called R Editor. From the console window, select `File` ▸ `New Script`. A script window opens, and the lines of code can be written in the window. When satisfied with the code, the user highlights all of the commands and presses Ctrl+R. The commands are automatically run at once in R and the output is shown. To save the script for later, click `File` ▸ `Save as...` in R Editor. The script can be reopened later with `File` ▸} `Open Script...` in `RGui`. Note that R Editor does not have the fancy syntax highlighting that the others do.
- **RStudio:**

- **Emacs/ESS:** Emacs is an all purpose text editor. It can do absolutely anything with respect to modifying, searching, editing, and manipulating, text. And if Emacs can't do it, then you can write a program that extends Emacs to do it. Once such extension is called ESS, which stands for /E/-macs /S/-peaks /S/-tatistics. With ESS a person can speak to R, do all of the tricks that the other script editors offer, and much, much, more. Please see the following for installation details, documentation, reference cards, and a whole lot more: http://ess.r-project.org. *Fair warning*: if you want to try Emacs and if you grew up with Microsoft® Windows or Macintosh, then you are going to need to relearn everything you thought you knew about computers your whole life. (Or, since Emacs is completely customizable, you can reconfigure Emacs to behave the way you want.) I have personally experienced this transformation and I will never go back.

2.2.3 Graphical User Interfaces (GUIs)

By the word "GUI" I mean an interface in which the user communicates with R by way of points-and-clicks in a menu of some sort. Again, there are many, many options and I only mention one that I have used and enjoyed.

- **R Commander** provides a point-and-click interface to many basic statistical tasks. It is called the "Commander" because every time one makes a selection from the menus, the code corresponding to the task is listed in the output window. One can take this code, copy-and-paste it to a text file, then re-run it again at a later time without the R Commander's assistance. It is well suited for the introductory level. `Rcmdr` [45] also allows for user-contributed "Plugins" which are separate packages on `CRAN` that add extra functionality to the `Rcmdr` package. The plugins are typically named with the prefix `RcmdrPlugin` to make them easy to identify in the `CRAN` package list. One such plugin is the `RcmdrPlugin.IPSUR` package [152] which accompanies this text.

2.3 Basic R Operations and Concepts

The R developers have written an introductory document entitled "An Introduction to R". There is a sample session included which shows what basic interaction with R looks like. I recommend that all new users of R read that document, but bear in mind that there are concepts mentioned which will be unfamiliar to the beginner.

Below are some of the most basic operations that can be done with R. Almost every book about R begins with a section like the one below; look around to see all sorts of things that can be done at this most basic level.

2.3.1 Arithmetic

```
2 + 3       # add
```

```
[1] 5
```

```
4 # 5 / 6   # multiply and divide
```

```
[1] 4
```

```
7^8         # 7 to the 8th power
```

```
[1] 5764801
```

Notice the comment character #. Anything typed after a # symbol is ignored by R. We know that 20/6 is a repeating decimal, but the above example shows only 7 digits. We can change the number of digits displayed with `options`:

```
options(digits = 16)
10/3                # see more digits
```

```
[1] 3.333333333333333
```

```
sqrt(2)             # square root
```

```
[1] 1.414213562373095
```

```
exp(1)              # Euler's constant, e
```

```
[1] 2.718281828459045
```

```
pi
```

```
[1] 3.141592653589793
```

```
options(digits = 7)  # back to default
```

Note that it is possible to set `digits` up to 22, but setting them over 16 is not recommended (the extra significant digits are not necessarily reliable). Above notice the `sqrt` function for square roots and the `exp` function for powers of e, Euler's number.

2.3.2 Assignment, Object names, and Data types

It is often convenient to assign numbers and values to variables (objects) to be used later. The proper way to assign values to a variable is with the <- operator (with a space on either side). The = symbol works too, but it is recommended by the R masters to reserve = for specifying arguments to functions (discussed later). In this book we will follow their advice and use <- for assignment. Once a variable is assigned, its value can be printed by simply entering the variable name by itself.

```
x <- 7*41/pi    # don't see the calculated value
x               # take a look
```

```
[1] 91.35494
```

When choosing a variable name you can use letters, numbers, dots ".", or underscore "_" characters. You cannot use mathematical operators, and a leading dot may not be followed by a number. Examples of valid names are: `x`, `x1`, `y.value`, and `!y_hat`. (More precisely, the set of allowable

characters in object names depends on one's particular system and locale; see An Introduction to R for more discussion on this.)

Objects can be of many *types*, *modes*, and *classes*. At this level, it is not necessary to investigate all of the intricacies of the respective types, but there are some with which you need to become familiar: * integer: the values 0, ±1, ±2, ...; these are represented exactly by R. * double: real numbers (rational and irrational); these numbers are not represented exactly (save integers or fractions with a denominator that is a power of 2, see Venables and Smith [164]). * character: elements that are wrapped with pairs of " or '; * logical: includes `TRUE`, `FALSE`, and `NA` (which are reserved words); the `NA` stands for "not available", *i.e.*, a missing value.

You can determine an object's type with the `typeof` function. In addition to the above, there is the `complex` data type:

```
sqrt(-1)               # isn't defined
```

```
Warning in sqrt(-1): NaNs produced
```

```
[1] NaN
```

```
sqrt(-1+0i)            # is defined
```

```
[1] 0+1i
```

```
sqrt(as.complex(-1))   # same thing
```

```
[1] 0+1i
```

```
(0 + 1i)^2             # should be -1
```

```
[1] -1+0i
```

```
typeof((0 + 1i)^2)
```

```
[1] "complex"
```

Note that you can just type `(1i)^2` to get the same answer. The `NaN` stands for "not a number"; it is represented internally as `double` .

2.3.3 Vectors

:PROPERTIES: :CUSTOM_ID: sub-Vectors :END:

All of this time we have been manipulating vectors of length 1. Now let us move to vectors with multiple entries.

Entering data vectors

The long way: If you would like to enter the data `74,31,95,61,76,34,23,54,96` into R, you may create a data vector with the `c` function (which is short for *concatenate*).

```
x <- c(74, 31, 95, 61, 76, 34, 23, 54, 96)
x
```

```
[1] 74 31 95 61 76 34 23 54 96
```

The elements of a vector are usually coerced by R to the the most general type of any of the elements, so if you do `c(1, "2")` then the result will be `c("1", "2")`.

A shorter way: : The `scan` method is useful when the data are stored somewhere else. For instance, you may type `x <- scan()` at the command prompt and R will display `1:` to indicate that it is waiting for the first data value. Type a value and press `Enter`, at which point R will display `2:`, and so forth. Note that entering an empty line stops the scan. This method is especially handy when you have a column of values, say, stored in a text file or spreadsheet. You may copy and paste them all at the `1:` prompt, and R will store all of the values instantly in the vector `x`.

*Repeated data; regular patterns:# the `seq` function will generate all sorts of sequences of numbers. It has the arguments `from`, `to`, `by`, and `length.out` which can be set in concert with one another. We will do a couple of examples to show you how it works.

```
seq(from = 1, to = 5)
```

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

```
seq(from = 2, by = -0.1, length.out = 4)
```

```
[1] 2.0 1.9 1.8 1.7
```

Note that we can get the first line much quicker with the colon operator.

```
1:5
```

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

The vector `LETTERS` has the 26 letters of the English alphabet in uppercase and `letters` has all of them in lowercase.

Indexing data vectors

Sometimes we do not want the whole vector, but just a piece of it. We can access the intermediate parts with the `[]` operator. Observe (with `x` defined above)

```
x[1]
```

```
[1] 74
```

```
x[2:4]
```

```
[1] 31 95 61
```

```
x[c(1,3,4,8)]
```

```
[1] 74 95 61 54
```

```
x[-c(1,3,4,8)]
```

```
[1] 31 76 34 23 96
```

Notice that we used the minus sign to specify those elements that we do *not* want.

```
LETTERS[1:5]
```

```
[1] "A" "B" "C" "D" "E"
```

```
letters[-(6:24)]
```

```
[1] "a" "b" "c" "d" "e" "y" "z"
```

2.3.4 Functions and Expressions

A function takes arguments as input and returns an object as output. There are functions to do all sorts of things. We show some examples below.

```
x <- 1:5
sum(x)
```

```
[1] 15
```

```
length(x)
```

```
[1] 5
```

```
min(x)
```

```
[1] 1
```

```
mean(x)       # sample mean
```

```
[1] 3
```

```
sd(x)         # sample standard deviation
```

```
[1] 1.581139
```

It will not be long before the user starts to wonder how a particular function is doing its job, and since R is open-source, anybody is free to look under the hood of a function to see how things are calculated. For detailed instructions see the article "Accessing the Sources" by Uwe Ligges [86]. In short:

Type the name of the function without any parentheses or arguments. If you are lucky then the code for the entire function will be printed, right there looking at you. For instance, suppose that we would like to see how the `intersect` function works:

```
intersect
```

```
function (x, y)
{
```

```
    y <- as.vector(y)
    unique(y[match(as.vector(x), y, 0L)])
}
<bytecode: 0x3e34ef8>
<environment: namespace:base>
```

If instead it shows `UseMethod(something)` then you will need to choose the *class* of the object to be inputted and next look at the *method* that will be *dispatched* to the object. For instance, typing `rev` says

```
rev
```

```
function (x)
UseMethod("rev")
<bytecode: 0x2eece40>
<environment: namespace:base>
```

The output is telling us that there are multiple methods associated with the `rev` function. To see what these are, type

```
methods(rev)
```

```
[1] rev.default     rev.dendrogram*
see '?methods' for accessing help and source code
```

Now we learn that there are two different `rev(x)` functions, only one of which being chosen at each call depending on what `x` is. There is one for `dendrogram` objects and a `default` method for everything else. Simply type the name to see what each method does. For example, the `default` method can be viewed with

```
rev.default
```

```
function (x)
if (length(x)) x[length(x):1L] else x
<bytecode: 0x3aea6f0>
<environment: namespace:base>
```

Some functions are hidden by a namespace (see An Introduction to R Venables and Smith [164]), and are not visible on the first try. For example, if we try to look at the code for `wilcox.test` (see Chapter **??**) we get the following:

```
wilcox.test
```

```
function (x, ...)
UseMethod("wilcox.test")
<bytecode: 0x2dcadf0>
<environment: namespace:stats>
```

```
methods(wilcox.test)
```

```
[1] wilcox.test.default* wilcox.test.formula*
see '?methods' for accessing help and source code
```

If we were to try `wilcox.test.default` we would get a "not found" error, because it is hidden behind the namespace for the package `stats` [115] (shown in the last line when we tried `wilcox.test`). In cases like these we prefix the package name to the front of the function name with three colons; the command `stats:::wilcox.test.default` will show the source code, omitted here for brevity.

If it shows `.Internal(something)` or `.Primitive(something)` , then it will be necessary to download the source code of R (which is *not* a binary version with an `.exe` extension) and search inside the code there. See Ligges [86] for more discussion on this. An example is `exp`:

```
exp
```

```
function (x)  .Primitive("exp")
```

Be warned that most of the `.Internal` functions are written in other computer languages which the beginner may not understand, at least initially.

2.4 Getting Help

When you are using R, it will not take long before you find yourself needing help. Fortunately, R has extensive help resources and you should immediately become familiar with them. Begin by clicking `Help` on `RGui`. The following options are available.

- Console: gives useful shortcuts, for instance, `Ctrl+L`, to clear the R console screen.
- FAQ on R: frequently asked questions concerning general R operation.
- FAQ on R for Windows: frequently asked questions about R, tailored to the Microsoft Windows operating system.
- Manuals: technical manuals about all features of the R system including installation, the complete language definition, and add-on packages.
- R functions (text). . . : use this if you know the *exact* name of the function you want to know more about, for example, `mean` or `plot`. Typing `mean` in the window is equivalent to typing `help("mean")` at the command line, or more simply, `?mean` . Note that this method only works if the function of interest is contained in a package that is already loaded into the search path with `library`.
- HTML Help: use this to browse the manuals with point-and-click links. It also has a Search Engine & Keywords for searching the help page titles, with point-and-click links for the search results. This is possibly the best help method for beginners. It can be started from the command line with the command `help.start()` .
- Search help . . . : use this if you do not know the exact name of the function of interest, or if the function is in a package that has not been loaded yet. For example, you may enter `plo` and a text window will return listing all the help files with an alias, concept, or title matching `plo` using regular expression matching; it is equivalent to typing `help.search("plo")` at the command line. The advantage is that you do not need to know the exact name of the function; the disadvantage is that you cannot point-and-click the results. Therefore, one may wish to use the HTML Help search engine instead. An equivalent way is `??plo` at the command line.
- search.r-project.org . . . : this will search for words in help lists and email archives of the R Project. It can be very useful for finding other questions that other users have asked.

- Apropos ...: use this for more sophisticated partial name matching of functions. See `?apropos` for details.

On the help pages for a function there are sometimes "Examples" listed at the bottom of the page, which will work if copy-pasted at the command line (unless marked otherwise). The `example` function will run the code automatically, skipping the intermediate step. For instance, we may try `example(mean)` to see a few examples of how the `mean` function works.

2.4.1 R Help Mailing Lists

There are several mailing lists associated with R, and there is a huge community of people that read and answer questions related to R. See here for an idea of what is available. Particularly pay attention to the bottom of the page which lists several special interest groups (SIGs) related to R.

Bear in mind that R is free software, which means that it was written by volunteers, and the people that frequent the mailing lists are also volunteers who are not paid by customer support fees. Consequently, if you want to use the mailing lists for free advice then you must adhere to some basic etiquette, or else you may not get a reply, or even worse, you may receive a reply which is a bit less cordial than you are used to. Below are a few considerations:

1. Read the FAQ. Note that there are different FAQs for different operating systems. You should read these now, even without a question at the moment, to learn a lot about the idiosyncrasies of R.
2. Search the archives. Even if your question is not a FAQ, there is a very high likelihood that your question has been asked before on the mailing list. If you want to know about topic `foo`, then you can do `RSiteSearch("foo")` to search the mailing list archives (and the online help) for it.
3. Do a Google search and an `RSeek.org` search.

If your question is not a FAQ, has not been asked on R-help before, and does not yield to a Google (or alternative) search, then, and only then, should you even consider writing to R-help. Below are a few additional considerations.

- Read the posting guide before posting. This will save you a lot of trouble and pain.
- Get rid of the command prompts (>) from output. Readers of your message will take the text from your mail and copy-paste into an R session. If you make the readers' job easier then it will increase the likelihood of a response.
- Questions are often related to a specific data set, and the best way to communicate the data is with a `dump` command. For instance, if your question involves data stored in a vector `x`, you can type `dump("x","")` at the command prompt and copy-paste the output into the body of your email message. Then the reader may easily copy-paste the message from your email into R and `x` will be available to him/her.
- Sometimes the answer the question is related to the operating system used, the attached packages, or the exact version of R being used. The `sessionInfo()` command collects all of this information to be copy-pasted into an email (and the Posting Guide requests this information). See Appendix **??** for an example.

2.5 External Resources

There is a mountain of information on the Internet about R. Below are a few of the important ones.

- The R- Project for Statistical Computing : Go there first.
- The Comprehensive R Archive Network : That is where R is stored along with thousands of contributed packages. There are also loads of contributed information (books, tutorials, *etc*.). There are mirrors all over the world with duplicate information.
- R-Forge : This is another location where R packages are stored. Here you can find development code which has not yet been released to `CRAN`.
- R Seek is a search engine based on Google specifically tailored for R queries.

2.6 Other Tips

It is unnecessary to retype commands repeatedly, since R remembers what you have recently entered on the command line. On the Microsoft® Windows R Gui, to cycle through the previous commands just push the ↑ (up arrow) key. On Emacs/ESS the command is `M-p` (which means hold down the `Alt` button and press "p"). More generally, the command `history()` will show a whole list of recently entered commands.

- To find out what all variables are in the current work environment, use the commands `objects()` or `ls()` . These list all available objects in the workspace. If you wish to remove one or more variables, use `remove(var1, var2, var3)` , or more simply use `rm(var1, var2, var3)`, and to remove all objects use `rm(list = ls())`.
- Another use of `scan` is when you have a long list of numbers (separated by spaces or on different lines) already typed somewhere else, say in a text file To enter all the data in one fell swoop, first highlight and copy the list of numbers to the Clipboard with `Edit` ▸ `Copy` (or by right-clicking and selecting `Copy`). Next type the `x <- scan()` command in the R console, and paste the numbers at the `1:` prompt with `Edit` ▸ `Paste`. All of the numbers will automatically be entered into the vector `x`.
- The command `Ctrl+l` clears the display in the Microsoft® Windows R Gui. In Emacs/ESS, press `Ctrl+l` repeatedly to cycle point (the place where the cursor is) to the bottom, middle, and top of the display.
- Once you use R for awhile there may be some commands that you wish to run automatically whenever R starts. These commands may be saved in a file called `Rprofile.site` which is usually in the `etc` folder, which lives in the R home directory (which on Microsoft® Windows usually is `C:\Program Files\R`). Alternatively, you can make a file `.Rprofile` to be stored in the user's home directory, or anywhere R is invoked. This allows for multiple configurations for different projects or users. See "Customizing the Environment" of *An Introduction to R* for more details.
- When exiting R the user is given the option to "save the workspace". I recommend that beginners DO NOT save the workspace when quitting. If `Yes` is selected, then all of the objects and data currently in R's memory is saved in a file located in the working directory called `.RData` . This file is then automatically loaded the next time R starts (in which case

R will say `[previously saved workspace restored]`). This is a valuable feature for experienced users of R, but I find that it causes more trouble than it saves with beginners.

2.7 Exercises

Chapter 3

Data Description

In this chapter we introduce the different types of data that a statistician is likely to encounter, and in each subsection we give some examples of how to display the data of that particular type. Once we see how to display data distributions, we next introduce the basic properties of data distributions. We qualitatively explore several data sets. Once that we have intuitive properties of data sets, we next discuss how we may numerically measure and describe those properties with descriptive statistics.

What do I want them to know?

- different data types, such as quantitative versus qualitative, nominal versus ordinal, and discrete versus continuous
- basic graphical displays for assorted data types, and some of their (dis)advantages
- fundamental properties of data distributions, including center, spread, shape, and crazy observations
- methods to describe data (visually/numerically) with respect to the properties, and how the methods differ depending on the data type
- all of the above in the context of grouped data, and in particular, the concept of a factor

3.1 Types of Data

Loosely speaking, a datum is any piece of collected information, and a data set is a collection of data related to each other in some way. We will categorize data into five types and describe each in turn:

- **Quantitative:** data associated with a measurement of some quantity on an observational unit,
- **Qualitative:** data associated with some quality or property of an observational unit,
- **Logical:** data which represent true or false and play an important role later,
- **Missing:** data which should be there but are not, and
- **Other types:** everything else under the sun.

In each subsection we look at some examples of the type in question and introduce methods to display them.

3.1.1 Quantitative data

Quantitative data are any data that measure or are associated with a measurement of the quantity of something. They invariably assume numerical values. Quantitative data can be further subdivided into two categories. * *Discrete data* take values in a finite or countably infinite set of numbers, that is, all possible values could (at least in principle) be written down in an ordered list. Examples include: counts, number of arrivals, or number of successes. They are often represented by integers, say, 0, 1, 2, *etc*. * *Continuous data* take values in an interval of numbers. These are also known as scale data, interval data, or measurement data. Examples include: height, weight, length, time, *etc*. Continuous data are often characterized by fractions or decimals: 3.82, 7.0001, 4 $\frac{5}{8}$, *etc*.

Note that the distinction between discrete and continuous data is not always clear-cut. Sometimes it is convenient to treat data as if they were continuous, even though strictly speaking they are not continuous. See the examples.

Example 3.1 (Annual Precipitation in US Cities)**.** The vector `precip` contains average amount of rainfall (in inches) for each of 70 cities in the United States and Puerto Rico. Let us take a look at the data:

```
str(precip)
```

```
 Named num [1:70] 67 54.7 7 48.5 14 17.2 20.7 13 43.4 40.2 ...
 - attr(*, "names")= chr [1:70] "Mobile" "Juneau" "Phoenix" "Little Rock" ...
```

```
precip[1:4]
```

```
    Mobile     Juneau    Phoenix Little Rock 
      67.0       54.7        7.0       48.5 
```

The output shows that `precip` is a numeric vector which has been *named*, that is, each value has a name associated with it (which can be set with the `names` function). These are quantitative continuous data.

Example 3.2 (Lengths of Major North American Rivers)**.** The U.S. Geological Survey recorded the lengths (in miles) of several rivers in North America. They are stored in the vector `rivers` in the `datasets` package [108] (which ships with base R). See `?rivers`. Let us take a look at the data with the `str` function.

```
str(rivers)
```

```
 num [1:141] 735 320 325 392 524 ...
```

The output says that `rivers` is a numeric vector of length 141, and the first few values are 735, 320, 325, *etc*. These data are definitely quantitative and it appears that the measurements have been rounded to the nearest mile. Thus, strictly speaking, these are discrete data. But we will find it convenient later to take data like these to be continuous for some of our statistical procedures.

Example 3.3 (Yearly Numbers of Important Discoveries)**.** The vector `discoveries` contains numbers of "great" inventions/discoveries in each year from 1860 to 1959, as reported by the 1975 World Almanac. Let us take a look at the data:

```
str(discoveries)
```

```
 Time-Series [1:100] from 1860 to 1959: 5 3 0 2 0 3 2 3 6 1 ...
```

The output is telling us that `discoveries` is a *time series* (see Section 3.1.7 for more) of length 100. The entries are integers, and since they represent counts this is a good example of discrete quantitative data. We will take a closer look in the following sections.

3.1.2 Displaying Quantitative Data

One of the first things to do when confronted by quantitative data (or any data, for that matter) is to make some sort of visual display to gain some insight into the data's structure. There are almost as many display types from which to choose as there are data sets to plot. We describe some of the more popular alternatives.

Strip charts also known as Dot plots

These can be used for discrete or continuous data, and usually look best when the data set is not too large. Along the horizontal axis is a numerical scale above which the data values are plotted. We can do it in R with a call to the `stripchart` function. There are three available methods.

- `overplot`: plots ties covering each other. This method is good to display only the distinct values assumed by the data set.
- `jitter`: adds some noise to the data in the y direction in which case the data values are not covered up by ties.
- `stack`: plots repeated values stacked on top of one another. This method is best used for discrete data with a lot of ties; if there are no repeats then this method is identical to overplot.

See Figure **??**, which was produced by the following code.

```
stripchart(precip, xlab="rainfall")
stripchart(rivers, method="jitter", xlab="length")
stripchart(discoveries, method="stack", xlab="number")
```

The leftmost graph is a strip chart of the `precip` data. The graph shows tightly clustered values in the middle with some others falling balanced on either side, with perhaps slightly more falling to the left. Later we will call this a symmetric distribution, see Section 3.2.3. The middle graph is of the `rivers` data, a vector of length 141. There are several repeated values in the rivers data, and if we were to use the overplot method we would lose some of them in the display. This plot shows a what we will later call a right-skewed shape with perhaps some extreme values on the far right of the display. The third graph strip charts `discoveries` data which are literally a textbook example of a right skewed distribution.

The `DOTplot` function in the `UsingR` package [166] is another alternative.

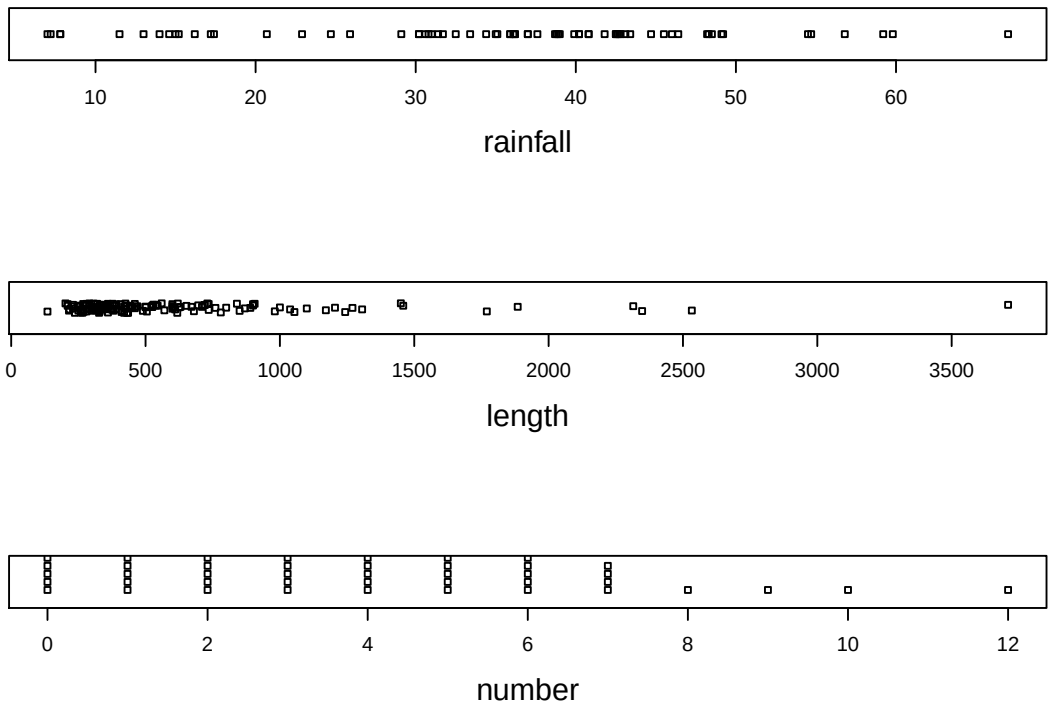

Figure 3.1: Three stripcharts of three data sets. The first graph uses the `overplot` method, the second the `jitter` method, and the third the `stack` method.

Histogram

These are typically used for continuous data. A histogram is constructed by first deciding on a set of classes, or bins, which partition the real line into a set of boxes into which the data values fall. Then vertical bars are drawn over the bins with height proportional to the number of observations that fell into the bin.

These are one of the most common summary displays, and they are often misidentified as "Bar Graphs" (see below.) The scale on the y axis can be frequency, percentage, or density (relative frequency). The term histogram was coined by Karl Pearson in 1891, see [98].

Example 3.4 (Annual Precipitation in US Cities)**.** We are going to take another look at the `precip` data that we investigated earlier. The strip chart in Figure 3.1 suggested a loosely balanced distribution; let us now look to see what a histogram says.

There are many ways to plot histograms in R, and one of the easiest is with the `hist` function. The following code produces the plots in Figure **??**.

```
hist(precip, main = "")
hist(precip, freq = FALSE, main = "")
```

Notice the argument `main = ""` which suppresses the main title from being displayed – it would have said "Histogram of `precip`" otherwise. The plot on the left is a frequency histogram (the default), and the plot on the right is a relative frequency histogram (`freq = FALSE`).

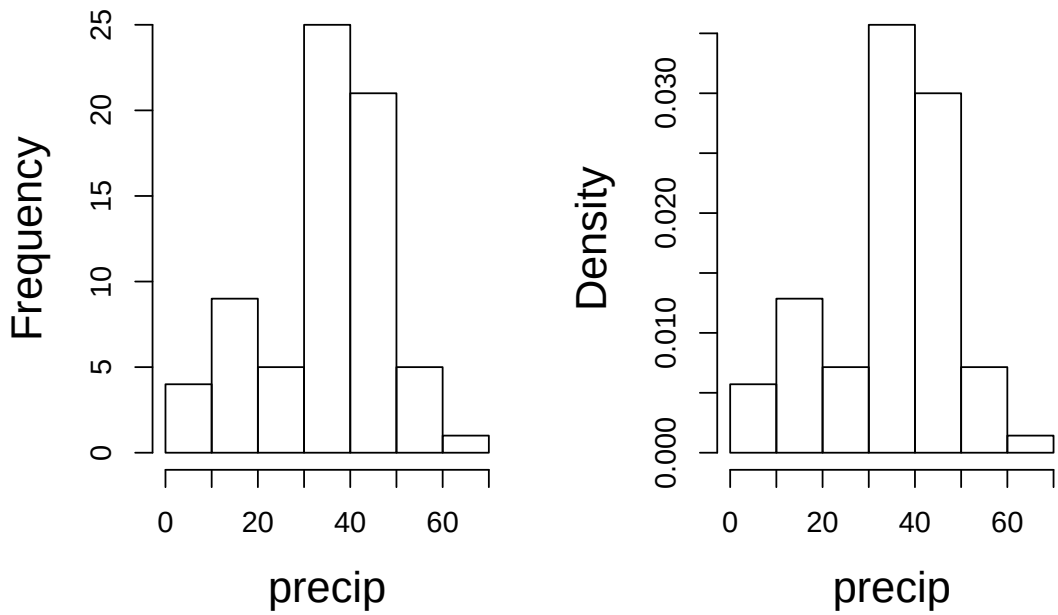

Figure 3.2: (Relative) frequency histograms of the `precip` data.

Please mind the biggest weakness of histograms: the graph obtained strongly depends on the bins chosen. Choose another set of bins, and you will get a different histogram. Moreover, there are not

any definitive criteria by which bins should be defined; the best choice for a given data set is the one which illuminates the data set's underlying structure (if any). Luckily for us there are algorithms to automatically choose bins that are likely to display well, and more often than not the default bins do a good job. This is not always the case, however, and a responsible statistician will investigate many bin choices to test the stability of the display.

Recall that the strip chart in Figure 3.1 suggested a relatively balanced shape to the `precip` data distribution. Watch what happens when we change the bins slightly (with the `breaks` argument to `hist`). See Figure 3.3 which was produced by the following code.

```
hist(precip, breaks = 10)
hist(precip, breaks = 25)
hist(precip, breaks = 50)
```

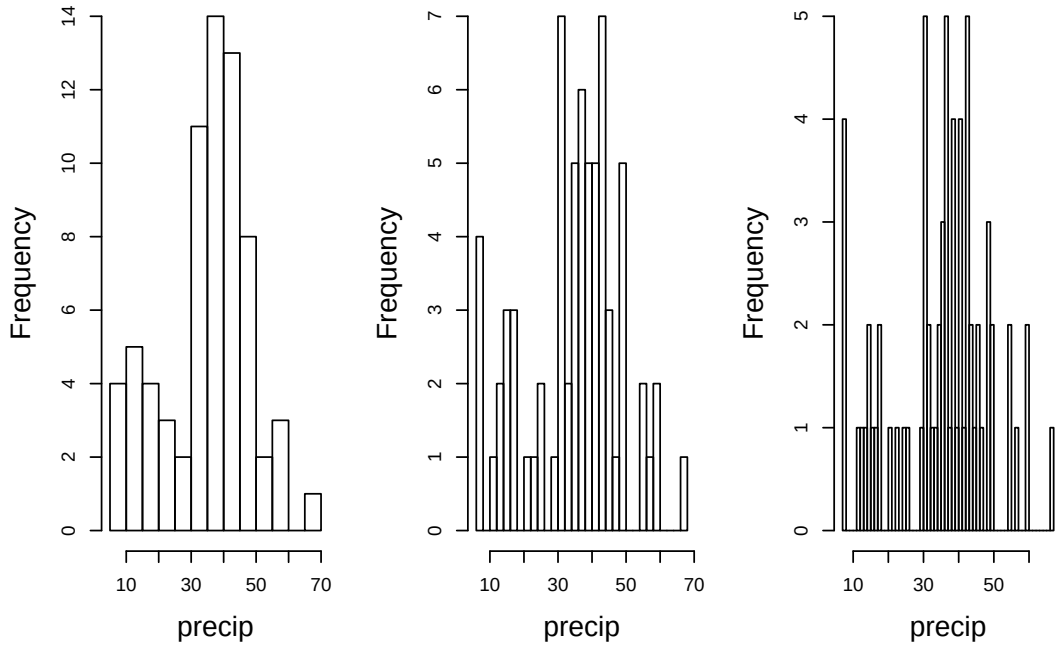

Figure 3.3: More histograms of the `precip` data.

The leftmost graph (with `breaks = 10`) shows that the distribution is not balanced at all. There are two humps: a big one in the middle and a smaller one to the left. Graphs like this often indicate some underlying group structure to the data; we could now investigate whether the cities for which rainfall was measured were similar in some way, with respect to geographic region, for example.

The rightmost graph in Figure 3.3 shows what happens when the number of bins is too large: the histogram is too grainy and hides the rounded appearance of the earlier histograms. If we were to continue increasing the number of bins we would eventually get all observed bins to have exactly one element, which is nothing more than a glorified strip chart.

Stem-and-leaf displays (more to be said in Section 3.4)

Stem-and-leaf displays (also known as stemplots) have two basic parts: *stems* and *leaves*. The final digit of the data values is taken to be a *leaf*, and the leading digit(s) is (are) taken to be *stems*. We draw a vertical line, and to the left of the line we list the stems. To the right of the line, we list the leaves beside their corresponding stem. There will typically be several leaves for each stem, in which case the leaves accumulate to the right. It is sometimes necessary to round the data values, especially for larger data sets.

Example 3.5 (Driver Deaths in the United Kingdom)**.** `UKDriverDeaths` is a time series that contains the total car drivers killed or seriously injured in Great Britain monthly from Jan 1969 to Dec 1984. See `?UKDriverDeaths`. Compulsory seat belt use was introduced on January 31, 1983. We construct a stem and leaf diagram in R with the `stem.leaf` function from the `aplpack` package [178].

```
stem.leaf(UKDriverDeaths, depth = FALSE)
```

```
1 | 2: represents 120
 leaf unit: 10
            n: 192
   10 | 57
   11 | 136678
   12 | 123889
   13 | 0255666888899
   14 | 000012223444445555566667788889
   15 | 000011111222222344445555566677779
   16 | 012223334444455555555678888889
   17 | 11233344566667799
   18 | 00011235568
   19 | 01234455667799
   20 | 0000113557788899
   21 | 145599
   22 | 013467
   23 | 9
   24 | 7
HI: 2654
```

The display shows a more or less balanced mound-shaped distribution, with one or maybe two humps, a big one and a smaller one just to its right. Note that the data have been rounded to the tens place so that each datum gets only one leaf to the right of the dividing line.

Notice that the `depth`s have been suppressed. To learn more about this option and many others, see Section 3.4. Unlike a histogram, the original data values may be recovered from the stem-and-leaf display – modulo the rounding – that is, starting from the top and working down we can read off the data values 1050, 1070, 1110, 1130, and so forth.

Index plots

Done with the `plot` function. These are good for plotting data which are ordered, for example, when the data are measured over time. That is, the first observation was measured at time 1, the second at time 2, *etc*. It is a two dimensional plot, in which the index (or time) is the x variable and the measured value is the y variable. There are several plotting methods for index plots, and we mention two of them:

- `spikes`: draws a vertical line from the x-axis to the observation height.
- `points`: plots a simple point at the observation height.

Example 3.6 (Level of Lake Huron 1875-1972)**.** Brockwell and Davis [17] give the annual measurements of the level (in feet) of Lake Huron from 1875–1972. The data are stored in the time series `LakeHuron`. See `?LakeHuron`. Figure **??** was produced with the following code:

```
plot(LakeHuron)
plot(LakeHuron, type = "p")
plot(LakeHuron, type = "h")
```

The plots show an overall decreasing trend to the observations, and there appears to be some seasonal variation that increases over time.

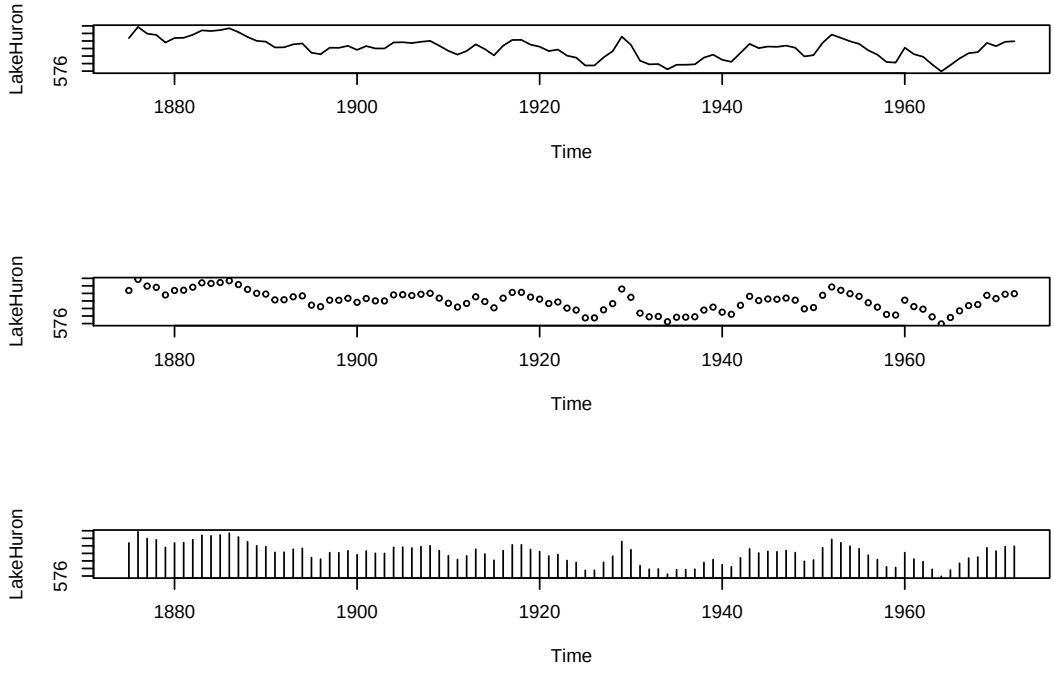

Figure 3.4: Index plots of the `LakeHuron` data.

Density estimates

The default method uses a Gaussian kernel density estimate.

```
# The Old Faithful geyser data
d <- density(faithful$eruptions, bw = "sj")
d
plot(d)
hist(precip, freq = FALSE)
lines(density(precip))
```

3.1.3 Qualitative Data, Categorical Data, and Factors

Qualitative data are simply any type of data that are not numerical, or do not represent numerical quantities. Examples of qualitative variables include a subject's name, gender, race/ethnicity, political party, socioeconomic status, class rank, driver's license number, and social security number (SSN).

Please bear in mind that some data *look* to be quantitative but are *not*, because they do not represent numerical quantities and do not obey mathematical rules. For example, a person's shoe size is typically written with numbers: 8, or 9, or 12, or 12 $\frac{1}{2}$. Shoe size is not quantitative, however, because if we take a size 8 and combine with a size 9 we do not get a size 17.

Some qualitative data serve merely to *identify* the observation (such a subject's name, driver's license number, or SSN). This type of data does not usually play much of a role in statistics. But other qualitative variables serve to *subdivide* the data set into categories; we call these *factors*. In the above examples, gender, race, political party, and socioeconomic status would be considered factors (shoe size would be another one). The possible values of a factor are called its *levels*. For instance, the factor of gender would have two levels, namely, male and female. Socioeconomic status typically has three levels: high, middle, and low.

Factors may be of two types: *nominal* and *ordinal*. Nominal factors have levels that correspond to names of the categories, with no implied ordering. Examples of nominal factors would be hair color, gender, race, or political party. There is no natural ordering to "Democrat" and "Republican"; the categories are just names associated with different groups of people.

In contrast, ordinal factors have some sort of ordered structure to the underlying factor levels. For instance, socioeconomic status would be an ordinal categorical variable because the levels correspond to ranks associated with income, education, and occupation. Another example of ordinal categorical data would be class rank.

Factors have special status in R. They are represented internally by numbers, but even when they are written numerically their values do not convey any numeric meaning or obey any mathematical rules (that is, Stage III cancer is not Stage I cancer + Stage II cancer).

Example 3.7. The `state.abb` vector gives the two letter postal abbreviations for all 50 states.

```
str(state.abb)
```

```
 chr [1:50] "AL" "AK" "AZ" "AR" "CA" "CO" ...
```

These would be ID data. The `state.name` vector lists all of the complete names and those data would also be ID.

Example 3.8 (U.S. State Facts and Features)**.** The U.S. Department of Commerce of the U.S. Census Bureau releases all sorts of information in the *Statistical Abstract of the United States*, and the `state.region` data lists each of the 50 states and the region to which it belongs, be it Northeast, South, North Central, or West. See `?state.region`.

```
str(state.region)
```

```
 Factor w/ 4 levels "Northeast","South",..: 2 4 4 2 4 4 1 2 2 2 ...
```

```
state.region[1:5]
```

```
[1] South West  West  South West
Levels: Northeast South North Central West
```

The `str` output shows that `state.region` is already stored internally as a factor and it lists a couple of the factor levels. To see all of the levels we printed the first five entries of the vector in the second line.

3.1.4 Displaying Qualitative Data

Tables

One of the best ways to summarize qualitative data is with a table of the data values. We may count frequencies with the `table` function or list proportions with the `prop.table` function (whose input is a frequency table). In the R Commander you can do it with `Statistics` ▸ `Frequency Distribution...` Alternatively, to look at tables for all factors in the `Active data set` you can do `Statistics` ▸ `Summaries` ▸ `Active Dataset`.

```
Tbl <- table(state.division)
Tbl
```

```
state.division
       New England    Middle Atlantic     South Atlantic
                 6                  3                  8
East South Central West South Central East North Central
                 4                  4                  5
West North Central           Mountain            Pacific
                 7                  8                  5
```

```
Tbl/sum(Tbl)      # relative frequencies
```

```
state.division
```

```
       New England    Middle Atlantic     South Atlantic
              0.12               0.06               0.16
East South Central West South Central East North Central
              0.08               0.08               0.10
West North Central           Mountain            Pacific
              0.14               0.16               0.10
```

```
prop.table(Tbl)   # same thing
```

```
state.division
       New England    Middle Atlantic     South Atlantic
              0.12               0.06               0.16
East South Central West South Central East North Central
              0.08               0.08               0.10
West North Central           Mountain            Pacific
              0.14               0.16               0.10
```

Bar Graphs

A bar graph is the analogue of a histogram for categorical data. A bar is displayed for each level of a factor, with the heights of the bars proportional to the frequencies of observations falling in the respective categories. A disadvantage of bar graphs is that the levels are ordered alphabetically (by default), which may sometimes obscure patterns in the display.

Example 3.9 (U.S. State Facts and Features)**.** The `state.region` data lists each of the 50 states and the region to which it belongs, be it Northeast, South, North Central, or West. See `?state.region`. It is already stored internally as a factor. We make a bar graph with the `barplot` function:

```
barplot(table(state.region), cex.names = 1.20)
barplot(prop.table(table(state.region)), cex.names = 1.20)
```

See Figure 3.5. The display on the left is a frequency bar graph because the y axis shows counts, while the display on the left is a relative frequency bar graph. The only difference between the two is the scale. Looking at the graph we see that the majority of the fifty states are in the South, followed by West, North Central, and finally Northeast. Over 30% of the states are in the South.

Notice the `cex.names` argument that we used, above. It expands the names on the x axis by 20% which makes them easier to read. See `?par` for a detailed list of additional plot parameters.

Pareto Diagrams

A pareto diagram is a lot like a bar graph except the bars are rearranged such that they decrease in height going from left to right. The rearrangement is handy because it can visually reveal structure (if any) in how fast the bars decrease – this is much more difficult when the bars are jumbled.

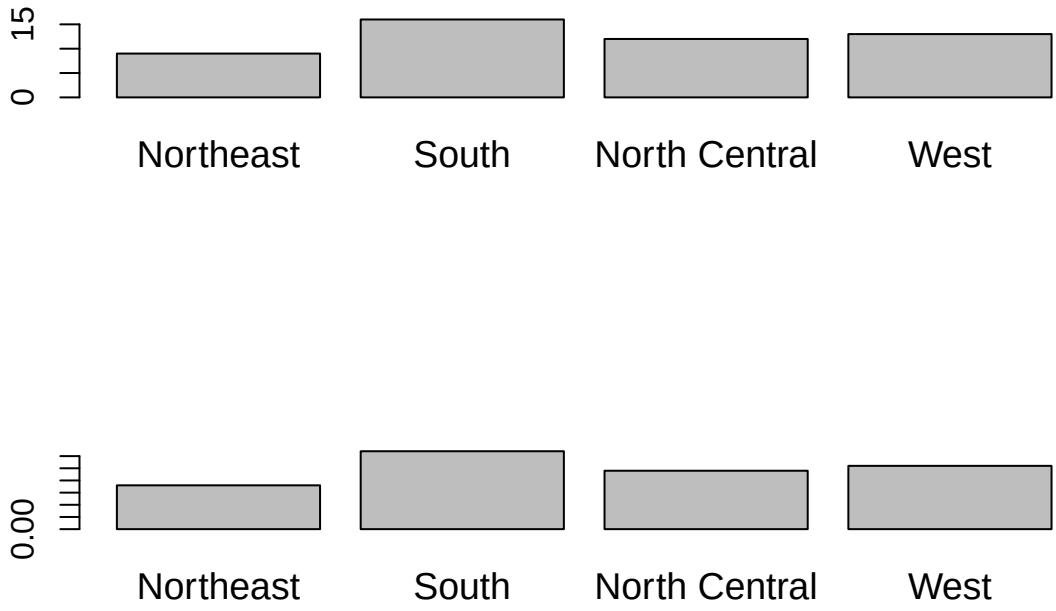

Figure 3.5: The top graph is a frequency barplot made with `table` and the bottom is a relative frequency barplot made with `prop.table`.

Example 3.10 (U.S. State Facts and Features)**.** The `state.division` data record the division (New England, Middle Atlantic, South Atlantic, East South Central, West South Central, East North Central, West North Central, Mountain, and Pacific) of the fifty states. We can make a pareto diagram with either the `RcmdrPlugin.IPSUR` package [152] or with the `pareto.chart` function from the `qcc` package [138]. See Figure 3.6. The code follows.

```
pareto.chart(table(state.division), ylab="Frequency", cex.lab = cexlab)
```

```
Pareto chart analysis for table(state.division)
                     Frequency Cum.Freq. Percentage
  South Atlantic             8         8         16
  Mountain                   8        16         16
  West North Central         7        23         14
  New England                6        29         12
  East North Central         5        34         10
  Pacific                    5        39         10
  East South Central         4        43          8
  West South Central         4        47          8
  Middle Atlantic            3        50          6

Pareto chart analysis for table(state.division)
                     Cum.Percent.
  South Atlantic               16
  Mountain                     32
```

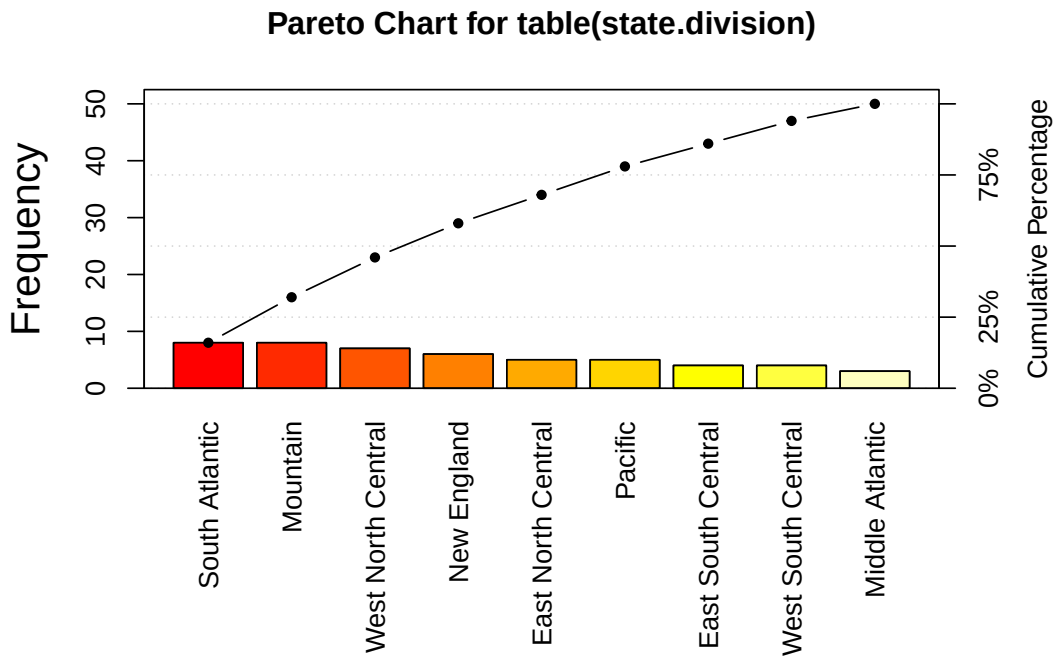

Figure 3.6: Pareto chart of the `state.division` data.

```
West North Central          46
New England                 58
East North Central          68
Pacific                     78
East South Central          86
West South Central          94
Middle Atlantic            100
```

Dot Charts

These are a lot like a bar graph that has been turned on its side with the bars replaced by dots on horizontal lines. They do not convey any more (or less) information than the associated bar graph, but the strength lies in the economy of the display. Dot charts are so compact that it is easy to graph very complicated multi-variable interactions together in one graph. See Section 3.6. We will give an example here using the same data as above for comparison. The graph was produced by the following code.

```
x <- table(state.region)
dotchart(as.vector(x), labels = names(x), cex.lab = cexlab)
```

See Figure 3.7. Compare it to Figure 3.5.

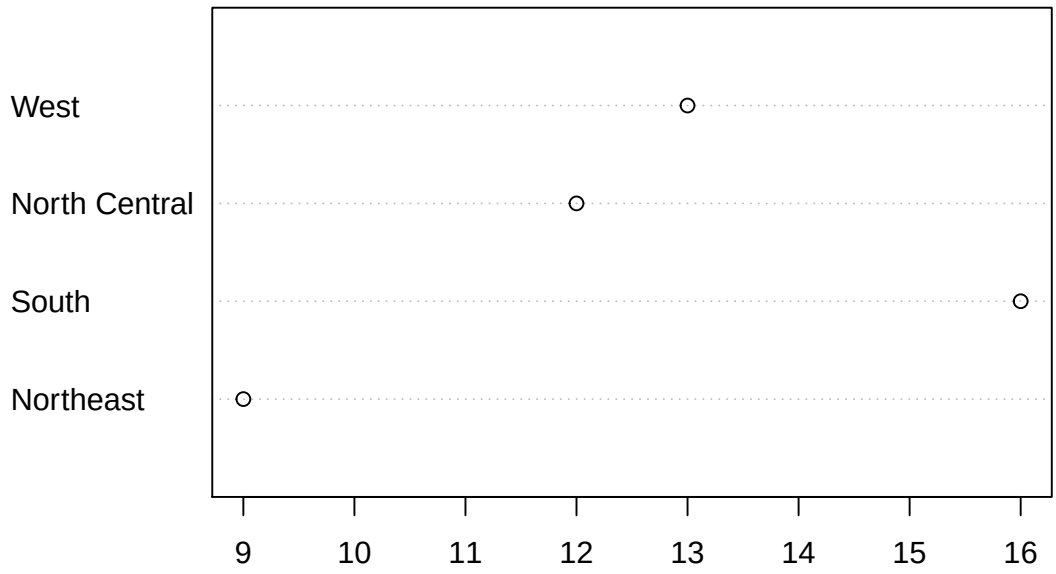

Figure 3.7: Dot chart of the `state.region` data.

Pie Graphs

These can be done with R and the R Commander, but they fallen out of favor in recent years because researchers have determined that while the human eye is good at judging linear measures, it is notoriously bad at judging relative areas (such as those displayed by a pie graph). Pie charts are consequently a very bad way of displaying information unless the number of categories is two or three. A bar chart or dot chart is a preferable way of displaying qualitative data. See `?pie` for more information.

We are not going to do any examples of a pie graph and discourage their use elsewhere.

3.1.5 Logical Data

There is another type of information recognized by R which does not fall into the above categories. The value is either `TRUE` or `FALSE` (note that equivalently you can use `1 = TRUE`, `0 = FALSE`). Here is an example of a logical vector:

```
x <- 5:9
y <- (x < 7.3)
y
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

Many functions in R have options that the user may or may not want to activate in the function call. For example, the `stem.leaf` function has the `depths` argument which is `TRUE` by default. We saw

in Section 3.1.1 how to turn the option off, simply enter `stem.leaf(x, depths = FALSE)` and they will not be shown on the display.

We can swap `TRUE` with `FALSE` with the exclamation point `!`.

```
!y
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE
```

3.1.6 Missing Data

Missing data are a persistent and prevalent problem in many statistical analyses, especially those associated with the social sciences. R reserves the special symbol `NA` to representing missing data.

Ordinary arithmetic with `NA` values give `NA`'s (addition, subtraction, *etc*.) and applying a function to a vector that has an `NA` in it will usually give an `NA`.

```
x <- c(3, 7, NA, 4, 7)
y <- c(5, NA, 1, 2, 2)
x + y
```

```
[1]  8 NA NA  6  9
```

Some functions have a `na.rm` argument which when `TRUE` will ignore missing data as if they were not there (such as `mean`, `var`, `sd`, `IQR`, `mad`, ...).

```
sum(x)
```

```
[1] NA
```

```
sum(x, na.rm = TRUE)
```

```
[1] 21
```

Other functions do not have a `na.rm` argument and will return `NA` or an error if the argument has `NA`s. In those cases we can find the locations of any `NA`s with the `is.na` function and remove those cases with the `[]` operator.

```
is.na(x)
```

```
[1] FALSE FALSE  TRUE FALSE FALSE
```

```
z <- x[!is.na(x)]
sum(z)
```

```
[1] 21
```

The analogue of `is.na` for rectangular data sets (or data frames) is the `complete.cases` function. See Appendix **??**.

3.1.7 Other Data Types

3.2 Features of Data Distributions

Given that the data have been appropriately displayed, the next step is to try to identify salient features represented in the graph. The acronym to remember is *C*-enter, *U*-nusual features, *S*-pread, and *S*-hape. (CUSS).

3.2.1 Center

One of the most basic features of a data set is its center. Loosely speaking, the center of a data set is associated with a number that represents a middle or general tendency of the data. Of course, there are usually several values that would serve as a center, and our later tasks will be focused on choosing an appropriate one for the data at hand. Judging from the histogram that we saw in Figure 3.3, a measure of center would be about 35.

3.2.2 Spread

The spread of a data set is associated with its variability; data sets with a large spread tend to cover a large interval of values, while data sets with small spread tend to cluster tightly around a central value.

3.2.3 Shape

When we speak of the *shape* of a data set, we are usually referring to the shape exhibited by an associated graphical display, such as a histogram. The shape can tell us a lot about any underlying structure to the data, and can help us decide which statistical procedure we should use to analyze them.

Symmetry and Skewness

A distribution is said to be *right-skewed* (or *positively skewed*) if the right tail seems to be stretched from the center. A *left-skewed* (or *negatively skewed*) distribution is stretched to the left side. A symmetric distribution has a graph that is balanced about its center, in the sense that half of the graph may be reflected about a central line of symmetry to match the other half.

We have already encountered skewed distributions: both the discoveries data in Figure 3.1 and the `precip` data in Figure 3.3 appear right-skewed. The `UKDriverDeaths` data in Example 3.5 is relatively symmetric (but note the one extreme value 2654 identified at the bottom of the stem-and-leaf display).

Kurtosis

Another component to the shape of a distribution is how "peaked" it is. Some distributions tend to have a flat shape with thin tails. These are called *platykurtic*, and an example of a platykurtic distribution is the uniform distribution; see Section **??**. On the other end of the spectrum are distributions with a steep peak, or spike, accompanied by heavy tails; these are called *leptokurtic*. Examples of leptokurtic distributions are the Laplace distribution and the logistic distribution. See Section **??**. In between are distributions (called *mesokurtic*) with a rounded peak and moderately sized tails. The standard example of a mesokurtic distribution is the famous bell-shaped curve, also known as the Gaussian, or normal, distribution, and the binomial distribution can be mesokurtic for specific choices of p. See Sections **??** and **??**.

3.2.4 Clusters and Gaps

Clusters or gaps are sometimes observed in quantitative data distributions. They indicate clumping of the data about distinct values, and gaps may exist between clusters. Clusters often suggest an underlying grouping to the data. For example, take a look at the `faithful` data which contains the duration of `eruptions` and the `waiting` time between eruptions of the Old Faithful geyser in Yellowstone National Park. Do not be frightened by the complicated information at the left of the display for now; we will learn how to interpret it in Section 3.4.

```
with(faithful, stem.leaf(eruptions))
```

```
1 | 2: represents 1.2
 leaf unit: 0.1
            n: 272
   12     s | 667777777777
   51    1. | 88888888888888888888888888888999999999999
   71    2* | 00000000000011111111
   87     t | 2222222222333333
   92     f | 44444
   94     s | 66
   97    2. | 889
   98    3* | 0
  102     t | 3333
  108     f | 445555
  118     s | 6666677777
  (16)   3. | 8888888889999999
  138    4* | 00000000000000001111111111111111
  107     t | 22222222222233333333333333333
   78     f | 44444444444445555555555555555555
   43     s | 6666666666677777777777
   21    4. | 88888888888899999
    4    5* | 0001
```

There are definitely two clusters of data here; an upper cluster and a lower cluster.

3.2.5 Extreme Observations and other Unusual Features

Extreme observations fall far from the rest of the data. Such observations are troublesome to many statistical procedures; they cause exaggerated estimates and instability. It is important to identify extreme observations and examine the source of the data more closely. There are many possible reasons underlying an extreme observation:

- *Maybe the value is a typographical error.* Especially with large data sets becoming more prevalent, many of which being recorded by hand, mistakes are a common problem. After closer scrutiny, these can often be fixed.
- *Maybe the observation was not meant for the study*, because it does not belong to the population of interest. For example, in medical research some subjects may have relevant complications in their genealogical history that would rule out their participation in the experiment. Or when a manufacturing company investigates the properties of one of its devices, perhaps a particular product is malfunctioning and is not representative of the majority of the items.
- *Maybe it indicates a deeper trend or phenomenon.* Many of the most influential scientific discoveries were made when the investigator noticed an unexpected result, a value that was not predicted by the classical theory. Albert Einstein, Louis Pasteur, and others built their careers on exactly this circumstance.

3.3 Descriptive Statistics

One of my favorite professors would repeatedly harp, "You cannot do statistics without data."

What do I want them to know?

- The fundamental data types we encounter most often, how to classify given data into a likely type, and that sometimes the distinction is blurry.

3.3.1 Frequencies and Relative Frequencies

These are used for categorical data. The idea is that there are a number of different categories, and we would like to get some idea about how the categories are represented in the population.

3.3.2 Measures of Center

The *sample mean* is denoted $\overline{x}$ (read "x-bar") and is simply the arithmetic average of the observations:

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{3.1}$$

- Good: natural, easy to compute, has nice mathematical properties
- Bad: sensitive to extreme values

It is appropriate for use with data sets that are not highly skewed without extreme observations.

The *sample median* is another popular measure of center and is denoted $\tilde{x}$. To calculate its value, first sort the data into an increasing sequence of numbers. If the data set has an odd number of observations then $\tilde{x}$ is the value of the middle observation, which lies in position $(n+1)/2$; otherwise, there are two middle observations and $\tilde{x}$ is the average of those middle values.

- Good: resistant to extreme values, easy to describe
- Bad: not as mathematically tractable, need to sort the data to calculate

One desirable property of the sample median is that it is *resistant* to extreme observations, in the sense that the value of $\tilde{x}$ depends only on those data values in the middle, and is quite unaffected by the actual values of the outer observations in the ordered list. The same cannot be said for the sample mean. Any significant changes in the magnitude of an observation x_k results in a corresponding change in the value of the mean. Consequently, the sample mean is said to be *sensitive* to extreme observations.

The *trimmed mean* is a measure designed to address the sensitivity of the sample mean to extreme observations. The idea is to "trim" a fraction (less than 1/2) of the observations off each end of the ordered list, and then calculate the sample mean of what remains. We will denote it by $\overline{x}_{t=0.05}$.

- Good: resistant to extreme values, shares nice statistical properties
- Bad: need to sort the data

How to do it with R

- You can calculate frequencies or relative frequencies with the `table` function, and relative frequencies with `prop.table(table())`.
- You can calculate the sample mean of a data vector `x` with the command `mean(x)`.
- You can calculate the sample median of `x` with the command `median(x)`.
- You can calculate the trimmed mean with the `trim` argument; `mean(x, trim = 0.05)`.

3.3.3 Order Statistics and the Sample Quantiles

A common first step in an analysis of a data set is to sort the values. Given a data set $x_1, x_2, \ldots, x_n$, we may sort the values to obtain an increasing sequence

$$x_{(1)} \leq x_{(2)} \leq x_{(3)} \leq \cdots \leq x_{(n)} \tag{3.2}$$

and the resulting values are called the *order statistics*. The k^{th} entry in the list, $x_{(k)}$, is the k^{th} order statistic, and approximately $100(k/n)\%$ of the observations fall below $x_{(k)}$. The order statistics give an indication of the shape of the data distribution, in the sense that a person can look at the order statistics and have an idea about where the data are concentrated, and where they are sparse.

The *sample quantiles* are related to the order statistics. Unfortunately, there is not a universally accepted definition of them. Indeed, R is equipped to calculate quantiles using nine distinct

definitions! We will describe the default method (`type = 7`), but the interested reader can see the details for the other methods with `?quantile`.

Suppose the data set has n observations. Find the sample quantile of order p ($0 < p < 1$), denoted $\tilde{q}_p$, as follows:

- **First step:** sort the data to obtain the order statistics $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$.
- **Second step:** calculate $(n-1)p + 1$ and write it in the form $k.d$, where k is an integer and d is a decimal.
- **Third step:** The sample quantile $\tilde{q}_p$ is

$$\tilde{q}_p = x_{(k)} + d(x_{(k+1)} - x_{(k)}). \tag{3.3}$$

The interpretation of $\tilde{q}_p$ is that approximately $100p$ % of the data fall below the value $\tilde{q}_p$.

Keep in mind that there is not a unique definition of percentiles, quartiles, *etc*. Open a different book, and you'll find a different definition. The difference is small and seldom plays a role except in small data sets with repeated values. In fact, most people do not even notice in common use.

Clearly, the most popular sample quantile is $\tilde{q}_{0.50}$, also known as the sample median, $\tilde{x}$. The closest runners-up are the *first quartile* $\tilde{q}_{0.25}$ and the *third quartile* $\tilde{q}_{0.75}$ (the *second quartile* is the median).

How to do it with R

At the command prompt We can find the order statistics of a data set stored in a vector `x` with the command `sort(x)`.

We can calculate the sample quantiles of any order p where $0 < p < 1$ for a data set stored in a data vector `x` with the `quantile` function, for instance, the command `quantile(x, probs = c(0, 0.25, 0.37))` will return the smallest observation, the first quartile, $\tilde{q}_{0.25}$, and the 37th sample quantile, $\tilde{q}_{0.37}$. For $\tilde{q}_p$ simply change the values in the `probs` argument to the value p.

With the R Commander we can find the order statistics of a variable in the `Active data set` by doing `Data` ▹ `Manage variables in Active data set...` ▹ `Compute new variable...` In the `Expression to compute` dialog simply type `sort(varname)`, where `varname` is the variable that it is desired to sort.

In `Rcmdr`, we can calculate the sample quantiles for a particular variable with the sequence `Statistics` ▹ `Summaries` ▹ `Numerical Summaries...` We can automatically calculate the quartiles for all variables in the `Active data set` with the sequence `Statistics` ▹ `Summaries` ▹ `Active Dataset`.

3.3.4 Measures of Spread

Sample Variance and Standard Deviation

The *sample variance* is denoted s^2 and is calculated with the formula

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2. \tag{3.4}$$

The *sample standard deviation* is $s = \sqrt{s^2}$. Intuitively, the sample variance is approximately the average squared distance of the observations from the sample mean. The sample standard deviation is used to scale the estimate back to the measurement units of the original data.

- Good: tractable, has nice mathematical/statistical properties
- Bad: sensitive to extreme values

We will spend a lot of time with the variance and standard deviation in the coming chapters. In the meantime, the following two rules give some meaning to the standard deviation, in that there are bounds on how much of the data can fall past a certain distance from the mean.

Fact: (Chebychev's Rule). The proportion of observations within k standard deviations of the mean is at least $1 - 1/k^2$, *i.e.*, at least 75%, 89%, and 94% of the data are within 2, 3, and 4 standard deviations of the mean, respectively.

Note that Chebychev's Rule does not say anything about when $k = 1$, because $1 - 1/1^2 = 0$, which states that at least 0% of the observations are within one standard deviation of the mean (which is not saying much).

Chebychev's Rule applies to any data distribution, *any* list of numbers, no matter where it came from or what the histogram looks like. The price for such generality is that the bounds are not very tight; if we know more about how the data are shaped then we can say more about how much of the data can fall a given distance from the mean.

Fact: (Empirical Rule). If data follow a bell-shaped curve, then approximately 68%, 95%, and 99.7% of the data are within 1, 2, and 3 standard deviations of the mean, respectively.

Interquartile Range

Just as the sample mean is sensitive to extreme values, so the associated measure of spread is similarly sensitive to extremes. Further, the problem is exacerbated by the fact that the extreme distances are squared. We know that the sample quartiles are resistant to extremes, and a measure of spread associated with them is the *interquartile range* (IQR) defined by $IQR = q_{0.75} - q_{0.25}$.

- Good: stable, resistant to outliers, robust to nonnormality, easy to explain
- Bad: not as tractable, need to sort the data, only involves the middle 50% of the data.

Median Absolute Deviation

A measure even more robust than the IQR is the *median absolute deviation* (MAD). To calculate it we first get the median $\widetilde{x}$, next the *absolute deviations* $|x_1 - \tilde{x}|, |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|$, and the MAD is proportional to the median of those deviations:

$$MAD \propto \text{median}(|x_1 - \tilde{x}|,\ |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|). \tag{3.5}$$

That is, the $MAD = c \cdot \text{median}(|x_1 - \tilde{x}|,\ |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|)$, where c is a constant chosen so that the MAD has nice properties. The value of c in R is by default $c = 1.4286$. This value is chosen to ensure that the estimator of σ is correct, on the average, under suitable sampling assumptions (see Section **??**).

- Good: stable, very robust, even more so than the IQR.
- Bad: not tractable, not well known and less easy to explain.

Comparing Apples to Apples

We have seen three different measures of spread which, for a given data set, will give three different answers. Which one should we use? It depends on the data set. If the data are well behaved, with an approximate bell-shaped distribution, then the sample mean and sample standard deviation are natural choices with nice mathematical properties. However, if the data have an unusual or skewed shape with several extreme values, perhaps the more resistant choices among the IQR or MAD would be more appropriate.

However, once we are looking at the three numbers it is important to understand that the estimators are not all measuring the same quantity, on the average. In particular, it can be shown that when the data follow an approximately bell-shaped distribution, then on the average, the sample standard deviation s and the MAD will be the approximately the same value, namely, σ, but the IQR will be on the average 1.349 times larger than s and the MAD. See **??** for more details.

How to do it with R

At the command prompt we may compute the sample range with `range(x)` and the sample variance with `var(x)`, where `x` is a numeric vector. The sample standard deviation is `sqrt(var(x))` or just `sd(x)`. The IQR is `IQR(x)` and the median absolute deviation is `mad(x)`.

With the R Commander we can calculate the sample standard deviation with the `Statistics` ▸ `Summaries` ▸ `Numerical Summaries...` combination. R Commander does not calculate the IQR or MAD in any of the menu selections, by default.

3.3.5 Measures of Shape

Sample Skewness

The *sample skewness*, denoted by g_1, is defined by the formula

$$g_1 = \frac{1}{n} \frac{\sum_{i=1}^{n} (x_i - \overline{x})^3}{s^3}. \tag{3.6}$$

The sample skewness can be any value $-\infty < g_1 < \infty$. The sign of g_1 indicates the direction of skewness of the distribution. Samples that have $g_1 > 0$ indicate right-skewed distributions (or positively skewed), and samples with $g_1 < 0$ indicate left-skewed distributions (or negatively skewed). Values of g_1 near zero indicate a symmetric distribution. These are not hard and fast rules, however. The value of g_1 is subject to sampling variability and thus only provides a suggestion to the skewness of the underlying distribution.

We still need to know how big is "big", that is, how do we judge whether an observed value of g_1 is far enough away from zero for the data set to be considered skewed to the right or left? A good rule of thumb is that data sets with skewness larger than $2\sqrt{6/n}$ in magnitude are substantially skewed, in the direction of the sign of g_1. See Tabachnick & Fidell [149] for details.

Sample Excess Kurtosis

The *sample excess kurtosis*, denoted by g_2, is given by the formula

$$g_2 = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \overline{x})^4}{s^4} - 3. \tag{3.7}$$

The sample excess kurtosis takes values $-2 \leq g_2 < \infty$. The subtraction of 3 may seem mysterious but it is done so that mound shaped samples have values of g_2 near zero. Samples with $g_2 > 0$ are called *leptokurtic*, and samples with $g_2 < 0$ are called *platykurtic*. Samples with $g_2 \approx 0$ are called *mesokurtic*.

As a rule of thumb, if $|g_2| > 4\sqrt{6/n}$ then the sample excess kurtosis is substantially different from zero in the direction of the sign of g_2. See Tabachnick & Fidell [149] for details.

Notice that both the sample skewness and the sample kurtosis are invariant with respect to location and scale, that is, the values of g_1 and g_2 do not depend on the measurement units of the data.

How to do it with R

The `e1071` package [96] has the `skewness` function for the sample skewness and the `kurtosis` function for the sample excess kurtosis. Both functions have a `na.rm` argument which is `FALSE` by default.

Example 3.11. We said earlier that the `discoveries` data looked positively skewed; let's see what the statistics say:

```
skewness(discoveries)
```

```
[1] 1.2076
```

```
2*sqrt(6/length(discoveries))
```

```
[1] 0.4898979
```

The data are definitely skewed to the right. Let us check the sample excess kurtosis of the UKDriverDeaths data:

```
kurtosis(UKDriverDeaths)
```

```
[1] 0.07133848
```

```
4*sqrt(6/length(UKDriverDeaths))
```

```
[1] 0.7071068
```

so that the UKDriverDeaths data appear to be mesokurtic, or at least not substantially leptokurtic.

3.4 Exploratory Data Analysis

This field was founded (mostly) by John Tukey (1915-2000). Its tools are useful when not much is known regarding the underlying causes associated with the data set, and are often used for checking assumptions. For example, suppose we perform an experiment and collect some data... now what? We look at the data using exploratory visual tools.

3.4.1 More About Stem-and-leaf Displays

There are many bells and whistles associated with stemplots, and the `stem.leaf` function can do many of them.

- **Trim Outliers:** Some data sets have observations that fall far from the bulk of the other data (in a sense made more precise in Section 3.4.4). These extreme observations often obscure the underlying structure to the data and are best left out of the data display. The `trim.outliers` argument (which is `TRUE` by default) will separate the extreme observations from the others and graph the stemplot without them; they are listed at the bottom (respectively, top) of the stemplot with the label `HI` (respectively `LO`).
- **Split Stems:** The standard stemplot has only one line per stem, which means that all observations with first digit `3` are plotted on the same line, regardless of the value of the second digit. But this gives some stemplots a "skyscraper" appearance, with too many observations stacked onto the same stem. We can often fix the display by increasing the number of lines available for a given stem. For example, we could make two lines per stem, say, `3*` and `3.`. Observations with second digit 0 through 4 would go on the upper line, while observations with second digit 5 through 9 would go on the lower line. (We could do a similar thing with five lines per stem, or even ten lines per stem.) The end result is a more spread out stemplot which often looks better. A good example of this was shown on page **??**.
- **Depths:** these are used to give insight into the balance of the observations as they accumulate toward the median. In a column beside the standard stemplot, the frequency of the stem containing the sample median is shown in parentheses. Next, frequencies are accumulated from the outside inward, including the outliers. Distributions that are more symmetric will have better balanced depths on either side of the sample median.

How to do it with R

The basic command is `stem(x)` or a more sophisticated version written by Peter Wolf called `stem.leaf(x)` in the R Commander. We will describe `stem.leaf` since that is the one used by R Commander.

WARNING: Sometimes when making a stem-and-leaf display the result will not be what you expected. There are several reasons for this:

- Stemplots by default will trim extreme observations (defined in Section 3.4.4) from the display. This in some cases will result in stemplots that are not as wide as expected.
- The leafs digit is chosen automatically by `stem.leaf` according to an algorithm that the computer believes will represent the data well. Depending on the choice of the digit, `stem.leaf` may drop digits from the data or round the values in unexpected ways.

Let us take a look at the `rivers` data set.

```
stem.leaf(rivers)
```

```
1 | 2: represents 120
 leaf unit: 10
            n: 141
    1     1 | 3
   29     2 | 01111333345555566666778888899
   64     3 | 000001111222233334555556666778888999
  (18)    4 | 011222233344566679
   59     5 | 000222234467
   47     6 | 0000112235789
   34     7 | 12233368
   26     8 | 04579
   21     9 | 0008
   17    10 | 035
   14    11 | 07
   12    12 | 047
    9    13 | 0
HI: 1450 1459 1770 1885 2315 2348 2533 3710
```

The stem-and-leaf display shows a right-skewed shape to the `rivers` data distribution. Notice that the last digit of each of the data values were dropped from the display. Notice also that there were eight extreme observations identified by the computer, and their exact values are listed at the bottom of the stemplot. Look at the scale on the left of the stemplot and try to imagine how ridiculous the graph would have looked had we tried to include enough stems to include these other eight observations; the stemplot would have stretched over several pages. Notice finally that we can use the depths to approximate the sample median for these data. The median lies in the row identified by `(18)`, which means that the median is the average of the ninth and tenth observation on that row. Those two values correspond to `43` and `43`, so a good guess for the median would be 430. (For the record, the sample median is $\widetilde{x} = 425$. Recall that stemplots round the data to the nearest stem-leaf pair.)

Next let us see what the `precip` data look like.

```
stem.leaf(precip)
```

```
1 | 2: represents 12
 leaf unit: 1
            n: 70
LO: 7 7.2 7.8 7.8
    8    1* | 1344
   13    1. | 55677
   16    2* | 024
   18    2. | 59
   28    3* | 0000111234
  (15)   3. | 555566677788899
   27    4* | 0000122222334
   14    4. | 56688899
    6    5* | 44
    4    5. | 699
HI: 67
```

Here is an example of split stems, with two lines per stem. The final digit of each datum has been dropped for the display. The data appear to be left skewed with four extreme values to the left and one extreme value to the right. The sample median is approximately 37 (it turns out to be 36.6).

3.4.2 Hinges and the Five Number Summary

Given a data set $x_1, x_2, \ldots, x_n$, the hinges are found by the following method:

- Find the order statistics $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$.
- The *lower hinge* h_L is in position $L = \lfloor (n+3)/2 \rfloor / 2$, where the symbol $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. If the position L is not an integer, then the hinge h_L is the average of the adjacent order statistics.
- The *upper hinge* h_U is in position $n + 1 - L$.

Given the hinges, the *five number summary* ($5NS$) is

$$5NS = (x_{(1)},\ h_L,\ \tilde{x},\ h_U,\ x_{(n)}). \tag{3.8}$$

An advantage of the $5NS$ is that it reduces a potentially large data set to a shorter list of only five numbers, and further, these numbers give insight regarding the shape of the data distribution similar to the sample quantiles in Section 3.3.3.

How to do it with R

If the data are stored in a vector `x`, then you can compute the $5NS$ with the `fivenum` function.

3.4.3 Boxplots

A boxplot is essentially a graphical representation of the $5NS$. It can be a handy alternative to a stripchart when the sample size is large.

A boxplot is constructed by drawing a box alongside the data axis with sides located at the upper and lower hinges. A line is drawn parallel to the sides to denote the sample median. Lastly, whiskers are extended from the sides of the box to the maximum and minimum data values (more precisely, to the most extreme values that are not potential outliers, defined below).

Boxplots are good for quick visual summaries of data sets, and the relative positions of the values in the $5NS$ are good at indicating the underlying shape of the data distribution, although perhaps not as effectively as a histogram. Perhaps the greatest advantage of a boxplot is that it can help to objectively identify extreme observations in the data set as described in the next section.

Boxplots are also good because one can visually assess multiple features of the data set simultaneously:

- **Center:** can be estimated by the sample median, $\tilde{x}$.
- **Spread:** can be judged by the width of the box, $h_U - h_L$. We know that this will be close to the IQR, which can be compared to s and the MAD, perhaps after rescaling if appropriate.
- **Shape:** is indicated by the relative lengths of the whiskers, and the position of the median inside the box. Boxes with unbalanced whiskers indicate skewness in the direction of the long whisker. Skewed distributions often have the median tending in the opposite direction of skewness. Kurtosis can be assessed using the box and whiskers. A wide box with short whiskers will tend to be platykurtic, while a skinny box with wide whiskers indicates leptokurtic distributions.
- **Extreme observations:** are identified with open circles (see below).

3.4.4 Outliers

A *potential outlier* is any observation that falls beyond 1.5 times the width of the box on either side, that is, any observation less than $h_L - 1.5(h_U - h_L)$ or greater than $h_U + 1.5(h_U - h_L)$. A *suspected outlier* is any observation that falls beyond 3 times the width of the box on either side. In R, both potential and suspected outliers (if present) are denoted by open circles; there is no distinction between the two.

When potential outliers are present, the whiskers of the boxplot are then shortened to extend to the most extreme observation that is not a potential outlier. If an outlier is displayed in a boxplot, the index of the observation may be identified in a subsequent plot in `Rcmdr` by clicking the `Identify outliers with mouse` option in the `Boxplot` dialog.

What do we do about outliers? They merit further investigation. The primary goal is to determine why the observation is outlying, if possible. If the observation is a typographical error, then it should be corrected before continuing. If the observation is from a subject that does not belong to the population of interest, then perhaps the datum should be removed. Otherwise, perhaps the value is hinting at some hidden structure to the data.

How to do it with R

The quickest way to visually identify outliers is with a boxplot, described above. Another way is with the `boxplot.stats` function.

Example 3.12 (Lengths of Major North American Rivers)**.** We will look for potential outliers in the `rivers` data.

```
boxplot.stats(rivers)$out
```

```
 [1] 1459 1450 1243 2348 3710 2315 2533 1306 1270 1885 1770
```

We may change the `coef` argument to 3 (it is 1.5 by default) to identify suspected outliers.

```
boxplot.stats(rivers, coef = 3)$out
```

```
[1] 2348 3710 2315 2533 1885
```

3.4.5 Standardizing variables

It is sometimes useful to compare data sets with each other on a scale that is independent of the measurement units. Given a set of observed data $x_1, x_2, \ldots, x_n$ we get z scores, denoted $z_1, z_2, \ldots, z_n$, by means of the following formula

$$z_i = \frac{x_i - \overline{x}}{s}, \quad i = 1, 2, \ldots, n.$$

How to do it with R

The `scale` function will rescale a numeric vector (or data frame) by subtracting the sample mean from each value (column) and/or by dividing each observation by the sample standard deviation.

3.5 Multivariate Data and Data Frames

We have had experience with vectors of data, which are long lists of numbers. Typically, each entry in the vector is a single measurement on a subject or experimental unit in the study. We saw in Section **??** how to form vectors with the `c` function or the `scan` function.

However, statistical studies often involve experiments where there are two (or more) measurements associated with each subject. We display the measured information in a rectangular array in which each row corresponds to a subject, and the columns contain the measurements for each respective variable. For instance, if one were to measure the height and weight and hair color of each of 11 persons in a research study, the information could be represented with a rectangular array. There would be 11 rows. Each row would have the person's height in the first column and hair color in the second column.

The corresponding objects in R are called *data frames*, and they can be constructed with the `data.frame` function. Each row is an observation, and each column is a variable.

Example 3.13. Suppose we have two vectors `x` and `y` and we want to make a data frame out of them.

```
x <- 5:8
y <- letters[3:6]
A <- data.frame(v1 = x, v2 = y)
```

Notice that `x` and `y` are the same length. This is *necessary*. Also notice that `x` is a numeric vector and `y` is a character vector. We may choose numeric and character vectors (or even factors) for the columns of the data frame, but each column must be of exactly one type. That is, we can have a column for `height` and a column for `gender`, but we will get an error if we try to mix function `height` (numeric) and `gender` (character or factor) information in the same column.

Indexing of data frames is similar to indexing of vectors. To get the entry in row i and column j do `A[i,j]`. We can get entire rows and columns by omitting the other index.

```
A[3, ]
```

```
  v1 v2
3  7  e
```

```
A[ , 1]
```

```
[1] 5 6 7 8
```

```
A[ , 2]
```

```
[1] c d e f
Levels: c d e f
```

There are several things happening above. Notice that `A[3, ]` gave a data frame (with the same entries as the third row of `A`) yet `A[ , 1]` is a numeric vector. `A[ ,2]` is a factor vector because the default setting for `data.frame` is `stringsAsFactors = TRUE`.

Data frames have a `names` attribute and the names may be extracted with the `names` function. Once we have the names we may extract given columns by way of the dollar sign.

```
names(A)
```

```
[1] "v1" "v2"
```

```
A['v1']
```

```
  v1
1  5
2  6
3  7
4  8
```

The above is identical to `A[ ,1]`.

3.5.1 Bivariate Data

- Stacked bar charts
- odds ratio and relative risk
- Introduce the sample correlation coefficient.

The *sample Pearson product-moment correlation coefficient*:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})}}$$

- independent of scale
- $-1 < r < 1$
- measures *strength* and *direction* of linear association
- Two-Way Tables. Done with `table`, or in the R Commander by following `Statistics` ▸ `Contingency Tables` ▸} `Two-way Tables`. You can also enter and analyze a two-way table.
- table
- prop.table
- addmargins
- rowPercents (Rcmdr)
- colPercents (Rcmdr)
- totPercents(Rcmdr)
- A <- xtabs(~ gender + race, data = RcmdrTestDrive)
- xtabs(Freq ~ Class + Sex, data = Titanic) from built in table
- barplot(A, legend.text=TRUE)
- barplot(t(A), legend.text=TRUE)
- barplot(A, legend.text=TRUE, beside = TRUE)
- spineplot(gender ~ race, data = RcmdrTestDrive)
- Spine plot: plots categorical versus categorical
- Scatterplot: look for linear association and correlation.
- carb ~ optden, data = Formaldehyde (boring)
- conc ~ rate, data = Puromycin
- xyplot(accel ~ dist, data = attenu) nonlinear association
- xyplot(eruptions ~ waiting, data = faithful) (linear, two groups)
- xyplot(Petal.Width ~ Petal.Length, data = iris)

- xyplot(pressure ~ temperature, data = pressure) (exponential growth)
- xyplot(weight ~ height, data = women) (strong positive linear)

3.5.2 Multivariate Data

Multivariate Data Display

- Multi-Way Tables. You can do this with `table`, or in R Commander by following `Statistics` ▸ `Contingency Tables` ▸ `Multi-way Tables`.
- Scatterplot matrix. used for displaying pairwise scatterplots simultaneously. Again, look for linear association and correlation.
- 3D Scatterplot. See Figure 6.2
- `plot(state.region, state.division)`
- `barplot(table(state.division,state.region), legend.text=TRUE)`

```
require(graphics)
mosaicplot(HairEyeColor)
x <- apply(HairEyeColor, c(1, 2), sum)
x
mosaicplot(x, main = "Relation between hair and eye color")
y <- apply(HairEyeColor, c(1, 3), sum)
y
mosaicplot(y, main = "Relation between hair color and sex")
z <- apply(HairEyeColor, c(2, 3), sum)
z
mosaicplot(z, main = "Relation between eye color and sex")
```

3.6 Comparing Populations

Sometimes we have data from two or more groups (or populations) and we would like to compare them and draw conclusions. Some issues that we would like to address:

- Comparing centers and spreads: variation within versus between groups
- Comparing clusters and gaps
- Comparing outliers and unusual features
- Comparing shapes.

3.6.1 Numerically

I am thinking here about the `Statistics` ▸ `Numerical Summaries` ▸ `Summarize by groups` option or the `Statistics` ▸ `Summaries` ▸ `Table of Statistics` option.

3.6.2 Graphically

- Boxplots
- Variable width: the width of the drawn boxplots are proportional to $\sqrt{n_i}$, where n_i is the size of the i^{th} group. Why? Because many statistics have variability proportional to the reciprocal of the square root of the sample size.
- Notches: extend to $1.58 \cdot (h_U - h_L)/\sqrt{n}$. The idea is to give roughly a 95% confidence interval for the difference in two medians. See Chapter **??**.
- Stripcharts
- stripchart(weight ~ feed, method= "stack", data=chickwts)
- Bar Graphs
- barplot(xtabs(Freq ~ Admit + Gender, data = UCBAdmissions)) stacked bar chart
- barplot(xtabs(Freq ~ Admit, data = UCBAdmissions))
- barplot(xtabs(Freq ~ Gender + Admit, data = UCBAdmissions, legend = TRUE, beside = TRUE) oops, discrimination.
- barplot(xtabs(Freq ~ Admit+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE) different departments have different standards
- barplot(xtabs(Freq ~ Gender+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE) men mostly applied to easy departments, women mostly applied to difficult departments
- barplot(xtabs(Freq ~ Gender+Dept, data = UCBAdmissions), legend = TRUE, beside = TRUE)
- barchart(Admit ~ Freq, data = C)
- barchart(Admit ~ Freq|Gender, data = C)
- barchart(Admit ~ Freq | Dept, groups = Gender, data = C)
- barchart(Admit ~ Freq | Dept, groups = Gender, data = C, auto.key = TRUE)
- Histograms
- ~ breaks | wool{*}tension, data = warpbreaks
- ~ weight | feed, data = chickwts
- ~ weight | group, data = PlantGrowth
- ~ count | spray, data = InsectSprays
- ~ len | dose, data = ToothGrowth
- ~ decrease | treatment, data = OrchardSprays (or rowpos or colpos)
- Scatterplots

```
xyplot(Petal.Width ~ Petal.Length, data = iris, group = Species)
```

- Scatterplot matrices
- splom(~ cbind(GNP.deflator,GNP,Unemployed,Armed.Forces,Population,Year,Employed), data = longley)
- splom(~ cbind(pop15,pop75,dpi), data = LifeCycleSavings)
- splom(~ cbind(Murder, Assault, Rape), data = USArrests)
- splom(~ cbind(CONT, INTG, DMNR), data = USJudgeRatings)
- splom(~ cbind(area,peri,shape,perm), data = rock)

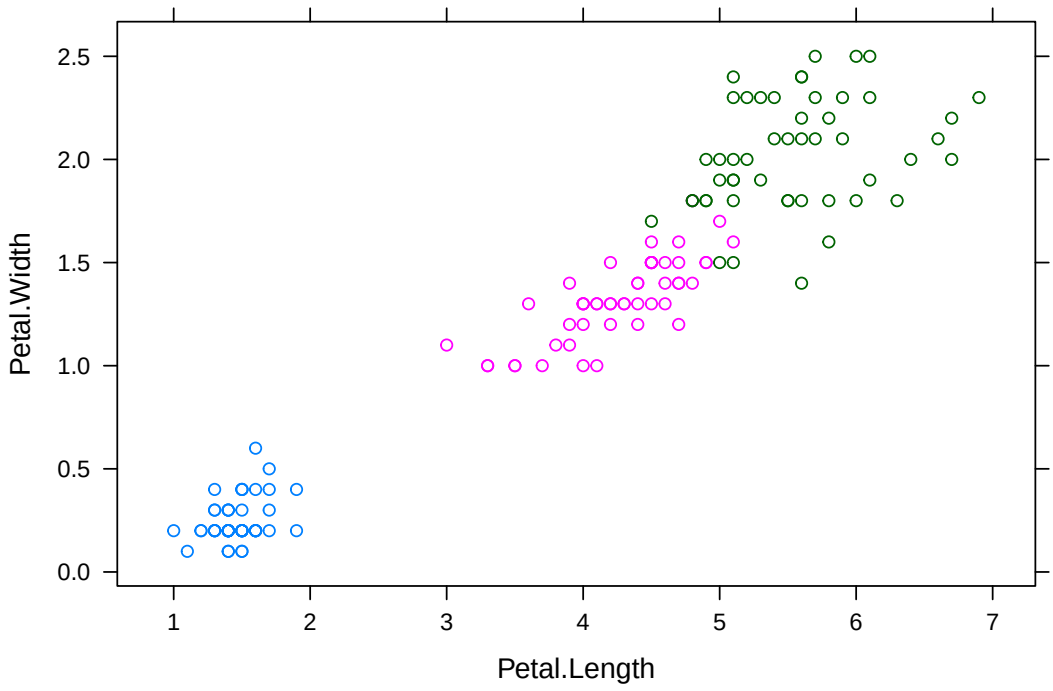

Figure 3.8: Scatterplot of Petal width versus length in the `iris` data.

- splom(~ cbind(Air.Flow, Water.Temp, Acid.Conc., stack.loss), data = stackloss)
- splom(~ cbind(Fertility,Agriculture,Examination,Education,Catholic,Infant.Mortality), data = swiss)
- splom(~ cbind(Fertility,Agriculture,Examination), data = swiss) (positive and negative)
- Dot charts
- dotchart(USPersonalExpenditure)
- dotchart(t(USPersonalExpenditure))
- dotchart(WorldPhones) (transpose is no good)
- freeny.x is no good, neither is volcano
- dotchart(UCBAdmissions{[},,1{]})
- dotplot(Survived ~ Freq | Class, groups = Sex, data = B)
- dotplot(Admit ~ Freq | Dept, groups = Gender, data = C)
- Mosaic plot
- mosaic(~ Survived + Class + Age + Sex, data = Titanic) (or just mosaic(Titanic))
- mosaic(~ Admit + Dept + Gender, data = UCBAdmissions)

- Spine plots
- spineplot(xtabs(Freq ~ Admit + Gender, data = UCBAdmissions))
- rescaled barplot
- Quantile-quantile plots: There are two ways to do this. One way is to compare two independent samples (of the same size). qqplot(x,y). Another way is to compare the sample quantiles of one variable to the theoretical quantiles of another distribution.

Given two samples $x_1, x_2, \ldots, x_n$, and $y_1, y_2, \ldots, y_n$, we may find the order statistics $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ and $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(n)}$. Next, plot the n points $(x_{(1)}, y_{(1)})$, $(x_{(2)}, y_{(2)})$, $\ldots$, $(x_{(n)}, y_{(n)})$.

It is clear that if $x_{(k)} = y_{(k)}$ for all $k = 1, 2, \ldots, n$, then we will have a straight line. It is also clear that in the real world, a straight line is NEVER observed, and instead we have a scatterplot that hopefully had a general linear trend. What do the rules tell us?

- If the y-intercept of the line is greater (less) than zero, then the center of the Y data is greater (less) than the center of the X data.
- If the slope of the line is greater (less) than one, then the spread of the Y data is greater (less) than the spread of the X data.

3.6.3 Lattice Graphics

The following types of plots are useful when there is one variable of interest and there is a factor in the data set by which the variable is categorized.

It is sometimes nice to set `lattice.options(default.theme = "col.whitebg")`

Side by side boxplots

```
bwplot(~weight | feed, data = chickwts)
```

Histograms

```
histogram(~age | education, data = infert)
```

Scatterplots

```
xyplot(Petal.Length ~ Petal.Width | Species, data = iris)
```

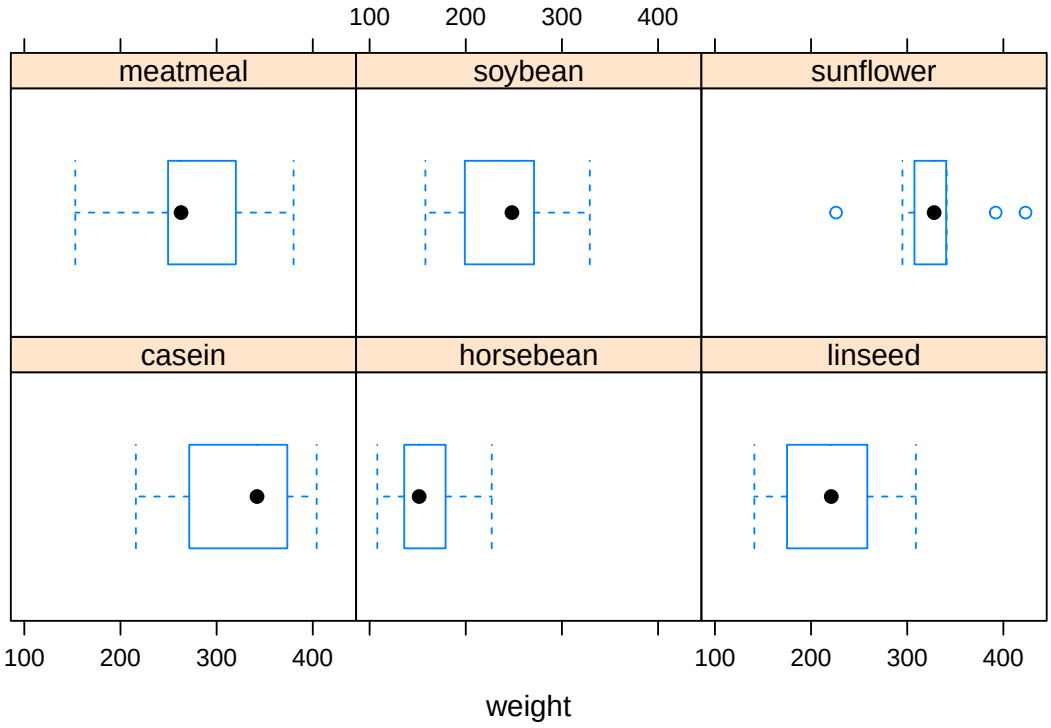

Figure 3.9: Boxplots of `weight` by `feed` type in the `chickwts` data.

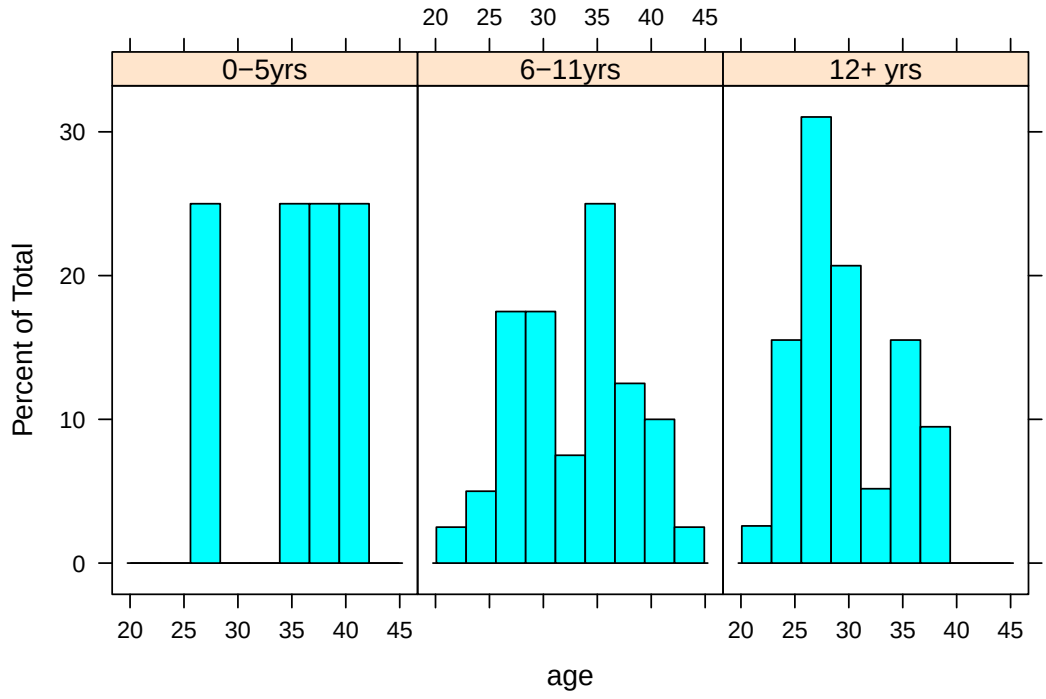

Figure 3.10: Histograms of `age` by `education` level from the `infert` data.

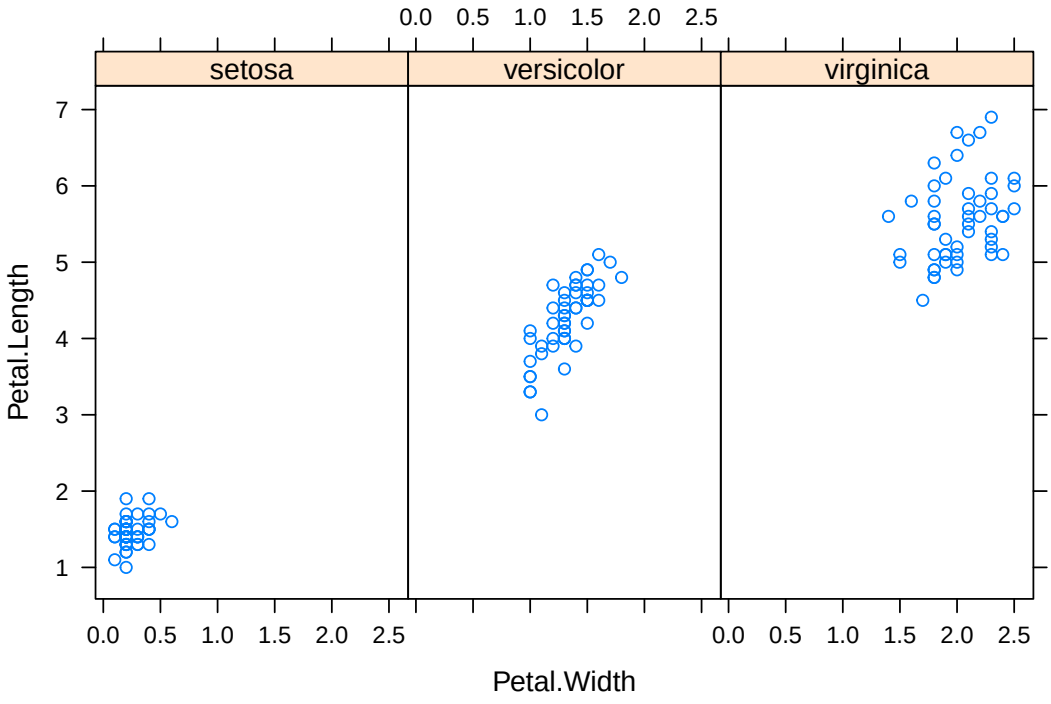

Figure 3.11: An `xyplot` of `Petal.Length` versus `Petal.Width` by `Species` in the `iris` data.

Coplots

```
coplot(conc ~ uptake | Type * Treatment, data = CO2)
```

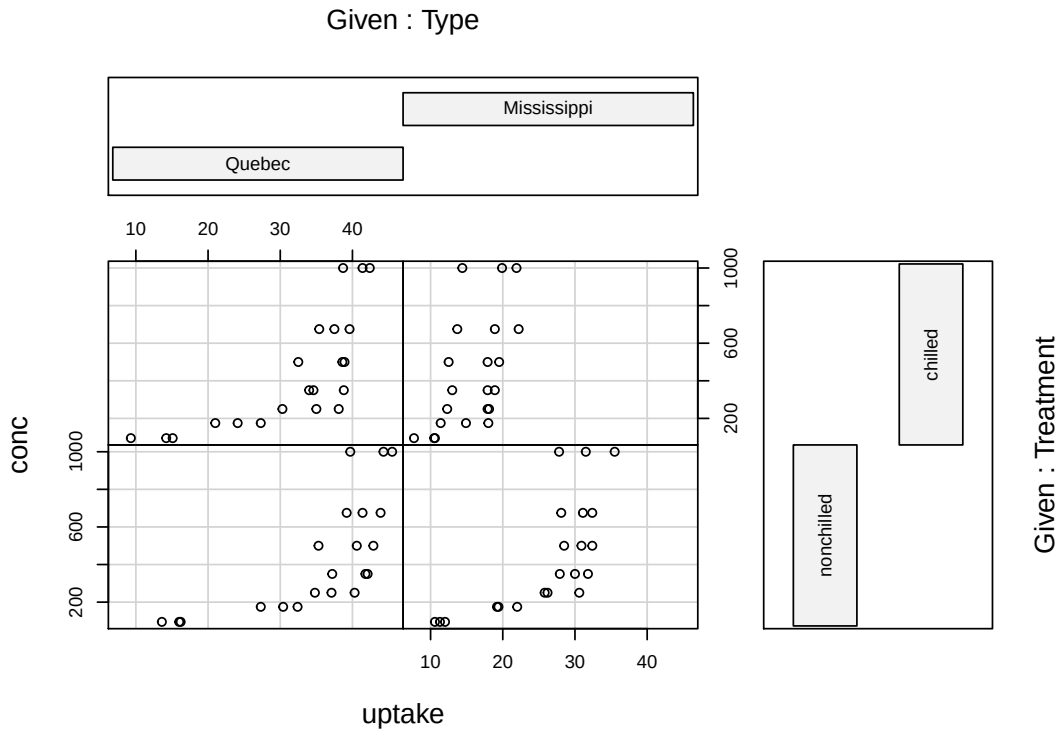

Figure 3.12: A `coplot` of `conc` versus `uptake` by `Type` and `Treatment`.

```
NULL
```

3.7 Exercises

Open R and issue the following commands at the command line to get started. Note that you need to have the `RcmdrPlugin.IPSUR` package [152] installed, and for some exercises you need the `e1071` package [96].

```
library("RcmdrPlugin.IPSUR")
data(RcmdrTestDrive)
attach(RcmdrTestDrive)
names(RcmdrTestDrive)
```

To load the data in the R Commander (`Rcmdr`), click the `Data Set` button, and select `RcmdrTestDrive` as the active data set. To learn more about the data set and where it comes from,

type `?RcmdrTestDrive` at the command line.

Exercise 3.1. Perform a summary of all variables in `RcmdrTestDrive`. You can do this with the command `summary(RcmdrTestDrive)`.

Alternatively, you can do this in the `Rcmdr` with the sequence `Statistics` ▹ `Summaries` ▹ `Active Data Set`. Report the values of the summary statistics for each variable.

Exercise 3.2. Make a table of the `race` variable. Do this with `Statistics` ▹ `Summaries` ▹ `Frequency Distributions - IPSUR...` 1. Which ethnicity has the highest frequency? 1. Which ethnicity has the lowest frequency? 1. Include a bar graph of `race`. Do this with `Graphs` ▹ `IPSUR - Bar Graph...`

Exercise 3.3. Calculate the average `salary` by the factor `gender`. Do this with `Statistics` ▹ `Summaries` ▹ `Table of Statistics...`

1. Which `gender` has the highest mean `salary`?
2. Report the highest mean `salary`.
3. Compare the spreads for the genders by calculating the standard deviation of `salary` by `gender`. Which `gender` has the biggest standard deviation?
4. Make boxplots of `salary` by `gender` with the following method:
   i) On the `Rcmdr`, click `Graphs` ▹ `IPSUR - Boxplot...`
   ii) In the `Variable` box, select `salary`.
   iii) Click the `Plot by groups...` box and select `gender`. Click `OK`.
   iv) Click `OK` to graph the boxplot.

How does the boxplot compare to your answers to (1) and (3)?

Exercise 3.4. For this problem we will study the variable `reduction`.

1. Find the order statistics and store them in a vector `x`. *Hint:* `x <- sort(reduction)`
2. Find $x_{(137)}$, the 137^{th} order statistic.
3. Find the IQR.
4. Find the Five Number Summary (5NS).
5. Use the 5NS to calculate what the width of a boxplot of `reduction` would be.
6. Compare your answers (3) and (5). Are they the same? If not, are they close?
7. Make a boxplot of `reduction`, and include the boxplot in your report. You can do this with the `boxplot` function, or in `Rcmdr` with `Graphs` ▹ `IPSUR - Boxplot...`
8. Are there any potential/suspected outliers? If so, list their values. *Hint:* use your answer to (a).
9. Using the rules discussed in the text, classify answers to (8), if any, as *potential* or *suspected* outliers.

Exercise 3.5. In this problem we will compare the variables `before` and `after`. Don't forget `library("e1071")`.

1. Examine the two measures of center for both variables. Judging from these measures, which variable has a higher center?

2. Which measure of center is more appropriate for `before`? (You may want to look at a boxplot.) Which measure of center is more appropriate for `after`?
3. Based on your answer to (2), choose an appropriate measure of spread for each variable, calculate it, and report its value. Which variable has the biggest spread? (Note that you need to make sure that your measures are on the same scale.)
4. Calculate and report the skewness and kurtosis for `before`. Based on these values, how would you describe the shape of `before`?
5. Calculate and report the skewness and kurtosis for `after`. Based on these values, how would you describe the shape of `after`?
6. Plot histograms of `before` and `after` and compare them to your answers to (4) and (5).

Exercise 3.6. Describe the following data sets just as if you were communicating with an alien, but one who has had a statistics class. Mention the salient features (data type, important properties, anything special). Support your answers with the appropriate visual displays and descriptive statistics.

1. Conversion rates of Euro currencies stored in `euro`.
2. State abbreviations stored in `state.abb`.
3. Areas of the world's landmasses stored in `islands`.
4. Areas of the 50 United States stored in `state.area`.
5. Region of the 50 United States stored in `state.region`.

Chapter 4

Probability

In this chapter we define the basic terminology associated with probability and derive some of its properties. We discuss three interpretations of probability. We discuss conditional probability and independent events, along with Bayes' Theorem. We finish the chapter with an introduction to random variables, which paves the way for the next two chapters.

In this book we distinguish between two types of experiments: *deterministic* and *random*. A *deterministic* experiment is one whose outcome may be predicted with certainty beforehand, such as combining Hydrogen and Oxygen, or adding two numbers such as $2 + 3$. A *random* experiment is one whose outcome is determined by chance. We posit that the outcome of a random experiment may not be predicted with certainty beforehand, even in principle. Examples of random experiments include tossing a coin, rolling a die, and throwing a dart on a board, how many red lights you encounter on the drive home, how many ants traverse a certain patch of sidewalk over a short period, *etc*.

What do I want them to know?

- that there are multiple interpretations of probability, and the methods used depend somewhat on the philosophy chosen
- nuts and bolts of basic probability jargon: sample spaces, events, probability functions, *etc*.
- how to count
- conditional probability and its relationship with independence
- Bayes' Rule and how it relates to the subjective view of probability
- what we mean by 'random variables', and where they come from

4.1 Sample Spaces

For a random experiment E, the set of all possible outcomes of E is called the *sample space* and is denoted by the letter S. For a coin-toss experiment, S would be the results "Head" and "Tail", which we may represent by $S = \{H, T\}$. Formally, the performance of a random experiment is the unpredictable selection of an outcome in S.

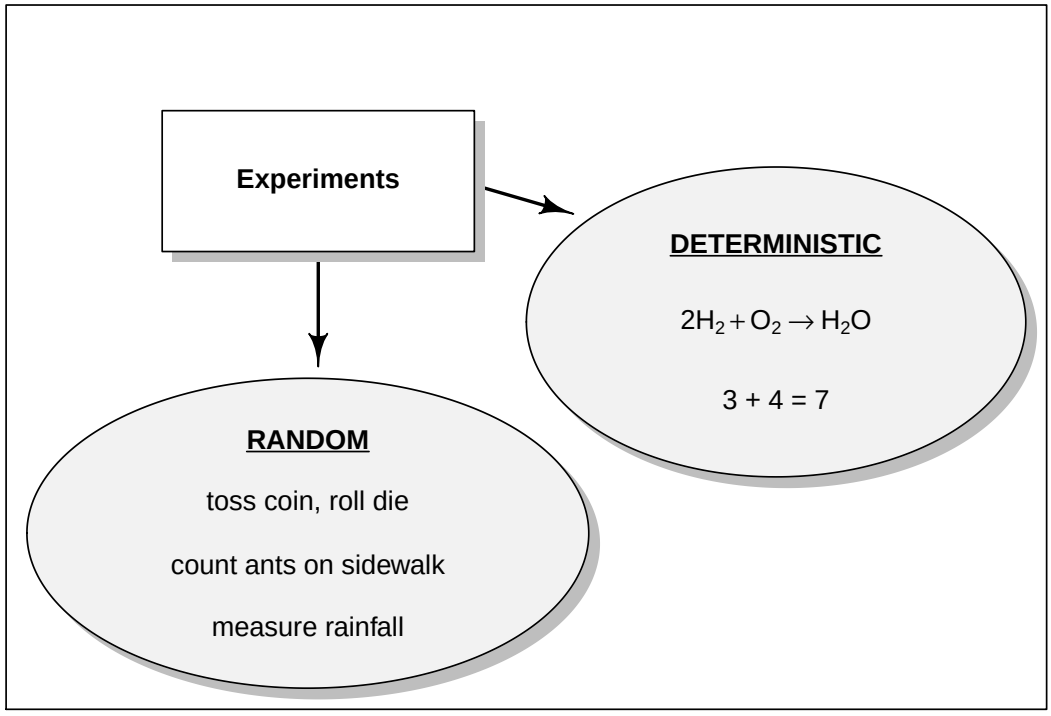

Figure 4.1: Two types of experiments.

4.1.1 How to do it with R

Most of the probability work in this book is done with the `prob` package [77]. A sample space is (usually) represented by a *data frame*, that is, a rectangular collection of variables (see Section 3.5.2). Each row of the data frame corresponds to an outcome of the experiment. The data frame choice is convenient both for its simplicity and its compatibility with the R Commander. Data frames alone are, however, not sufficient to describe some of the more interesting probabilistic applications we will study later; to handle those we will need to consider a more general *list* data structure. See Section 4.6.2 for details.

Example 4.1. Consider the random experiment of dropping a Styrofoam cup onto the floor from a height of four feet. The cup hits the ground and eventually comes to rest. It could land upside down, right side up, or it could land on its side. We represent these possible outcomes of the random experiment by the following.

```
S <- data.frame(lands = c("down","up","side"))
S
```

```
  lands
1  down
2    up
3  side
```

The sample space `S` contains the column `lands` which stores the outcomes `down`, `up`, and `side`.

Some sample spaces are so common that convenience wrappers were written to set them up with minimal effort. The underlying machinery that does the work includes the `expand.grid` function in the `base` package [106], `combn` in the `combinat` package [22], and `permsn` in the `prob` package [77][1].

Consider the random experiment of tossing a coin. The outcomes are H and T. We can set up the sample space quickly with the `tosscoin` function:

```
tosscoin(1)
```

```
  toss1
1     H
2     T
```

The number 1 tells `tosscoin` that we only want to toss the coin once. We could toss it three times:

```
tosscoin(3)
```

```
  toss1 toss2 toss3
1     H     H     H
2     T     H     H
3     H     T     H
4     T     T     H
```

[1] We can find solutions of the normal equations even when $\mathbf{X}^T\mathbf{X}$ is not of full rank, but the topic falls outside the scope of this book. The interested reader can consult an advanced text such as Rao [121].

```
5     H     H     T
6     T     H     T
7     H     T     T
8     T     T     T
```

Alternatively we could roll a fair die:

```
rolldie(1)
```

```
  X1
1  1
2  2
3  3
4  4
5  5
6  6
```

The `rolldie` function defaults to a 6-sided die, but we can specify others with the `nsides` argument. The command `rolldie(3, nsides = 4)` would be used to roll a 4-sided die three times.

Perhaps we would like to draw one card from a standard set of playing cards (it is a long data frame):

```
head(cards())
```

```
  rank suit
1    2 Club
2    3 Club
3    4 Club
4    5 Club
5    6 Club
6    7 Club
```

The `cards` function that we just used has optional arguments `jokers` (if you would like Jokers to be in the deck) and `makespace` which we will discuss later. There is also a `roulette` function which returns the sample space associated with one spin on a roulette wheel. There are EU and USA versions available. Interested readers may contribute any other game or sample spaces that may be of general interest.

4.1.2 Sampling from Urns

This is perhaps the most fundamental type of random experiment. We have an urn that contains a bunch of distinguishable objects (balls) inside. We shake up the urn, reach inside, grab a ball, and take a look. That's all.

But there are all sorts of variations on this theme. Maybe we would like to grab more than one ball – say, two balls. What are all of the possible outcomes of the experiment now? It depends on how we sample. We could select a ball, take a look, put it back, and sample again. Another way would be to select a ball, take a look – but do not put it back – and sample again (equivalently, just reach in and

grab two balls). There are certainly more possible outcomes of the experiment in the former case than in the latter. In the first (second) case we say that sampling is done *with (without) replacement*.

There is more. Suppose we do not actually keep track of which ball came first. All we observe are the two balls, and we have no idea about the order in which they were selected. We call this *unordered sampling* (in contrast to *ordered*) because the order of the selections does not matter with respect to what we observe. We might as well have selected the balls and put them in a bag before looking.

Note that this one general class of random experiments contains as a special case all of the common elementary random experiments. Tossing a coin twice is equivalent to selecting two balls labeled H and T from an urn, with replacement. The die-roll experiment is equivalent to selecting a ball from an urn with six elements, labeled 1 through 6.

How to do it with R

The `prob` package [77] accomplishes sampling from urns with the `urnsamples` function, which has arguments `x`, `size`, `replace`, and `ordered`. The argument `x` represents the urn from which sampling is to be done. The `size` argument tells how large the sample will be. The `ordered` and `replace` arguments are logical and specify how sampling will be performed. We will discuss each in turn.

Example 4.2. Let our urn simply contain three balls, labeled 1, 2, and 3, respectively. We are going to take a sample of size 2 from the urn.

Ordered, With Replacement

If sampling is with replacement, then we can get any outcome 1, 2, or 3 on any draw. Further, by "ordered" we mean that we shall keep track of the order of the draws that we observe. We can accomplish this in R with

```
urnsamples(1:3, size = 2, replace = TRUE, ordered = TRUE)
```

```
  X1 X2
1  1  1
2  2  1
3  3  1
4  1  2
5  2  2
6  3  2
7  1  3
8  2  3
9  3  3
```

Notice that rows 2 and 4 are identical, save for the order in which the numbers are shown. Further, note that every possible pair of the numbers 1 through 3 are listed. This experiment is equivalent to rolling a 3-sided die twice, which we could have accomplished with `rolldie(2, nsides = 3)`.

Ordered, Without Replacement

Here sampling is without replacement, so we may not observe the same number twice in any row. Order is still important, however, so we expect to see the outcomes `1,2` and `2,1` somewhere in our data frame.

```
urnsamples(1:3, size = 2, replace = FALSE, ordered = TRUE)
```

```
  X1 X2
1  1  2
2  2  1
3  1  3
4  3  1
5  2  3
6  3  2
```

This is just as we expected. Notice that there are less rows in this answer due to the more restrictive sampling procedure. If the numbers 1, 2, and 3 represented "Fred", "Mary", and "Sue", respectively, then this experiment would be equivalent to selecting two people of the three to serve as president and vice-president of a company, respectively, and the sample space shown above lists all possible ways that this could be done.

Unordered, Without Replacement

Again, we may not observe the same outcome twice, but in this case, we will only retain those outcomes which (when jumbled) would not duplicate earlier ones.

```
urnsamples(1:3, size = 2, replace = FALSE, ordered = FALSE)
```

```
  X1 X2
1  1  2
2  1  3
3  2  3
```

This experiment is equivalent to reaching in the urn, picking a pair, and looking to see what they are. This is the default setting of `urnsamples`, so we would have received the same output by simply typing `urnsamples(1:3, 2)`.

Unordered, With Replacement

The last possibility is perhaps the most interesting. We replace the balls after every draw, but we do not remember the order in which the draws came.

```
urnsamples(1:3, size = 2, replace = TRUE, ordered = FALSE)
```

```
  X1 X2
1  1  1
2  1  2
```

```
3  1  3
4  2  2
5  2  3
6  3  3
```

We may interpret this experiment in a number of alternative ways. One way is to consider this as simply putting two 3-sided dice in a cup, shaking the cup, and looking inside – as in a game of *Liar's Dice*, for instance. Each row of the sample space is a potential pair we could observe. Another way is to view each outcome as a separate method to distribute two identical golf balls into three boxes labeled 1, 2, and 3. Regardless of the interpretation, `urnsamples` lists every possible way that the experiment can conclude.

Note that the urn does not need to contain numbers; we could have just as easily taken our urn to be `x = c("Red","Blue","Green")`. But, there is an *important* point to mention before proceeding. Astute readers will notice that in our example, the balls in the urn were *distinguishable* in the sense that each had a unique label to distinguish it from the others in the urn. A natural question would be, "What happens if your urn has indistinguishable elements, for example, what if `x = c("Red","Red","Blue")`?" The answer is that `urnsamples` behaves as if each ball in the urn is distinguishable, regardless of its actual contents. We may thus imagine that while there are two red balls in the urn, the balls are such that we can tell them apart (in principle) by looking closely enough at the imperfections on their surface.

In this way, when the `x` argument of `urnsamples` has repeated elements, the resulting sample space may appear to be `ordered = TRUE` even when, in fact, the call to the function was `urnsamples(..., ordered = FALSE)`. Similar remarks apply for the `replace` argument.

4.2 Events

An *event* A is merely a collection of outcomes, or in other words, a subset of the sample space[2]. After the performance of a random experiment E we say that the event A *occurred* if the experiment's outcome belongs to A. We say that a bunch of events $A_1, A_2, A_3, \ldots$ are *mutually exclusive* or *disjoint* if $A_i \cap A_j = \emptyset$ for any distinct pair $A_i \neq A_j$. For instance, in the coin-toss experiment the events $A = \{\text{Heads}\}$ and $B = \{\text{Tails}\}$ would be mutually exclusive. Now would be a good time to review the algebra of sets in Appendix **??**.

4.2.1 How to do it with R

Given a data frame sample/probability space `S`, we may extract rows using the `[]` operator:

```
S <- tosscoin(2, makespace = TRUE)
S[1:3, ]
```

[2]We are taking great leaps over the mathematical details. In particular, we have yet to show that s^2 has a chi-square distribution and we have not even come close to showing that b_i and s_{b_i} are independent. But these are entirely outside the scope of the present book and the reader may rest assured that the proofs await in later classes. See C.R. Rao for more.

```
  toss1 toss2 probs
1     H     H  0.25
2     T     H  0.25
3     H     T  0.25
```

```
S[c(2,4), ]
```

```
  toss1 toss2 probs
2     T     H  0.25
4     T     T  0.25
```

and so forth. We may also extract rows that satisfy a logical expression using the `subset` function, for instance

```
S <- cards()
```

```
subset(S, suit == "Heart")
```

```
   rank  suit
27    2 Heart
28    3 Heart
29    4 Heart
30    5 Heart
31    6 Heart
32    7 Heart
33    8 Heart
34    9 Heart
35   10 Heart
36    J Heart
37    Q Heart
38    K Heart
39    A Heart
```

```
subset(S, rank %in% 7:9)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
32    7   Heart
33    8   Heart
34    9   Heart
45    7   Spade
46    8   Spade
47    9   Spade
```

We could continue indefinitely. Also note that mathematical expressions are allowed:

```
subset(rolldie(3), X1+X2+X3 > 16)
```

```
    X1 X2 X3
180  6  6  5
210  6  5  6
215  5  6  6
216  6  6  6
```

4.2.2 Functions for Finding Subsets

It does not take long before the subsets of interest become complicated to specify. Yet the main idea remains: we have a particular logical condition to apply to each row. If the row satisfies the condition, then it should be in the subset. It should not be in the subset otherwise. The ease with which the condition may be coded depends of course on the question being asked. Here are a few functions to get started.

The `%in%` function

The function `%in%` helps to learn whether each value of one vector lies somewhere inside another vector.

```
x <- 1:10
y <- 8:12
y %in% x
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

Notice that the returned value is a vector of length 5 which tests whether each element of `y` is in `x`, in turn.

The `isin` function

It is more common to want to know whether the *whole* vector `y` is in `x`. We can do this with the `isin` function.

```
isin(x,y)
```

```
[1] FALSE
```

Of course, one may ask why we did not try something like `all(y %in% x)`, which would give a single result, `TRUE`. The reason is that the answers are different in the case that `y` has repeated values. Compare:

```
x <- 1:10
y <- c(3,3,7)
```

```
all(y %in% x)
```

```
[1] TRUE
```

```
isin(x,y)
```

```
[1] FALSE
```

The reason for the above is of course that `x` contains the value 3, but `x` does not have *two* 3's. The difference is important when rolling multiple dice, playing cards, *etc*. Note that there is an optional argument `ordered` which tests whether the elements of `y` appear in `x` in the order in which they are appear in `y`. The consequences are

```
isin(x, c(3,4,5), ordered = TRUE)
```

```
[1] TRUE
```

```
isin(x, c(3,5,4), ordered = TRUE)
```

```
[1] FALSE
```

The connection to probability is that have a data frame sample space and we would like to find a subset of that space. A `data.frame` method was written for `isin` that simply applies the function to each row of the data frame. We can see the method in action with the following:

```
S <- rolldie(4)
subset(S, isin(S, c(2,2,6), ordered = TRUE))
```

```
     X1 X2 X3 X4
188   2  2  6  1
404   2  2  6  2
620   2  2  6  3
836   2  2  6  4
1052  2  2  6  5
1088  2  2  1  6
1118  2  1  2  6
1123  1  2  2  6
1124  2  2  2  6
1125  3  2  2  6
1126  4  2  2  6
1127  5  2  2  6
1128  6  2  2  6
1130  2  3  2  6
1136  2  4  2  6
1142  2  5  2  6
1148  2  6  2  6
1160  2  2  3  6
1196  2  2  4  6
1232  2  2  5  6
1268  2  2  6  6
```

There are a few other functions written to find useful subsets, namely, `countrep` and `isrep`. Essentially these were written to test for (or count) a specific number of designated values in outcomes. See the documentation for details.

4.2.3 Set Union, Intersection, and Difference

Given subsets A and B, it is often useful to manipulate them in an algebraic fashion. To this end, we have three set operations at our disposal: union, intersection, and difference. Below is a table that summarizes the pertinent information about these operations.

Table 4.1: Basic set operations. The first column lists the name, the second shows the typical notation, the third describes set membership, and the fourth shows how to accomplish it with R.

Name	Denoted	Defined by elements	Code
Union	$A \cup B$	in A or B or both	`union(A,B)`
Intersection	$A \cap B$	in both A and B	`intersect(A,B)`
Difference	$A \backslash B$	in A but not in B	`setdiff(A,B)`

Some examples follow.

```
S <- cards()
A <- subset(S, suit == "Heart")
B <- subset(S, rank %in% 7:9)
```

We can now do some set algebra:

```
union(A,B)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
27    2   Heart
28    3   Heart
29    4   Heart
30    5   Heart
31    6   Heart
32    7   Heart
33    8   Heart
34    9   Heart
35   10   Heart
36    J   Heart
37    Q   Heart
```

```
38    K   Heart
39    A   Heart
45    7   Spade
46    8   Spade
47    9   Spade
```

```
intersect(A,B)
```

```
   rank  suit
32    7 Heart
33    8 Heart
34    9 Heart
```

```
setdiff(A,B)
```

```
   rank  suit
27    2 Heart
28    3 Heart
29    4 Heart
30    5 Heart
31    6 Heart
35   10 Heart
36    J Heart
37    Q Heart
38    K Heart
39    A Heart
```

```
setdiff(B,A)
```

```
   rank    suit
6     7    Club
7     8    Club
8     9    Club
19    7 Diamond
20    8 Diamond
21    9 Diamond
45    7   Spade
46    8   Spade
47    9   Spade
```

Notice that `setdiff` is not symmetric. Further, note that we can calculate the *complement* of a set A, denoted A^c and defined to be the elements of S that are not in A simply with `setdiff(S,A)`. There have been methods written for `intersect`, `setdiff`, `subset`, and `union` in the case that the input objects are of class `ps`. See Section 4.6.2.

Note 4.1. When the `prob` package [@prob] loads you will notice a message: `The following object(s) are masked from package:base: intersect, setdiff`. The reason for this message is that there already exist methods for the functions `intersect`, `setdiff`, `subset`, and `union` in the `base` package which ships with R. However, these methods were designed for when

the arguments are vectors of the same mode. Since we are manipulating sample spaces which are data frames and lists, it was necessary to write methods to handle those cases as well. When the `prob` package is loaded, R recognizes that there are multiple versions of the same function in the search path and acts to shield the new definitions from the existing ones. But there is no cause for alarm, thankfully, because the `prob` functions have been carefully defined to match the usual `base` package definition in the case that the arguments are vectors.

4.3 Model Assignment

Let us take a look at the coin-toss experiment more closely. What do we mean when we say "the probability of Heads" or write $\mathbb{P}$(Heads)? Given a coin and an itchy thumb, how do we go about finding what $\mathbb{P}$(Heads) should be?

4.3.1 The Measure Theory Approach

This approach states that the way to handle $\mathbb{P}$(Heads) is to define a mathematical function, called a *probability measure*, on the sample space. Probability measures satisfy certain axioms (to be introduced later) and have special mathematical properties, so not just any mathematical function will do. But in any given physical circumstance there are typically all sorts of probability measures from which to choose, and it is left to the experimenter to make a reasonable choice – one usually based on considerations of objectivity. For the tossing coin example, a valid probability measure assigns probability p to the event {Heads}, where p is some number $0 \leq p \leq 1$. An experimenter that wishes to incorporate the symmetry of the coin would choose $p = 1/2$ to balance the likelihood of {Heads} and {Tails}.

Once the probability measure is chosen (or determined), there is not much left to do. All assignments of probability are made by the probability function, and the experimenter needs only to plug the event {Heads} into to the probability function to find $\mathbb{P}$(Heads). In this way, the probability of an event is simply a calculated value, nothing more, nothing less. Of course this is not the whole story; there are many theorems and consequences associated with this approach that will keep us occupied for the remainder of this book. The approach is called *measure theory* because the measure (probability) of a set (event) is associated with how big it is (how likely it is to occur).

The measure theory approach is well suited for situations where there is symmetry to the experiment, such as flipping a balanced coin or spinning an arrow around a circle with well-defined pie slices. It is also handy because of its mathematical simplicity, elegance, and flexibility. There are literally volumes of information that one can prove about probability measures, and the cold rules of mathematics allow us to analyze intricate probabilistic problems with vigor.

The large degree of flexibility is also a disadvantage, however. When symmetry fails it is not always obvious what an "objective" choice of probability measure should be; for instance, what probability should we assign to {Heads} if we spin the coin rather than flip it? (It is not $1/2$.) Furthermore, the mathematical rules are restrictive when we wish to incorporate subjective knowledge into the model, knowledge which changes over time and depends on the experimenter, such as personal knowledge about the properties of the specific coin being flipped, or of the person doing the flipping.

The mathematician who revolutionized this way to do probability theory was Andrey Kolmogorov, who published a landmark monograph in 1933. See http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Kolmogorov.html for more information.

4.3.2 Relative Frequency Approach

This approach states that the way to determine $\mathbb{P}(\text{Heads})$ is to flip the coin repeatedly, in exactly the same way each time. Keep a tally of the number of flips and the number of Heads observed. Then a good approximation to $\mathbb{P}(\text{Heads})$ will be

$$\mathbb{P}(\text{Heads}) \approx \frac{\text{number of observed Heads}}{\text{total number of flips}}. \tag{4.1}$$

The mathematical underpinning of this approach is the celebrated *Law of Large Numbers* which may be loosely described as follows. Let E be a random experiment in which the event A either does or does not occur. Perform the experiment repeatedly, in an identical manner, in such a way that the successive experiments do not influence each other. After each experiment, keep a running tally of whether or not the event A occurred. Let S_n count the number of times that A occurred in the n experiments. Then the law of large numbers says that

$$\frac{S_n}{n} \to \mathbb{P}(A) \text{ as } n \to \infty. \tag{4.2}$$

As the reasoning goes, to learn about the probability of an event A we need only repeat the random experiment to get a reasonable estimate of the probability's value, and if we are not satisfied with our estimate then we may simply repeat the experiment more times all the while confident that with more and more experiments our estimate will stabilize to the true value.

The frequentist approach is good because it is relatively light on assumptions and does not worry about symmetry or claims of objectivity like the measure-theoretic approach does. It is perfect for the spinning coin experiment. One drawback to the method is that one can never know the exact value of a probability, only a long-run approximation. It also does not work well with experiments that can not be repeated indefinitely, say, the probability that it will rain today, the chances that you get will get an A in your Statistics class, or the probability that the world is destroyed by nuclear war.

This approach was espoused by Richard von Mises in the early twentieth century, and some of his main ideas were incorporated into the measure theory approach. See http://www-history.mcs.st-andrews.ac.uk/Biographies/Mises.html for more.

4.3.3 The Subjective Approach

The subjective approach interprets probability as the experimenter's *degree of belief* that the event will occur. The estimate of the probability of an event is based on the totality of the individual's knowledge at the time. As new information becomes available, the estimate is modified accordingly

to best reflect his/her current knowledge. The method by which the probabilities are updated is commonly done with Bayes' Rule, discussed in Section 4.8.

So for the coin toss example, a person may have $\mathbb{P}(\text{Heads}) = 1/2$ in the absence of additional information. But perhaps the observer knows additional information about the coin or the thrower that would shift the probability in a certain direction. For instance, parlor magicians may be trained to be quite skilled at tossing coins, and some are so skilled that they may toss a fair coin and get nothing but Heads, indefinitely. I have *seen* this. It was similarly claimed in *Bringing Down the House* [97] that MIT students were accomplished enough with cards to be able to cut a deck to the same location, every single time. In such cases, one clearly should use the additional information to assign $\mathbb{P}(\text{Heads})$ away from the symmetry value of $1/2$.

This approach works well in situations that cannot be repeated indefinitely, for example, to assign your probability that you will get an A in this class, the chances of a devastating nuclear war, or the likelihood that a cure for the common cold will be discovered.

The roots of subjective probability reach back a long time. See http://en.wikipedia.org/wiki/Subjective_probability for a short discussion and links to references about the subjective approach.

4.3.4 Equally Likely Model (ELM)

We have seen several approaches to the assignment of a probability model to a given random experiment and they are very different in their underlying interpretation. But they all cross paths when it comes to the equally likely model which assigns equal probability to all elementary outcomes of the experiment.

The ELM appears in the measure theory approach when the experiment boasts symmetry of some kind. If symmetry guarantees that all outcomes have equal "size", and if outcomes with equal "size" should get the same probability, then the ELM is a logical objective choice for the experimenter. Consider the balanced 6-sided die, the fair coin, or the dart board with equal-sized wedges.

The ELM appears in the subjective approach when the experimenter resorts to indifference or ignorance with respect to his/her knowledge of the outcome of the experiment. If the experimenter has no prior knowledge to suggest that (s)he prefer Heads over Tails, then it is reasonable for the him/her to assign equal subjective probability to both possible outcomes.

The ELM appears in the relative frequency approach as a fascinating fact of Nature: when we flip balanced coins over and over again, we observe that the proportion of times that the coin comes up Heads tends to $1/2$. Of course if we assume that the measure theory applies then we can prove that the sample proportion must tend to 1/2 as expected, but that is putting the cart before the horse, in a manner of speaking.

The ELM is only available when there are finitely many elements in the sample space.

How to do it with R

In the `prob` package [77], a probability space is an object of outcomes `S` and a vector of probabilities (called `probs`) with entries that correspond to each outcome in `S`. When `S` is a data frame, we may

simply add a column called `probs` to S and we will be finished; the probability space will simply be a data frame which we may call S. In the case that S is a list, we may combine the `outcomes` and `probs` into a larger list, `space`; it will have two components: `outcomes` and `probs`. The only requirements we need are for the entries of `probs` to be nonnegative and `sum(probs)` to be one.

To accomplish this in R, we may use the `probspace` function. The general syntax is `probspace(x, probs)`, where `x` is a sample space of outcomes and `probs` is a vector (of the same length as the number of outcomes in `x`). The specific choice of `probs` depends on the context of the problem, and some examples follow to demonstrate some of the more common choices.

Example 4.3. The Equally Likely Model asserts that every outcome of the sample space has the same probability, thus, if a sample space has n outcomes, then `probs` would be a vector of length n with identical entries $1/n$. The quickest way to generate `probs` is with the `rep` function. We will start with the experiment of rolling a die, so that $n = 6$. We will construct the sample space, generate the `probs` vector, and put them together with `probspace`.

```
outcomes <- rolldie(1)
p <- rep(1/6, times = 6)
probspace(outcomes, probs = p)
```

```
  X1     probs
1  1 0.1666667
2  2 0.1666667
3  3 0.1666667
4  4 0.1666667
5  5 0.1666667
6  6 0.1666667
```

The `probspace` function is designed to save us some time in the most common situations. For example, due to the especial simplicity of the sample space in this case, we could have achieved the same result with only (note the name change for the first column)

```
probspace(1:6, probs = p)
```

```
  x     probs
1 1 0.1666667
2 2 0.1666667
3 3 0.1666667
4 4 0.1666667
5 5 0.1666667
6 6 0.1666667
```

Further, since the equally likely model plays such a fundamental role in the study of probability the `probspace` function will assume that the equally model is desired if no `probs` are specified. Thus, we get the same answer with only

```
probspace(1:6)
```

```
  x     probs
1 1 0.1666667
```

```
2 2 0.1666667
3 3 0.1666667
4 4 0.1666667
5 5 0.1666667
6 6 0.1666667
```

And finally, since rolling dice is such a common experiment in probability classes, the `rolldie` function has an additional logical argument `makespace` that will add a column of equally likely `probs` to the generated sample space:

```
rolldie(1, makespace = TRUE)
```

```
  X1     probs
1  1 0.1666667
2  2 0.1666667
3  3 0.1666667
4  4 0.1666667
5  5 0.1666667
6  6 0.1666667
```

or just `rolldie(1, TRUE)`. Many of the other sample space functions (`tosscoin`, `cards`, `roulette`, *etc.*) have similar `makespace` arguments. Check the documentation for details.

One sample space function that does NOT have a `makespace` option is the `urnsamples` function. This was intentional. The reason is that under the varied sampling assumptions the outcomes in the respective sample spaces are NOT, in general, equally likely. It is important for the user to carefully consider the experiment to decide whether or not the outcomes are equally likely and then use `probspace` to assign the model.

Example 4.4 (An unbalanced coin)**.** While the `makespace` argument to `tosscoin` is useful to represent the tossing of a *fair* coin, it is not always appropriate. For example, suppose our coin is not perfectly balanced, for instance, maybe the H side is somewhat heavier such that the chances of a H appearing in a single toss is 0.70 instead of 0.5. We may set up the probability space with

```
probspace(tosscoin(1), probs = c(0.70, 0.30))
```

```
  toss1 probs
1     H   0.7
2     T   0.3
```

The same procedure can be used to represent an unbalanced die, roulette wheel, *etc*.

4.3.5 Words of Warning

It should be mentioned that while the splendour of R is uncontested, it, like everything else, has limits both with respect to the sample/probability spaces it can manage and with respect to the finite accuracy of the representation of most numbers (see the R FAQ 7.31). When playing around with probability, one may be tempted to set up a probability space for tossing 100 coins or rolling 50 dice

in an attempt to answer some scintillating question. (Bear in mind: rolling a die just 9 times has a sample space with over *10 million* outcomes.)

Alas, even if there were enough RAM to barely hold the sample space (and there were enough time to wait for it to be generated), the infinitesimal probabilities that are associated with *so many* outcomes make it difficult for the underlying machinery to handle reliably. In some cases, special algorithms need to be called just to give something that holds asymptotically. User beware.

4.4 Properties of Probability

4.4.1 Probability Functions

A *probability function* is a rule that associates with each event A of the sample space a single number $\mathbb{P}(A) = p$, called the *probability of* A. Any probability function $\mathbb{P}$ satisfies the following three Kolmogorov Axioms:

Axiom 4.2. $\mathbb{P}(A) \geq 0$ *for any event* $A \subset S$.

Axiom 4.3. $\mathbb{P}(S) = 1$.

Axiom 4.4. *If the events* $A_1, A_2, A_3 \ldots$ *are disjoint then*

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) \text{ for every } n, \tag{4.3}$$

and furthermore,

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{4.4}$$

The intuition behind the axioms goes like this: first, the probability of an event should never be negative. Second, since the sample space contains all possible outcomes, its probability should be one, or 100%. The last axiom may look intimidating but it simply means that in a sequence of disjoint events (in other words, sets that do not overlap), the total probability (measure) should equal the sum of its parts. For example, the chance of rolling a 1 or a 2 on a die should be the chance of rolling a 1 plus the chance of rolling a 2.

4.4.2 Properties

For any events A and B,

1. <> $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$. **Proof:** Since $A \cup A^c = S$ and $A \cap A^c = \emptyset$, we have

$$1 = \mathbb{P}(S) = \mathbb{P}(A \cup A^c) = \mathbb{P}(A) + \mathbb{P}(A^c).$$

2. $\mathbb{P}(\emptyset) = 0$. **Proof:** Note that $\emptyset = S^c$, and use Property 1.
3. If $A \subset B$, then $\mathbb{P}(A) \leq \mathbb{P}(B)$. **Proof:** Write $B = A \cup (B \cap A^c)$, and notice that $A \cap (B \cap A^c) = \emptyset$; thus
$$\mathbb{P}(B) = \mathbb{P}(A \cup (B \cap A^c)) = \mathbb{P}(A) + \mathbb{P}(B \cap A^c) \geq \mathbb{P}(A),$$
since $\mathbb{P}(B \cap A^c) \geq 0$.
4. $0 \leq \mathbb{P}(A) \leq 1$. **Proof:** The left inequality is immediate from Axiom **??**, and the second inequality follows from Property 3 since $A \subset S$.
5. **The General Addition Rule.**
$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B). \tag{4.5}$$
More generally, for events $A_1, A_2, A_3, \ldots, A_n$,
$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{P}(A_i \cap A_j) + \cdots + (-1)^{n-1} \mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right) \tag{4.6}$$
6. **The Theorem of Total Probability.** Let $B_1, B_2, \ldots, B_n$ be mutually exclusive and exhaustive. Then
$$\mathbb{P}(A) = \mathbb{P}(A \cap B_1) + \mathbb{P}(A \cap B_2) + \cdots + \mathbb{P}(A \cap B_n). \tag{4.7}$$

4.4.3 Assigning Probabilities

A model of particular interest is the *equally likely model*. The idea is to divide the sample space S into a finite collection of elementary events $\{a_1, a_2, \ldots, a_N\}$ that are equally likely in the sense that each a_i has equal chances of occurring. The probability function associated with this model must satisfy $\mathbb{P}(S) = 1$, by Axiom 2. On the other hand, it must also satisfy
$$\mathbb{P}(S) = \mathbb{P}(\{a_1, a_2, \ldots, a_N\}) = \mathbb{P}(a_1 \cup a_2 \cup \cdots \cup a_N) = \sum_{i=1}^{N} \mathbb{P}(a_i),$$
by Axiom 3. Since $\mathbb{P}(a_i)$ is the same for all i, each one necessarily equals $1/N$.

For an event $A \subset S$, we write A as a collection of elementary outcomes: if $A = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ then A has k elements and
$$\begin{aligned} \mathbb{P}(A) &= \mathbb{P}(a_{i_1}) + \mathbb{P}(a_{i_2}) + \cdots + \mathbb{P}(a_{i_k}), \\ &= \frac{1}{N} + \frac{1}{N} + \cdots + \frac{1}{N}, \\ &= \frac{k}{N} = \frac{\#(A)}{\#(S)}. \end{aligned}$$

In other words, under the equally likely model, the probability of an event A is determined by the number of elementary events that A contains.

Example 4.5. Consider the random experiment E of tossing a coin. Then the sample space is $S = \{H, T\}$, and under the equally likely model, these two outcomes have $\mathbb{P}(H) = \mathbb{P}(T) = 1/2$. This model is taken when it is reasonable to assume that the coin is fair.

Example 4.6. Suppose the experiment E consists of tossing a fair coin twice. The sample space may be represented by $S = \{HH, HT, TH, TT\}$. Given that the coin is fair and that the coin is tossed in an independent and identical manner, it is reasonable to apply the equally likely model.

What is $\mathbb{P}$(at least 1 Head)? Looking at the sample space we see the elements HH, HT, and TH have at least one Head; thus, $\mathbb{P}$(at least 1 Head) = 3/4.

What is $\mathbb{P}$(no Heads)? Notice that the event {no Heads} = {at least one Head}^{c}, which by Property **??** means $\mathbb{P}$(no Heads) = 1 − $\mathbb{P}$(at least one head) = 1 − 3/4 = 1/4. It is obvious in this simple example that the only outcome with no Heads is TT, however, this complementation trick can be handy in more complicated problems.

Example 4.7. Imagine a three child family, each child being either Boy (B) or Girl (G). An example sequence of siblings would be BGB. The sample space may be written

$$S = \{BBB,\ BGB,\ GBB,\ GGB,\ BBG,\ BGG,\ GBG,\ GGG,\ \}.$$

Note that for many reasons (for instance, it turns out that girls are slightly more likely to be born than boys), this sample space is *not* equally likely. For the sake of argument, however, we will assume that the elementary outcomes each have probability 1/8.

What is $\mathbb{P}$(exactly 2 Boys)? Inspecting the sample space reveals three outcomes with exactly two boys: $\{BBG, BGB, GBB\}$. Therefore $\mathbb{P}$(exactly 2 Boys) = 3/8.

What is $\mathbb{P}$(at most 2 Boys)? One way to solve the problem would be to count the outcomes that have 2 or less Boys, but a quicker way would be to recognize that the only way that the event {at most 2 Boys} does *not* occur is the event {all Boys}.

Thus

$$\mathbb{P}(\text{at most 2 Boys}) = 1 - \mathbb{P}(BBB) = 1 - 1/8 = 7/8.$$

Example 4.8. Consider the experiment of rolling a six-sided die, and let the outcome be the face showing up when the die comes to rest. Then $S = \{1, 2, 3, 4, 5, 6\}$. It is usually reasonable to suppose that the die is fair, so that the six outcomes are equally likely.

Example 4.9. Consider a standard deck of 52 cards. These are usually labeled with the four *suits*: Clubs, Diamonds, Hearts, and Spades, and the 13 *ranks*: 2, 3, 4, ..., 10, Jack (J), Queen (Q), King (K), and Ace (A). Depending on the game played, the Ace may be ranked below 2 or above King.

Let the random experiment E consist of drawing exactly one card from a well-shuffled deck, and let the outcome be the face of the card. Define the events A = {draw an Ace} and B = {draw a Club}. Bear in mind: we are only drawing one card.

Immediately we have $\mathbb{P}(A) = 4/52$ since there are four Aces in the deck; similarly, there are 13 Clubs which implies $\mathbb{P}(B) = 13/52$.

What is $\mathbb{P}(A \cap B)$? We realize that there is only one card of the 52 which is an Ace and a Club at the same time, namely, the Ace of Clubs. Therefore $\mathbb{P}(A \cap B) = 1/52$.

To find $\mathbb{P}(A \cup B)$ we may use the above with the General Addition Rule to get

$$\begin{aligned} \mathbb{P}(A \cup B) &= \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B), \\ &= 4/52 + 13/52 - 1/52, \\ &= 16/52. \end{aligned}$$

Example 4.10. Staying with the deck of cards, let another random experiment be the selection of a five card stud poker hand, where "five card stud" means that we draw exactly five cards from the deck without replacement, no more, and no less. It turns out that the sample space S is so large and complicated that we will be obliged to settle for the trivial description $S = \{\text{all possible 5 card hands}\}$ for the time being. We will have a more precise description later.

What is $\mathbb{P}$(Royal Flush), or in other words, $\mathbb{P}$(A, K, Q, J, 10 all in the same suit)?

It should be clear that there are only four possible royal flushes. Thus, if we could only count the number of outcomes in S then we could simply divide four by that number and we would have our answer under the equally likely model. This is the subject of Section 4.5.

How to do it with R

Probabilities are calculated in the `prob` package [77] with the `Prob` function.

Consider the experiment of drawing a card from a standard deck of playing cards. Let's denote the probability space associated with the experiment as `S`, and let the subsets `A` and `B` be defined by the following:

```
S <- cards(makespace = TRUE)
A <- subset(S, suit == "Heart")
B <- subset(S, rank %in% 7:9)
```

Now it is easy to calculate

```
Prob(A)
```

```
[1] 0.25
```

Note that we can get the same answer with

```
Prob(S, suit == "Heart")
```

```
[1] 0.25
```

We also find `Prob(B) = 0.23` (listed here approximately, but 12/52 actually). Internally, the `Prob` function operates by summing the `probs` column of its argument. It will find subsets on-the-fly if desired.

We have as yet glossed over the details. More specifically, `Prob` has three arguments: `x`, which is a probability space (or a subset of one), `event`, which is a logical expression used to define a subset, and `given`, which is described in Section 4.6.

WARNING. The `event` argument is used to define a subset of `x`, that is, the only outcomes used in the probability calculation will be those that are elements of `x` and satisfy `event` simultaneously. In other words, `Prob(x, event)` calculates `Prob(intersect(x, subset(x, event)))`.

Consequently, `x` should be the entire probability space in the case that `event` is non-null.

4.5 Counting Methods

The equally-likely model is a convenient and popular way to analyze random experiments. And when the equally likely model applies, finding the probability of an event A amounts to nothing more than counting the number of outcomes that A contains (together with the number of events in S). Hence, to be a master of probability one must be skilled at counting outcomes in events of all kinds.

Proposition 4.1 (The Multiplication Principle)**.** Suppose that an experiment is composed of two successive steps. Further suppose that the first step may be performed in n_1 distinct ways while the second step may be performed in n_2 distinct ways. Then the experiment may be performed in $n_1 n_2$ distinct ways. More generally, if the experiment is composed of k successive steps which may be performed in $n_1, n_2, \ldots, n_k$ distinct ways, respectively, then the experiment may be performed in $n_1 n_2 \cdots n_k$ distinct ways.

Example 4.11. We would like to order a pizza. It will be sure to have cheese (and marinara sauce), but we may elect to add one or more of the following five (5) available toppings:

pepperoni, sausage, anchovies, olives, and green peppers.

How many distinct pizzas are possible?

There are many ways to approach the problem, but the quickest avenue employs the Multiplication Principle directly. We will separate the action of ordering the pizza into a series of stages. At the first stage, we will decide whether or not to include pepperoni on the pizza (two possibilities). At the next stage, we will decide whether or not to include sausage on the pizza (again, two possibilities). We will continue in this fashion until at last we will decide whether or not to include green peppers on the pizza.

At each stage we will have had two options, or ways, to select a pizza to be made. The Multiplication Principle says that we should multiply the 2's to find the total number of possible pizzas: $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5 = 32$.

Example 4.12. We would like to buy a desktop computer to study statistics. We go to a website to build our computer our way. Given a line of products we have many options to customize our computer. In particular, there are 2 choices for a processor, 3 different operating systems, 4 levels of memory, 4 hard drives of differing sizes, and 10 choices for a monitor. How many possible types of computer must the company be prepared to build? **Answer:** $2 \cdot 3 \cdot 4 \cdot 4 \cdot 10 = 960$

4.5.1 Ordered Samples

Imagine a bag with n distinguishable balls inside. Now shake up the bag and select k balls at random. How many possible sequences might we observe?

Proposition 4.2. The number of ways in which one may select an ordered sample of k subjects from a population that has n distinguishable members is

- n^k if sampling is done with replacement,
- $n(n-1)(n-2)\cdots(n-k+1)$ if sampling is done without replacement.

Recall from calculus the notation for *factorials*:

$$\begin{aligned} 1! &= 1, \\ 2! &= 2 \cdot 1 = 2, \\ 3! &= 3 \cdot 2 \cdot 1 = 6, \\ &\vdots \\ n! &= n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1. \end{aligned}$$

Fact: The number of permutations of n elements is $n!$.

Example 4.13. Take a coin and flip it 7 times. How many sequences of Heads and Tails are possible? **Answer:** $2^7 = 128$.

Example 4.14. In a class of 20 students, we randomly select a class president, a class vice-president, and a treasurer. How many ways can this be done? **Answer:** $20 \cdot 19 \cdot 18 = 6840$.

Example 4.15. We rent five movies to watch over the span of two nights. We wish to watch 3 movies on the first night. How many distinct sequences of 3 movies could we possibly watch? *Answer:# $5 \cdot 4 \cdot 3 = 60$.

4.5.2 Unordered Samples

Proposition 4.3. The number of ways in which one may select an unordered sample of k subjects from a population that has n distinguishable members is

- $(n-1+k)!/[(n-1)!k!]$ if sampling is done with replacement,
- $n!/[k!(n-k)!]$ if sampling is done without replacement.

The quantity $n!/[k!(n-k)!]$ is called a *binomial coefficient* and plays a special role in mathematics; it is denoted

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{4.8}$$

and is read "n choose k".

Example 4.16. You rent five movies to watch over the span of two nights, but only wish to watch 3 movies the first night. Your friend, Fred, wishes to borrow some movies to watch at his house on the first night. You owe Fred a favor, and allow him to select 2 movies from the set of 5. How many choices does Fred have? **Answer:** $\binom{5}{2} = 10$.

Example 4.17. Place 3 six-sided dice into a cup. Next, shake the cup well and pour out the dice. How many distinct rolls are possible? **Answer:** $(6-1+3)!/[(6-1)!3!] = \binom{8}{5} = 56$.

How to do it with R

The factorial $n!$ is computed with the command `factorial(n)` and the binomial coefficient $\binom{n}{k}$ with the command `choose(n,k)`.

The sample spaces we have computed so far have been relatively small, and we can visually study them without much trouble. However, it is *very* easy to generate sample spaces that are prohibitively large. And while R is wonderful and powerful and does almost everything except wash windows, even R has limits of which we should be mindful.

But we often do not need to actually generate the sample space; it suffices to count the number of outcomes. The `nsamp` function will calculate the number of rows in a sample space made by `urnsamples` without actually devoting the memory resources necessary to generate the space. The arguments are `n`, the number of (distinguishable) objects in the urn, `k`, the sample size, and `replace`, `ordered`, as above.

(#tab-sampling-k-from-n)

Table 4.2: Sampling k from n objects with `urnsamples`.

	`ordered = TRUE`	`ordered = FALSE`
`replace = TRUE`	n^k	$(n-1+k)!/[(n-1)!k!]$
`replace = FALSE`	$n!/(n-k)!$	$\binom{n}{k}$

Example 4.18. We will compute the number of outcomes for each of the four `urnsamples` examples that we saw in Example 4.2. Recall that we took a sample of size two from an urn with three distinguishable elements.

```
nsamp(n=3, k=2, replace = TRUE, ordered = TRUE)
```

```
[1] 9
```

```
nsamp(n=3, k=2, replace = FALSE, ordered = TRUE)
```

```
[1] 6
```

```
nsamp(n=3, k=2, replace = FALSE, ordered = FALSE)
```

```
[1] 3
```

```
nsamp(n=3, k=2, replace = TRUE, ordered = FALSE)
```

```
[1] 6
```

Compare these answers with the length of the data frames generated above.

The Multiplication Principle

A benefit of `nsamp` is that it is *vectorized* so that entering vectors instead of numbers for `n`, `k`, `replace`, and `ordered` results in a vector of corresponding answers. This becomes particularly convenient for combinatorics problems.

Example 4.19. There are 11 artists who each submit a portfolio containing 7 paintings for competition in an art exhibition. Unfortunately, the gallery director only has space in the winners' section to accommodate 12 paintings in a row equally spread over three consecutive walls. The director decides to give the first, second, and third place winners each a wall to display the work of their choice. The walls boast 31 separate lighting options apiece. How many displays are possible?

Answer: The judges will pick 3 (ranked) winners out of 11 (with `rep = FALSE`, `ord = TRUE`). Each artist will select 4 of his/her paintings from 7 for display in a row (`rep = FALSE`, `ord = TRUE`), and lastly, each of the 3 walls has 31 lighting possibilities (`rep = TRUE`, `ord = TRUE`). These three numbers can be calculated quickly with

```
n <- c(11,7,31)
k <- c(3,4,3)
r <- c(FALSE,FALSE,TRUE)
```

```
x <- nsamp(n, k, rep = r, ord = TRUE)
```

(Notice that `ordered` is always `TRUE`; `nsamp` will recycle `ordered` and `replace` to the appropriate length.) By the Multiplication Principle, the number of ways to complete the experiment is the product of the entries of `x`:

```
prod(x)
```

```
[1] 24774195600
```

Compare this with the some other ways to compute the same thing:

```
(11*10*9)*(7*6*5*4)*31^3
```

```
[1] 24774195600
```

or alternatively

```
prod(9:11)*prod(4:7)*31^3
```

```
[1] 24774195600
```

or even

```
prod(factorial(c(11,7))/factorial(c(8,3)))*31^3
```

```
[1] 24774195600
```

As one can guess, in many of the standard counting problems there aren't substantial savings in the amount of typing; it is about the same using `nsamp` versus `factorial` and `choose`. But the virtue of `nsamp` lies in its collecting the relevant counting formulas in a one-stop shop. Ultimately, it is up to the user to choose the method that works best for him/herself.

Example 4.20 (The Birthday Problem)**.** Suppose that there are n people together in a room. Each person announces the date of his/her birthday in turn. The question is: what is the probability of at least one match? If we let the event A represent

$$A = \{\text{there is at least one match}\},$$

then we are looking for $\mathbb{P}(A)$, but as we soon will see, it will be more convenient for us to calculate $\mathbb{P}(A^c)$.

For starters we will ignore leap years and assume that there are only 365 days in a year. Second, we will assume that births are equally distributed over the course of a year (which is not true due to all sorts of complications such as hospital delivery schedules). See http://en.wikipedia.org/wiki/Birthday_problem for more.

Let us next think about the sample space. There are 365 possibilities for the first person's birthday, 365 possibilities for the second, and so forth. The total number of possible birthday sequences is therefore $\#(S) = 365^n$.

Now we will use the complementation trick we saw in Example 4.7. We realize that the only situation in which A does *not* occur is if there are *no* matches among all people in the room, that is, only when everybody's birthday is different, so

$$\mathbb{P}(A) = 1 - \mathbb{P}(A^c) = 1 - \frac{\#(A^c)}{\#(S)},$$

since the outcomes are equally likely. Let us then suppose that there are no matches. The first person has one of 365 possible birthdays. The second person must not match the first, thus, the second person has only 364 available birthdays from which to choose. Similarly, the third person has only 363 possible birthdays, and so forth, until we reach the n^{th} person, who has only $365 - n + 1$ remaining possible days for a birthday. By the Multiplication Principle, we have $\#(A^c) = 365 \cdot 364 \cdots (365 - n + 1)$, and

$$\mathbb{P}(A) = 1 - \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n} = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{(365 - n + 1)}{365}. \tag{4.9}$$

As a surprising consequence, consider this: how many people does it take to be in the room so that the probability of at least one match is at least 0.50? Clearly, if there is only $n = 1$ person in the room then the probability of a match is zero, and when there are $n = 366$ people in the room there is a 100% chance of a match (recall that we are ignoring leap years). So how many people does it take so that there is an equal chance of a match and no match?

When I have asked this question to students, the usual response is "somewhere around $n = 180$ people" in the room. The reasoning seems to be that in order to get a 50% chance of a match, there should be 50% of the available days to be occupied. The number of students in a typical classroom is 25, so as a companion question I ask students to estimate the probability of a match when there are $n = 25$ students in the room. Common estimates are a 1%, or 0.5%, or even 0.1% chance of a match. After they have given their estimates, we go around the room and each student announces their birthday. More often than not, we observe a match in the class, to the students' disbelief.

Students are usually surprised to hear that, using the formula above, one needs only $n = 23$ students to have a greater than 50% chance of at least one match. Figure 4.2 shows a graph of the birthday probabilities:

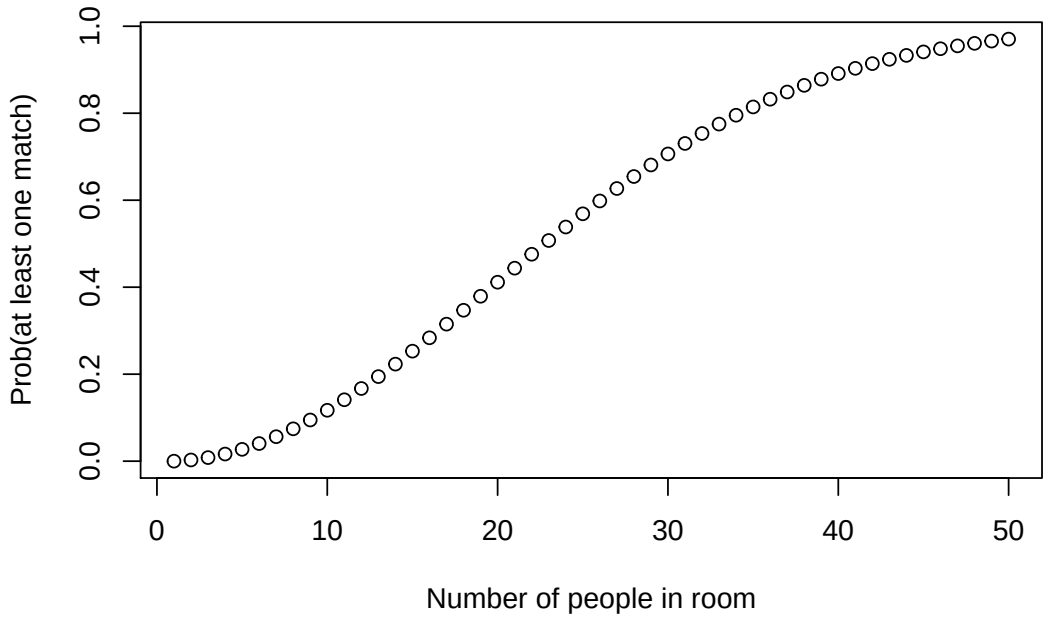

Figure 4.2: The birthday problem. The horizontal line is at $p = 0.50$ and the vertical line is at $n = 23$.

How to do it with R

We can make the plot in Figure 4.2 with the following sequence of commands.

```
library(RcmdrPlugin.IPSUR)
g <- Vectorize(pbirthday.ipsur)
plot(1:50, g(1:50), xlab = "Number of people in room",
  ylab = "Prob(at least one match)" )
abline(h = 0.5)
abline(v = 23, lty = 2)
remove(g)
```

There is a `Birthday problem` item in the `Probability` menu of `RcmdrPlugin.IPSUR`. In the base R version, one can compute approximate probabilities for the more general case of probabilities other than 1/2, for differing total number of days in the year, and even for more than two matches.

4.6 Conditional Probability

Consider a full deck of 52 standard playing cards. Now select two cards from the deck, in succession. Let $A = \{\text{first card drawn is an Ace}\}$ and $B = \{\text{second card drawn is an Ace}\}$. Since there are four Aces in the deck, it is natural to assign $\mathbb{P}(A) = 4/52$. Suppose we look at the first card. What now is the probability of B? Of course, the answer depends on the value of the first card. If the first card is an Ace, then the probability that the second also is an Ace should be $3/51$, but if the first card is not an Ace, then the probability that the second is an Ace should be $4/51$. As notation for these two situations we write

$$\mathbb{P}(B|A) = 3/51, \quad \mathbb{P}(B|A^c) = 4/51.$$

Definition 4.1. The conditional probability of B given A, denoted $\mathbb{P}(B|A)$, is defined by

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{4.10}$$

We will not be discussing a conditional probability of B given A when $\mathbb{P}(A) = 0$, even though this theory exists, is well developed, and forms the foundation for the study of stochastic processes[3].

Toss a coin twice. The sample space is given by $S = \{HH,\ HT,\ TH,\ TT\}$. Let $A = \{\text{a head occurs}\}$ and $B = \{\text{a head and tail occur}\}$. It should be clear that $\mathbb{P}(A) = 3/4$, $\mathbb{P}(B) = 2/4$, and $\mathbb{P}(A \cap B) = 2/4$. What now are the probabilities $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$?

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} = \frac{2/4}{2/4} = 1,$$

in other words, once we know that a Head and Tail occur, we may be certain that a Head occurs. Next

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)} = \frac{2/4}{3/4} = \frac{2}{3},$$

which means that given the information that a Head has occurred, we no longer need to account for the outcome TT, and the remaining three outcomes are equally likely with exactly two outcomes lying in the set B.

[3] In other words, a variable might be highly significant one moment but then fail to be significant when another variable is added to the model. When this happens it often indicates a problem with the explanatory variables, such as *multicollinearity*. See Section **??**.

Example 4.21. Toss a six-sided die twice. The sample space consists of all ordered pairs (i, j) of the numbers $1, 2, \ldots, 6$, that is, $S = \{(1,1),\ (1,2), \ldots, (6,6)\}$. We know from Section 4.5 that $\#(S) = 6^2 = 36$. Let $A = \{\text{outcomes match}\}$ and $B = \{\text{sum of outcomes at least 8}\}$. The sample space may be represented by a matrix:

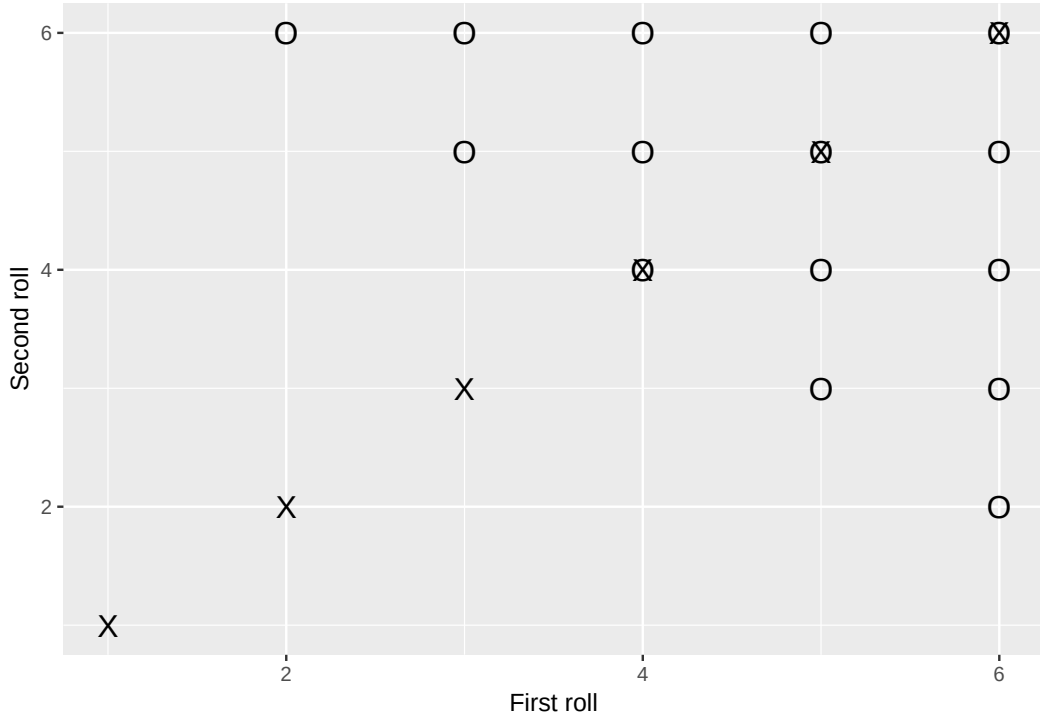

Figure 4.3: Rolling two dice. The outcomes in A are marked with X, the outcomes in B are marked with O.

The outcomes lying in the event A are marked with the symbol "X", the outcomes falling in B are marked with "O", and the outcomes in $A \cap B$ are those where the letters overlap. Now it is clear that $\mathbb{P}(A) = 6/36$, $\mathbb{P}(B) = 15/36$, and $\mathbb{P}(A \cap B) = 3/36$. Finally,

$$\mathbb{P}(A|B) = \frac{3/36}{15/36} = \frac{1}{5}, \quad \mathbb{P}(B|A) = \frac{3/36}{6/36} = \frac{1}{2}.$$

Again, we see that given the knowledge that B occurred (the 15 outcomes in the upper right triangle), there are 3 of the 15 that fall into the set A, thus the probability is $3/15$. Similarly, given that A occurred (we are on the diagonal), there are 3 out of 6 outcomes that also fall in B, thus, the probability of B given A is $1/2$.

4.6.1 How to do it with R

Continuing with Example 4.21, the first thing to do is set up the probability space with the `rolldie` function.

```
S <- rolldie(2, makespace = TRUE)  # assumes ELM
head(S)                            #  first few rows
```

```
  X1 X2      probs
1  1  1 0.02777778
2  2  1 0.02777778
3  3  1 0.02777778
4  4  1 0.02777778
5  5  1 0.02777778
6  6  1 0.02777778
```

Next we define the events

```
A <- subset(S, X1 == X2)
B <- subset(S, X1 + X2 >= 8)
```

And now we are ready to calculate probabilities. To do conditional probability, we use the `given` argument of the `Prob` function:

```
Prob(A, given = B)
```

```
[1] 0.2
```

```
Prob(B, given = A)
```

```
[1] 0.5
```

Note that we do not actually need to define the events A and B separately as long as we reference the original probability space S as the first argument of the `Prob` calculation:

```
Prob(S, X1==X2, given = (X1 + X2 >= 8) )
```

```
[1] 0.2
```

```
Prob(S, X1+X2 >= 8, given = (X1==X2) )
```

```
[1] 0.5
```

4.6.2 Properties and Rules

The following theorem establishes that conditional probabilities behave just like regular probabilities when the conditioned event is fixed.

Theorem 4.1. For any fixed event A with $\mathbb{P}(A) > 0$,

1. $\mathbb{P}(B|A) \geq 0$, for all events $B \subset S$,
2. $\mathbb{P}(S|A) = 1$, and
3. If B_1, B_2, B_3,... are disjoint events, then

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} B_k \,\middle|\, A\right) = \sum_{k=1}^{\infty} \mathbb{P}(B_k|A). \tag{4.11}$$

In other words, $\mathbb{P}(\cdot|A)$ is a legitimate probability function. With this fact in mind, the following properties are immediate:

Proposition 4.4. For any events A, B, and C with $\mathbb{P}(A) > 0$,

1. $\mathbb{P}(B^c|A) = 1 - \mathbb{P}(B|A)$.
2. If $B \subset C$ then $\mathbb{P}(B|A) \leq \mathbb{P}(C|A)$.
3. $\mathbb{P}[(B \cup C)|A] = \mathbb{P}(B|A) + \mathbb{P}(C|A) - \mathbb{P}[(B \cap C|A)]$.
4. **The Multiplication Rule.** For any two events A and B,

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A). \tag{4.12}$$

And more generally, for events $A_1, A_2, A_3, \ldots, A_n$,

$$\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1)\cdots\mathbb{P}(A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1}). \tag{4.13}$$

The Multiplication Rule is very important because it allows us to find probabilities in random experiments that have a sequential structure, as the next example shows.

Example 4.22. At the beginning of the section we drew two cards from a standard playing deck. Now we may answer our original question, what is $\mathbb{P}$(both Aces)?

$$\mathbb{P}(\text{both Aces}) = \mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A) = \frac{4}{52} \cdot \frac{3}{51} \approx 0.00452.$$

How to do it with R

Continuing Example 4.22, we set up the probability space by way of a three step process. First we employ the `cards` function to get a data frame `L` with two columns: `rank` and `suit`. Both columns are stored internally as factors with 13 and 4 levels, respectively.

Next we sample two cards randomly from the `L` data frame by way of the `urnsamples` function. It returns a list `M` which contains all possible pairs of rows from `L` (there are `choose(52,2)` of them). The sample space for this experiment is exactly the list `M`.

At long last we associate a probability model with the sample space. This is right down the `probspace` function's alley. It assumes the equally likely model by default. We call this result `N` which is an object of class `ps` – short for "probability space".

But do not be intimidated. The object `N` is nothing more than a list with two elements: `outcomes` and `probs`. The `outcomes` element is itself just another list, with `choose(52,2)` entries, each one a data frame with two rows which correspond to the pair of cards chosen. The `probs` element is just a vector with `choose(52,2)` entries all the same: `1/choose(52,2)`.

Putting all of this together we do

```
L <- cards()
M <- urnsamples(L, size = 2)
N <- probspace(M)
N[[1]][[1]];  N$probs[1]
```

```
  rank suit
1    2 Club
2    3 Club
```

```
[1] 0.0007541478
```

Now that we have the probability space `N` we are ready to do some probability. We use the `Prob` function, just like before. The only trick is to specify the event of interest correctly, and recall that we were interested in $\mathbb{P}$(both Aces). But if the cards are both Aces then the `rank` of both cards should be `A`, which sounds like a job for the `all` function:

```
Prob(N, all(rank == "A"))
```

```
[1] 0.004524887
```

Note that this value matches what we found in Example 4.22, above. We could calculate all sorts of probabilities at this point; we are limited only by the complexity of the event's computer representation.

Example 4.23. Consider an urn with 10 balls inside, 7 of which are red and 3 of which are green. Select 3 balls successively from the urn. Let $A = \{1^{\text{st}}$ ball is red$\}$, $B = \{2^{\text{nd}}$ ball is red$\}$, and $C = \{3^{\text{rd}}$ ball is red$\}$. Then

$$\mathbb{P}(\text{all 3 balls are red}) = \mathbb{P}(A \cap B \cap C) = \frac{7}{10} \cdot \frac{6}{9} \cdot \frac{5}{8} \approx 0.2917.$$

How to do it with R

Example 4.23 is similar to Example 4.22, but it is even easier. We need to set up an urn (vector `L`) to hold the balls, we sample from `L` to get the sample space (data frame `M`), and we associate a probability vector (column `probs`) with the outcomes (rows of `M`) of the sample space. The final result is a probability space (an ordinary data frame `N`).

It is easier for us this time because our urn is a vector instead of a `cards()` data frame. Before there were two dimensions of information associated with the outcomes (rank and suit) but presently we have only one dimension (color).

```
L <- rep(c("red","green"), times = c(7,3))
M <- urnsamples(L, size = 3, replace = FALSE, ordered = TRUE)
N <- probspace(M)
```

Now let us think about how to set up the event {all 3 balls are red}. Rows of `N` that satisfy this condition have `X1=="red" & X2=="red" & X3=="red"`, but there must be an easier way. Indeed, there is. The `isrep` function (short for "is repeated") in the `prob` package was written for this

purpose. The command `isrep(N,"red",3)` will test each row of `N` to see whether the value `red` appears 3 times. The result is exactly what we need to define an event with the `Prob` function. Observe

```
Prob(N, isrep(N, "red", 3))
```

```
[1] 0.2916667
```

Note that this answer matches what we found in Example 4.23. Now let us try some other probability questions. What is the probability of getting two `red`'s?

```
Prob(N, isrep(N, "red", 2))
```

```
[1] 0.525
```

Note that the exact value is 21/40; we will learn a quick way to compute this in Section **??**. What is the probability of observing `red`, then `green`, then `red`?

```
Prob(N, isin(N, c("red","green","red"), ordered = TRUE))
```

```
[1] 0.175
```

Note that the exact value is 7/40 (do it with the Multiplication Rule). What is the probability of observing `red`, `green`, and `red`, in no particular order?

```
Prob(N, isin(N, c("red","green","red")))
```

```
[1] 0.525
```

We already knew this. It is the probability of observing two `red`'s, above.

Example 4.24. Consider two urns, the first with 5 red balls and 3 green balls, and the second with 2 red balls and 6 green balls. Your friend randomly selects one ball from the first urn and transfers it to the second urn, without disclosing the color of the ball. You select one ball from the second urn. What is the probability that the selected ball is red?

Let A = {transferred ball is red} and B = {selected ball is red}. Write

$$\begin{aligned} B &= S \cap B \\ &= (A \cup A^c) \cap B \\ &= (A \cap B) \cup (A^c \cap B) \end{aligned}$$

and notice that $A \cap B$ and $A^c \cap B$ are disjoint. Therefore

$$\begin{aligned}\mathbb{P}(B) &= \mathbb{P}(A \cap B) + \mathbb{P}(A^c \cap B) \\ &= \mathbb{P}(A)\mathbb{P}(B|A) + \mathbb{P}(A^c)\mathbb{P}(B|A^c) \\ &= \frac{5}{8} \cdot \frac{3}{9} + \frac{3}{8} \cdot \frac{2}{9} \\ &= \frac{21}{72}\end{aligned}$$

(which is 7/24 in lowest terms).

Example 4.25. We saw the `RcmdrTestDrive` data set in Chapter 2 in which a two-way table of the smoking status versus the gender was

```
library(RcmdrPlugin.IPSUR)
data(RcmdrTestDrive)
.Table <- xtabs( ~ smoking + gender, data = RcmdrTestDrive)
addmargins(.Table) # Table with marginal distributions
```

```
           gender
smoking     Female Male Sum
  Nonsmoker     61   75 136
  Smoker         9   23  32
  Sum           70   98 168
```

If one person were selected at random from the data set, then we see from the two-way table that $\mathbb{P}(\text{Female}) = 70/168$ and $\mathbb{P}(\text{Smoker}) = 32/168$. Now suppose that one of the subjects quits smoking, but we do not know the person's gender. If we now select one nonsmoker at random, what would be $\mathbb{P}(\text{Female})$? This example is just like the last example, but with different labels. Let $A = \{\text{the quitter is a female}\}$ and $B = \{\text{selected nonsmoker is a female}\}$. Write

$$\begin{aligned}B &= S \cap B \\ &= (A \cup A^c) \cap B \\ &= (A \cap B) \cup (A^c \cap B)\end{aligned}$$

and notice that $A \cap B$ and $A^c \cap B$ are disjoint. Therefore

$$\begin{aligned}\mathbb{P}(B) &= \mathbb{P}(A \cap B) + \mathbb{P}(A^c \cap B), \\ &= \mathbb{P}(A)\mathbb{P}(B|A) + \mathbb{P}(A^c)\mathbb{P}(B|A^c), \\ &= \frac{9}{32} \cdot \frac{62}{137} + \frac{23}{32} \cdot \frac{76}{137}, \\ &= \frac{2306}{4384},\end{aligned}$$

(which is 1153/2192 in lowest terms).

Using the same reasoning, we can return to the example from the beginning of the section and show that

$$\mathbb{P}(\{\text{second card is an Ace}\}) = 4/52.$$

4.7 Independent Events

Toss a coin twice. The sample space is $S = \{HH,\ HT,\ TH,\ TT\}$. We know that $\mathbb{P}(1^{\text{st}}$ toss is $H) = 2/4$, $\mathbb{P}(2^{\text{nd}}$ toss is $H) = 2/4$, and $\mathbb{P}(\text{both } H) = 1/4$. Then

$$\begin{aligned}\mathbb{P}(2^{\text{nd}} \text{ toss is } H \mid 1^{\text{st}} \text{ toss is } H) &= \frac{\mathbb{P}(\text{both } H)}{\mathbb{P}(1^{\text{st}} \text{ toss is } H)},\\ &= \frac{1/4}{2/4},\\ &= \mathbb{P}(2^{\text{nd}} \text{ toss is } H).\end{aligned}$$

Intuitively, this means that the information that the first toss is H has no bearing on the probability that the second toss is H. The coin does not remember the result of the first toss.

Definition 4.2. Events A and B are said to be *independent* if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B). \tag{4.14}$$

Otherwise, the events are said to be *dependent*.

The connection with the above example stems from the following. We know from Section 4.6 that when $\mathbb{P}(B) > 0$ we may write

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{4.15}$$

In the case that A and B are independent, the numerator of the fraction factors so that $\mathbb{P}(B)$ cancels with the result:

$$\mathbb{P}(A|B) = \mathbb{P}(A) \text{ when } A,\ B \text{ are independent.} \tag{4.16}$$

The interpretation in the case of independence is that the information that the event B occurred does not influence the probability of the event A occurring. Similarly, $\mathbb{P}(B|A) = \mathbb{P}(B)$, and so the occurrence of the event A likewise does not affect the probability of event B. It may seem more natural to define A and B to be independent when $\mathbb{P}(A|B) = \mathbb{P}(A)$; however, the conditional probability $\mathbb{P}(A|B)$ is only defined when $\mathbb{P}(B) > 0$. Our definition is not limited by this restriction. It can be shown that when $\mathbb{P}(A),\ \mathbb{P}(B) > 0$ the two notions of independence are equivalent.

Proposition 4.5. If the events A and B are independent then

- A and B^c are independent,
- A^c and B are independent,
- A^c and B^c are independent.

Proof. Suppose that A and B are independent. We will show the second one; the others are similar. We need to show that

$$\mathbb{P}(A^c \cap B) = \mathbb{P}(A^c)\mathbb{P}(B).$$

To this end, note that the Multiplication Rule, Equation (4.12) implies

$$\begin{aligned} \mathbb{P}(A^c \cap B) &= \mathbb{P}(B)\mathbb{P}(A^c|B), \\ &= \mathbb{P}(B)[1 - \mathbb{P}(A|B)], \\ &= \mathbb{P}(B)\mathbb{P}(A^c). \end{aligned}$$

□

Definition 4.3. The events A, B, and C are *mutually independent* if the following four conditions are met:

$$\begin{aligned} \mathbb{P}(A \cap B) &= \mathbb{P}(A)\mathbb{P}(B), \\ \mathbb{P}(A \cap C) &= \mathbb{P}(A)\mathbb{P}(C), \\ \mathbb{P}(B \cap C) &= \mathbb{P}(B)\mathbb{P}(C), \end{aligned}$$

and

$$\mathbb{P}(A \cap B \cap C) = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C).$$

If only the first three conditions hold then A, B, and C are said to be independent *pairwise*. Note that pairwise independence is not the same as mutual independence when the number of events is larger than two.

We can now deduce the pattern for n events, $n > 3$. The events will be mutually independent only if they satisfy the product equality pairwise, then in groups of three, in groups of four, and so forth, up to all n events at once. For n events, there will be $2^n - n - 1$ equations that must be satisfied (see Exercise **??**). Although these requirements for a set of events to be mutually independent may seem stringent, the good news is that for most of the situations considered in this book the conditions will all be met (or at least we will suppose that they are).

Example 4.26. Toss ten coins. What is the probability of observing at least one Head?

Answer: Let $A_i = \{\text{the } i^{\text{th}} \text{ coin shows } H\},\ i = 1, 2, \ldots, 10$. Supposing that we toss the coins in such a way that they do not interfere with each other, this is one of the situations where all of the A_i may be considered mutually independent due to the nature of the tossing. Of course, the only way that there will not be at least one Head showing is if all tosses are Tails. Therefore,

$$\begin{aligned}\mathbb{P}(\text{at least one } H) &= 1 - \mathbb{P}(\text{all } T),\\ &= 1 - \mathbb{P}(A_1^c \cap A_2^c \cap \cdots \cap A_{10}^c),\\ &= 1 - \mathbb{P}(A_1^c)\mathbb{P}(A_2^c)\cdots\mathbb{P}(A_{10}^c),\\ &= 1 - \left(\frac{1}{2}\right)^{10},\end{aligned}$$

which is approximately 0.9990234.

4.7.1 How to do it with R

Example 4.27. Toss ten coins. What is the probability of observing at least one Head?

```
S <- tosscoin(10, makespace = TRUE)
A <- subset(S, isrep(S, vals = "T", nrep = 10))
1 - Prob(A)
```

```
[1] 0.9990234
```

Compare this answer to what we got in Example 4.26.

4.7.2 Independent, Repeated Experiments

Generalizing from above it is common to repeat a certain experiment multiple times under identical conditions and in an independent manner. We have seen many examples of this already: tossing a coin repeatedly, rolling a die or dice, *etc*.

The `iidspace` function was designed specifically for this situation. It has three arguments: `x`, which is a vector of outcomes, `ntrials`, which is an integer telling how many times to repeat the experiment, and `probs` to specify the probabilities of the outcomes of `x` in a single trial.

Example 4.28 (An unbalanced coin)**.** (Continued, see Example 4.4). It was easy enough to set up the probability space for one unbalanced toss, however, the situation becomes more complicated when there are many tosses involved. Clearly, the outcome HHH should not have the same probability as TTT, which should again not have the same probability as HTH. At the same time, there is symmetry in the experiment in that the coin does not remember the face it shows from toss to toss, and it is easy enough to toss the coin in a similar way repeatedly.

We may represent tossing our unbalanced coin three times with the following:

```
iidspace(c("H","T"), ntrials = 3, probs = c(0.7, 0.3))
```

```
  X1 X2 X3 probs
1  H  H  H 0.343
2  T  H  H 0.147
```

```
3  H  T  H 0.147
4  T  T  H 0.063
5  H  H  T 0.147
6  T  H  T 0.063
7  H  T  T 0.063
8  T  T  T 0.027
```

As expected, the outcome HHH has the largest probability, while TTT has the smallest. (Since the trials are independent, $\mathbb{P}(HHH) = 0.7^3$ and $\mathbb{P}(TTT) = 0.3^3$, *etc.*) Note that the result of the function call is a probability space, not a sample space (which we could construct already with the `tosscoin` or `urnsamples` functions). The same procedure could be used to model an unbalanced die or any other experiment that may be represented with a vector of possible outcomes.

Note that `iidspace` will assume `x` has equally likely outcomes if no `probs` argument is specified. Also note that the argument `x` is a *vector*, not a data frame. Something like : iidspace(tosscoin(1),...) would give an error.

4.8 Bayes' Rule

We mentioned the subjective view of probability in Section 4.3. In this section we introduce a rule that allows us to update our probabilities when new information becomes available.

Theorem 4.2 (Bayes' Rule). Let $B_1, B_2, \ldots, B_n$ be mutually exclusive and exhaustive and let A be an event with $\mathbb{P}(A) > 0$. Then

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n} \mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1, 2, \ldots, n. \tag{4.17}$$

Proof. The proof follows from looking at $\mathbb{P}(B_k \cap A)$ in two different ways. For simplicity, suppose that $P(B_k) > 0$ for all k. Then

$$\mathbb{P}(A)\mathbb{P}(B_k|A) = \mathbb{P}(B_k \cap A) = \mathbb{P}(B_k)\mathbb{P}(A|B_k).$$

Since $\mathbb{P}(A) > 0$ we may divide through to obtain

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\mathbb{P}(A)}.$$

Now remembering that $\{B_k\}$ is a partition, the Theorem of Total Probability (Equation (4.7)) gives the denominator of the last expression to be

$$\mathbb{P}(A) = \sum_{k=1}^{n} \mathbb{P}(B_k \cap A) = \sum_{k=1}^{n} \mathbb{P}(B_k)\mathbb{P}(A|B_k).$$

□

What does it mean? Usually in applications we are given (or know) *a priori* probabilities $\mathbb{P}(B_k)$. We go out and collect some data, which we represent by the event A. We want to know: how do we *update* $\mathbb{P}(B_k)$ to $\mathbb{P}(B_k|A)$? The answer: Bayes' Rule.

Example 4.29 (Misfiling Assistants)**.** In this problem, there are three assistants working at a company: Moe, Larry, and Curly. Their primary job duty is to file paperwork in the filing cabinet when papers become available. The three assistants have different work schedules:

Table 4.3: Misfiling assistants: workload.

	Moe	Larry	Curly
Workload	60%	30%	10%

That is, Moe works 60% of the time, Larry works 30% of the time, and Curly does the remaining 10%, and they file documents at approximately the same speed. Suppose a person were to select one of the documents from the cabinet at random. Let M be the event

$$M = \{\text{Moe filed the document}\}$$

and let L and C be the events that Larry and Curly, respectively, filed the document. What are these events' respective probabilities? In the absence of additional information, reasonable prior probabilities would just be

Table 4.4: Misfiling assistants: prior.

	Moe	Larry	Curly
Prior Probability	0.6	0.3	0.1

Now, the boss comes in one day, opens up the file cabinet, and selects a file at random. The boss discovers that the file has been misplaced. The boss is so angry at the mistake that (s)he threatens to fire the one who erred. The question is: who misplaced the file?

The boss decides to use probability to decide, and walks straight to the workload schedule. (S)he reasons that, since the three employees work at the same speed, the probability that a randomly selected file would have been filed by each one would be proportional to his workload. The boss notifies *Moe* that he has until the end of the day to empty his desk.

But Moe argues in his defense that the boss has ignored additional information. Moe's likelihood of having misfiled a document is smaller than Larry's and Curly's, since he is a diligent worker who pays close attention to his work. Moe admits that he works longer than the others, but he doesn't make as many mistakes as they do. Thus, Moe recommends that – before making a decision – the boss should update the probability (initially based on workload alone) to incorporate the likelihood of having observed a misfiled document.

And, as it turns out, the boss has information about Moe, Larry, and Curly's filing accuracy in the

past (due to historical performance evaluations). The performance information may be represented by the following table:

Table 4.5: Misfiling assistants: misfile rate.

	Moe	Larry	Curly
Misfile Rate	0.003	0.007	0.010

In other words, on the average, Moe misfiles 0.3% of the documents he is supposed to file. Notice that Moe was correct: he is the most accurate filer, followed by Larry, and lastly Curly. If the boss were to make a decision based only on the worker's overall accuracy, then *Curly* should get the axe. But Curly hears this and interjects that he only works a short period during the day, and consequently makes mistakes only very rarely; there is only the tiniest chance that he misfiled this particular document.

The boss would like to use this updated information to update the probabilities for the three assistants, that is, (s)he wants to use the additional likelihood that the document was misfiled to update his/her beliefs about the likely culprit. Let A be the event that a document is misfiled. What the boss would like to know are the three probabilities

$$\mathbb{P}(M|A),\ \mathbb{P}(L|A),\ \text{and}\ \mathbb{P}(C|A).$$

We will show the calculation for $\mathbb{P}(M|A)$, the other two cases being similar. We use Bayes' Rule in the form

$$\mathbb{P}(M|A) = \frac{\mathbb{P}(M \cap A)}{\mathbb{P}(A)}.$$

Let's try to find $\mathbb{P}(M \cap A)$, which is just $\mathbb{P}(M) \cdot \mathbb{P}(A|M)$ by the Multiplication Rule. We already know $\mathbb{P}(M) = 0.6$ and $\mathbb{P}(A|M)$ is nothing more than Moe's misfile rate, given above to be $\mathbb{P}(A|M) = 0.003$. Thus, we compute

$$\mathbb{P}(M \cap A) = (0.6)(0.003) = 0.0018.$$

Using the same procedure we may calculate

$$\mathbb{P}(L \cap A) = 0.0021 \text{ and } \mathbb{P}(C \cap A) = 0.0010.$$

Now let's find the denominator, $\mathbb{P}(A)$. The key here is the notion that if a file is misplaced, then either Moe or Larry or Curly must have filed it; there is no one else around to do the misfiling. Further, these possibilities are mutually exclusive. We may use the Theorem of Total Probability (4.7) to write

$$\mathbb{P}(A) = \mathbb{P}(A \cap M) + \mathbb{P}(A \cap L) + \mathbb{P}(A \cap C).$$

Luckily, we have computed these above. Thus

$$\mathbb{P}(A) = 0.0018 + 0.0021 + 0.0010 = 0.0049.$$

Therefore, Bayes' Rule yields

$$\mathbb{P}(M|A) = \frac{0.0018}{0.0049} \approx 0.37.$$

This last quantity is called the posterior probability that Moe misfiled the document, since it incorporates the observed data that a randomly selected file was misplaced (which is governed by the misfile rate). We can use the same argument to calculate

Table 4.6: Misfiling assistants: posterior.

	Moe	Larry	Curly
Posterior Probability	≈ 0.37	≈ 0.43	≈ 0.20

The conclusion: **Larry** gets the axe. What is happening is an intricate interplay between the time on the job and the misfile rate. It is not obvious who the winner (or in this case, loser) will be, and the statistician needs to consult Bayes' Rule to determine the best course of action.

Example 4.30. Suppose the boss gets a change of heart and does not fire anybody. But the next day (s)he randomly selects another file and again finds it to be misplaced. To decide whom to fire now, the boss would use the same procedure, with one small change. (S)he would not use the prior probabilities 60%, 30%, and 10%; those are old news. Instead, she would replace the prior probabilities with the posterior probabilities just calculated. After the math she will have new posterior probabilities, updated even more from the day before.

In this way, probabilities found by Bayes' rule are always on the cutting edge, always updated with respect to the best information available at the time.

4.8.1 How to do it with R

There are not any special functions for Bayes' Rule in the `prob` package [77], but problems like the ones above are easy enough to do by hand.

Example 4.31 (Misfiling Assistants)**.** Continued from Example 4.29. We store the prior probabilities and the likelihoods in vectors and go to town.

```
prior <- c(0.6, 0.3, 0.1)
like <- c(0.003, 0.007, 0.010)
post <- prior # like
post / sum(post)
```

```
[1] 0.6 0.3 0.1
```

Compare these answers with what we got in Example 4.29. We would replace `prior` with `post` in a future calculation. We could raise `like` to a power to see how the posterior is affected by future document mistakes. (Do you see why? Think back to Section 4.7.)

Example 4.32. Let us incorporate the posterior probability (`post`) information from the last example and suppose that the assistants misfile seven more documents. Using Bayes' Rule, what would the new posterior probabilities be?

```
newprior <- post
post <- newprior * like^7
post / sum(post)
```

```
[1] 0.001051126 0.197907584 0.801041290
```

We see that the individual with the highest probability of having misfiled all eight documents given the observed data is no longer Larry, but Curly.

There are two important points. First, we did not divide `post` by the sum of its entries until the very last step; we do not need to calculate it, and it will save us computing time to postpone normalization until absolutely necessary, namely, until we finally want to interpret them as probabilities.

Second, the reader might be wondering what the boss would get if (s)he skipped the intermediate step of calculating the posterior after only one misfiled document. What if she started from the *original* prior, then observed eight misfiled documents, and calculated the posterior? What would she get? It must be the same answer, of course.

```
fastpost <- prior * like^8
fastpost / sum(fastpost)
```

```
[1] 0.0003355044 0.1473949328 0.8522695627
```

Compare this to what we got in Example 4.30.

4.9 Random Variables

We already know about experiments, sample spaces, and events. In this section, we are interested in a *number* that is associated with the experiment. We conduct a random experiment E and after learning the outcome ω in S we calculate a number X. That is, to each outcome ω in the sample space we associate a number $X(\omega) = x$.

Definition 4.4. A *random variable* X is a function $X : S \to \mathbb{R}$ that associates to each outcome $\omega \in S$ exactly one number $X(\omega) = x$.

We usually denote random variables by uppercase letters such as X, Y, and Z, and we denote their observed values by lowercase letters x, y, and z. Just as S is the set of all possible outcomes of E, we call the set of all possible values of X the *support* of X and denote it by S_X.

Example 4.33. Let E be the experiment of flipping a coin twice. We have seen that the sample space is $S = \{HH,\ HT,\ TH,\ TT\}$. Now define the random variable X = the number of heads. That is, for example, $X(HH) = 2$, while $X(HT) = 1$. We may make a table of the possibilities:

{#tab-flip-coin-twice}

Table 4.7: Flipping a coin twice.

$\omega \in S$	HH	HT	TH	TT
$X(\omega) = x$	2	1	1	0

Taking a look at the second row of the table, we see that the support of X – the set of all numbers that X assumes – would be $S_X = \{0, 1, 2\}$.

Example 4.34. Let E be the experiment of flipping a coin repeatedly until observing a Head. The sample space would be $S = \{H, TH, TTH, TTTH, \ldots\}$. Now define the random variable Y = the number of Tails before the first head. Then the support of Y would be $S_Y = \{0, 1, 2, \ldots\}$.

Example 4.35. Let E be the experiment of tossing a coin in the air, and define the random variable Z = the time (in seconds) until the coin hits the ground. In this case, the sample space is inconvenient to describe. Yet the support of Z would be $(0, \infty)$. Of course, it is reasonable to suppose that the coin will return to Earth in a short amount of time; in practice, the set $(0, \infty)$ is admittedly too large. However, we will find that in many circumstances it is mathematically convenient to study the extended set rather than a restricted one.

There are important differences between the supports of X, Y, and Z. The support of X is a finite collection of elements that can be inspected all at once. And while the support of Y cannot be exhaustively written down, its elements can nevertheless be listed in a naturally ordered sequence. Random variables with supports similar to those of X and Y are called *discrete random variables*. We study these in Chapter **??**.

In contrast, the support of Z is a continuous interval, containing all rational and irrational positive real numbers. For this reason[4], random variables with supports like Z are called *continuous random variables*, to be studied in Chapter **??**.

4.9.1 How to do it with R

The primary vessel for this task is the `addrv` function. There are two ways to use it, and we will describe both.

Supply a Defining Formula

The first method is based on the `transform` function. See `?transform`. The idea is to write a formula defining the random variable inside the function, and it will be added as a column to the data frame. As an example, let us roll a 4-sided die three times, and let us define the random variable $U = X1 - X2 + X3$.

[4]Rescaling the data gets the job done but a better way to avoid the multicollinearity introduced by the higher order terms is with *orthogonal polynomials*, whose coefficients are chosen just right so that the polynomials are not correlated with each other. This is beginning to linger outside the scope of this book, however, so we will content ourselves with a brief mention and then stick with the rescaling approach in the discussion that follows. A nice example of orthogonal polynomials in action can be run with `example(cars)`.

```
S <- rolldie(3, nsides = 4, makespace = TRUE)
S <- addrv(S, U = X1-X2+X3)
```

Now let's take a look at the values of U. In the interest of space, we will only reproduce the first few rows of S (there are $4^3 = 64$ rows in total).

```
head(S)
```

```
  X1 X2 X3 U    probs
1  1  1  1 1 0.015625
2  2  1  1 2 0.015625
3  3  1  1 3 0.015625
4  4  1  1 4 0.015625
5  1  2  1 0 0.015625
6  2  2  1 1 0.015625
```

We see from the U column it is operating just like it should. We can now answer questions like

```
Prob(S, U > 6)
```

```
[1] 0.015625
```

Supply a Function

Sometimes we have a function laying around that we would like to apply to some of the outcome variables, but it is unfortunately tedious to write out the formula defining what the new variable would be. The `addrv` function has an argument `FUN` specifically for this case. Its value should be a legitimate function from R, such as `sum`, `mean`, `median`, and so forth. Or, you can define your own function. Continuing the previous example, let's define $V = \max(X1, X2, X3)$ and $W = X1+X2+X3$.

```
S <- addrv(S, FUN = max, invars = c("X1","X2","X3"), name = "V")
S <- addrv(S, FUN = sum, invars = c("X1","X2","X3"), name = "W")
head(S)
```

```
  X1 X2 X3 U V W    probs
1  1  1  1 1 1 3 0.015625
2  2  1  1 2 2 4 0.015625
3  3  1  1 3 3 5 0.015625
4  4  1  1 4 4 6 0.015625
5  1  2  1 0 2 4 0.015625
6  2  2  1 1 2 5 0.015625
```

Notice that `addrv` has an `invars` argument to specify exactly to which columns one would like to apply the function `FUN`. If no input variables are specified, then `addrv` will apply `FUN` to all non-`probs` columns. Further, `addrv` has an optional argument `name` to give the new variable; this can be useful when adding several random variables to a probability space (as above). If not specified, the default name is `X`.

4.9.2 Marginal Distributions

As we can see above, often after adding a random variable V to a probability space one will find that V has values that are repeated, so that it becomes difficult to understand what the ultimate behavior of V actually is. We can use the `marginal` function to aggregate the rows of the sample space by values of V, all the while accumulating the probability associated with V's distinct values. Continuing our example from above, suppose we would like to focus entirely on the values and probabilities of $V = \max(X1, X2, X3)$.

```
marginal(S, vars = "V")
```

```
  V    probs
1 1 0.015625
2 2 0.109375
3 3 0.296875
4 4 0.578125
```

We could save the probability space of V in a data frame and study it further, if we wish. As a final remark, we can calculate the marginal distributions of multiple variables desired using the `vars` argument. For example, suppose we would like to examine the joint distribution of V and W.

```
marginal(S, vars = c("V", "W"))
```

```
   V  W    probs
1  1  3 0.015625
2  2  4 0.046875
3  2  5 0.046875
4  3  5 0.046875
5  2  6 0.015625
6  3  6 0.093750
7  4  6 0.046875
8  3  7 0.093750
9  4  7 0.093750
10 3  8 0.046875
11 4  8 0.140625
12 3  9 0.015625
13 4  9 0.140625
14 4 10 0.093750
15 4 11 0.046875
16 4 12 0.015625
```

Note that the default value of `vars` is the names of all columns except `probs`. This can be useful if there are duplicated rows in the probability space.

4.10 Exercises

Exercise 4.1. Prove the assertion given in the text: the number of conditions that the events A_1, A_2, ..., A_n must satisfy in order to be mutually independent is $2^n - n - 1$. (*Hint*: think about Pascal's triangle.)

Chapter 5

Simple Linear Regression

What do I want them to know?

- basic philosophy of SLR and the regression assumptions
- point and interval estimation of the model parameters, and how to use it to make predictions
- point and interval estimation of future observations from the model
- regression diagnostics, including R^2 and basic residual analysis
- the concept of influential versus outlying observations, and how to tell the difference

5.1 Basic Philosophy

Here we have two variables X and Y. For our purposes, X is not random (so we will write x), but Y is random. We believe that Y depends in *some* way on x. Some typical examples of (x, Y) pairs are

- x = study time and Y = score on a test.
- x = height and Y = weight.
- x = smoking frequency and Y = age of first heart attack.

Given information about the relationship between x and Y, we would like to *predict* future values of Y for particular values of x. This turns out to be a difficult problem[1], so instead we first tackle an easier problem: we estimate $\mathbb{E}Y$. How can we accomplish this? Well, we know that Y depends somehow on x, so it stands to reason that

$$\mathbb{E}Y = \mu(x), \text{ a function of } x. \tag{5.1}$$

But we should be able to say more than that. To focus our efforts we impose some structure on the functional form of μ. For instance,

[1] We can find solutions of the normal equations even when $\mathbf{X}^T\mathbf{X}$ is not of full rank, but the topic falls outside the scope of this book. The interested reader can consult an advanced text such as Rao [121].

- if $\mu(x) = \beta_0 + \beta_1 x$, we try to estimate β_0 and β_1.
- if $\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2$, we try to estimate β_0, β_1, and β_2.
- if $\mu(x) = \beta_0 e^{\beta_1 x}$, we try to estimate β_0 and β_1.

This helps us in the sense that we concentrate on the estimation of just a few parameters, β_0 and β_1, say, rather than some nebulous function. Our *modus operandi* is simply to perform the random experiment n times and observe the n ordered pairs of data (x_1, Y_1), (x_2, Y_2), $\ldots$, (x_n, Y_n). We use these n data points to estimate the parameters.

More to the point, there are *three simple linear regression* (SLR) assumptions that will form the basis for the rest of this chapter:

Assumption 5.1. *We assume that μ is a linear function of x, that is,*

$$\mu(x) = \beta_0 + \beta_1 x, \tag{5.2}$$

where β_0 and β_1 are unknown constants to be estimated.

Assumption 5.2. *We further assume that Y_i is $\mu(x_i)$, a "signal", plus some "error" (represented by the symbol ϵ_i):*

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, 2, \ldots, n. \tag{5.3}$$

Assumption 5.3. *We lastly assume that the errors are IID normal with mean 0 and variance σ^2:*

$$\epsilon_1, \epsilon_2, \ldots, \epsilon_n \sim \texttt{norm}(\texttt{mean} = 0,\ \texttt{sd} = \sigma). \tag{5.4}$$

Remark 5.4. We assume both the normality of the errors ϵ and the linearity of the mean function μ. Recall from Proposition **??** of Chapter **??** that if $(X, Y) \sim$ mvnorm then the mean of $Y|x$ is a linear function of x. This is not a coincidence. In more advanced classes we study the case that both X and Y are random, and in particular, when they are jointly normally distributed.

5.1.1 What does it all mean?

See Figure 5.1. Shown in the figure is a solid line, the regression line μ, which in this display has slope 0.5 and y-intercept 2.5, that is, $\mu(x) = 2.5 + 0.5x$. The intuition is that for each given value of x, we observe a random value of Y which is normally distributed with a mean equal to the height of the regression line at that x value. Normal densities are superimposed on the plot to drive this point home; in principle, the densities stand outside of the page, perpendicular to the plane of the paper. The figure shows three such values of x, namely, $x = 1$, $x = 2.5$, and $x = 4$. Not only do we assume that the observations at the three locations are independent, but we also assume that their distributions have the same spread. In mathematical terms this means that the normal densities all

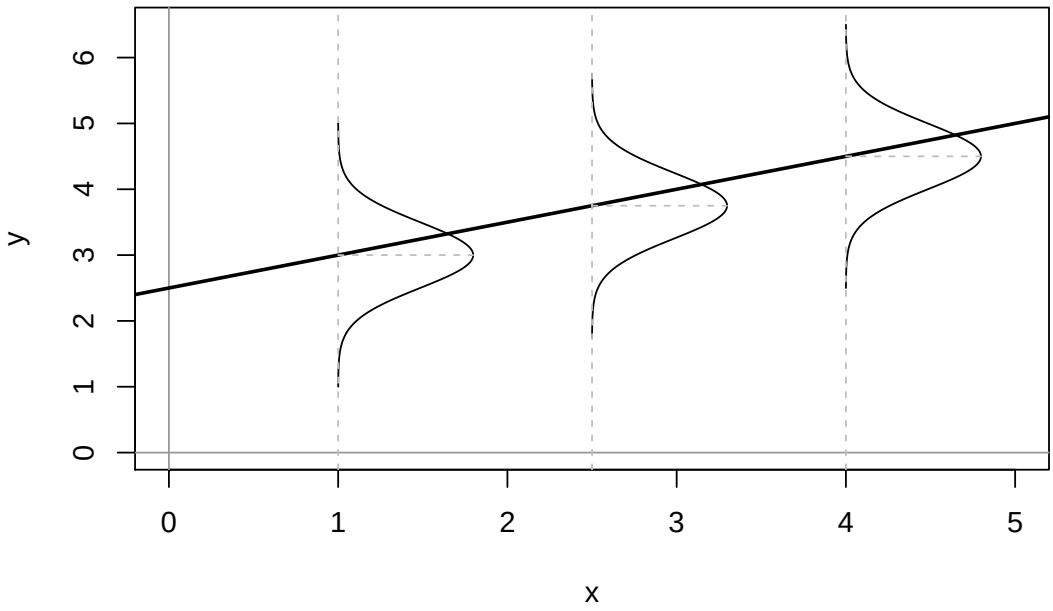

Figure 5.1: Philosophical foundations of SLR.

along the line have identical standard deviations – there is no "fanning out" or "scrunching in" of the normal densities as x increases[2].

Example 5.1 (Speed and stopping distance of cars)**.** We will use the data frame `cars` from the `datasets` package [108]. It has two variables: `speed` and `dist`. We can take a look at some of the values in the data frame:

```
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

The `speed` represents how fast the car was going (x) in miles per hour and `dist` (Y) measures how far it took the car to stop, in feet. We first make a simple scatterplot of the data.

```
plot(dist ~ speed, data = cars)
```

You can see the output in Figure 5.2, which was produced by the following code.

[2]We are taking great leaps over the mathematical details. In particular, we have yet to show that s^2 has a chi-square distribution and we have not even come close to showing that b_i and s_{b_i} are independent. But these are entirely outside the scope of the present book and the reader may rest assured that the proofs await in later classes. See C.R. Rao for more.

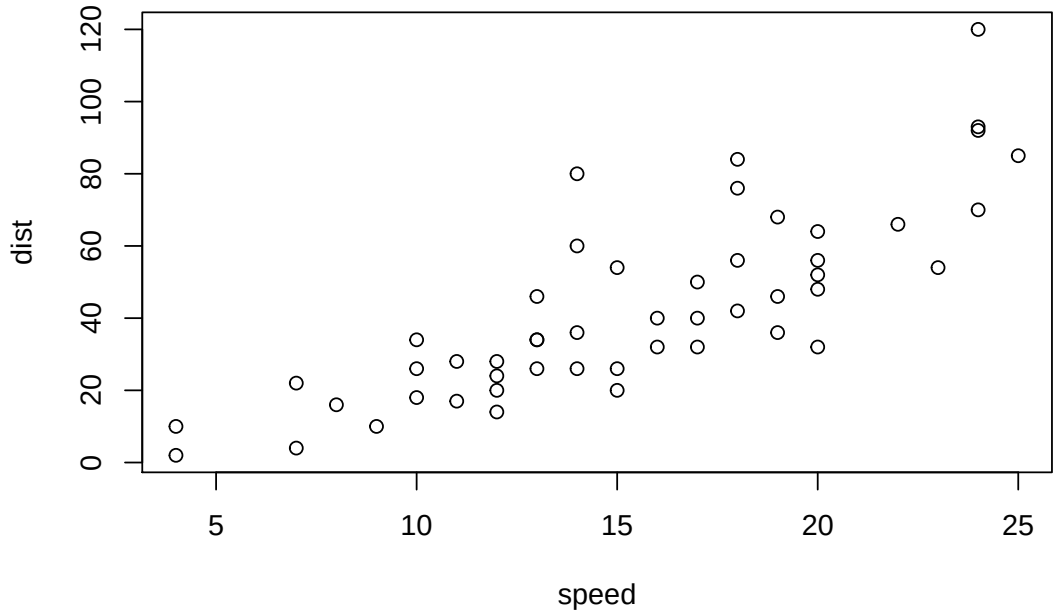

Figure 5.2: A scatterplot of `dist` versus `speed` for the `cars` data. There is clearly an upward trend to the plot which is approximately linear.

```
plot(dist ~ speed, data = cars)
```

There is a pronounced upward trend to the data points, and the pattern looks approximately linear. There does not appear to be substantial fanning out of the points or extreme values.

5.2 Estimation

5.2.1 Point Estimates of the Parameters

Where is $\mu(x)$? In essence, we would like to "fit" a line to the points. But how do we determine a "good" line? Is there a *best* line? We will use maximum likelihood to find it. We know:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, \ldots, n, \tag{5.5}$$

where the ϵ_i are IID $\mathsf{norm}(\mathtt{mean} = 0, \mathtt{sd} = \sigma)$. Thus $Y_i \sim \mathsf{norm}(\mathtt{mean} = \beta_0 + \beta_1 x_i, \mathtt{sd} = \sigma)$, $i = 1, \ldots, n$. Furthermore, $Y_1, \ldots, Y_n$ are independent – but not identically distributed. The likelihood function is:

$$L(\beta_0, \beta_1, \sigma) = \prod_{i=1}^{n} f_{Y_i}(y_i), \tag{5.6}$$

$$= \prod_{i=1}^{n} (2\pi\sigma^2)^{-1/2} \exp\left\{ \frac{-(y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2} \right\}, \tag{5.7}$$

$$= (2\pi\sigma^2)^{-n/2} \exp\left\{ \frac{-\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2} \right\}. \tag{5.8}$$

We take the natural logarithm to get

$$\ln L(\beta_0, \beta_1, \sigma) = -\frac{n}{2} \ln(2\pi\sigma^2) - \frac{\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}. \tag{5.9}$$

We would like to maximize this function of β_0 and β_1. See Appendix **??** which tells us that we should find critical points by means of the partial derivatives. Let us start by differentiating with respect to β_0:

$$\frac{\partial}{\partial \beta_0} \ln L = 0 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} 2(y_i - \beta_0 - \beta_1 x_i)(-1), \tag{5.10}$$

and the partial derivative equals zero when $\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0$, that is, when

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i. \tag{5.11}$$

Moving on, we next take the partial derivative of $\ln L$ (Equation (5.9)) with respect to β_1 to get

$$\frac{\partial}{\partial \beta_1} \ln L = 0 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} 2(y_i - \beta_0 - \beta_1 x_i)(-x_i), \tag{5.12}$$

$$= \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(x_i y_i - \beta_0 x_i - \beta_1 x_i^2 \right), \tag{5.13}$$

and this equals zero when the last sum equals zero, that is, when

$$\beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i. \tag{5.14}$$

Solving the system of equations (5.11) and (5.14)

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \tag{5.15}$$

$$\beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \tag{5.16}$$

for β_0 and β_1 (in Exercise **??**) gives

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)\Big] n}{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \Big/ n} \tag{5.17}$$

and

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}. \tag{5.18}$$

The conclusion? To estimate the mean line

$$\mu(x) = \beta_0 + \beta_1 x, \tag{5.19}$$

we use the "line of best fit"

$$\hat{\mu}(x) = \hat{\beta}_0 + \hat{\beta}_1 x, \tag{5.20}$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are given as above. For notation we will usually write $b_0 = \hat{\beta}_0$ and $b_1 = \hat{\beta}_1$ so that $\hat{\mu}(x) = b_0 + b_1 x$.

Remark 5.5. The formula for b_1 in Equation (5.17) gets the job done but does not really make any sense. There are many equivalent formulas for b_1 that are more intuitive, or at the least are easier to remember. One of the author's favorites is

$$b_1 = r\frac{s_y}{s_x}, \tag{5.21}$$

where r, s_y, and s_x are the sample correlation coefficient and the sample standard deviations of the Y and x data, respectively. See Exercise **??**. Also, notice the similarity between Equation (5.21) and Equation (**??**).

How to do it with R

Here we go. R will calculate the linear regression line with the `lm` function. We will store the result in an object which we will call `cars.lm`. Here is how it works:

```
cars.lm <- lm(dist ~ speed, data = cars)
```

The first part of the input to the `lm` function, `dist ~ speed`, is a *model formula*, read like this: `dist` is described (or modeled) by `speed`. The `data = cars` argument tells R where to look for the variables quoted in the model formula. The output object `cars.lm` contains a multitude of information. Let's first take a look at the coefficients of the fitted regression line, which are extracted by the `coef` function (alternatively, we could just type `cars.lm` to see the same thing):

```
coef(cars.lm)
```

```
(Intercept)       speed
 -17.579095    3.932409
```

The parameter estimates b_0 and b_1 for the intercept and slope, respectively, are shown above.

It is good practice to visually inspect the data with the regression line added to the plot. To do this we first scatterplot the original data and then follow with a call to the `abline` function. The inputs to `abline` are the coefficients of `cars.lm`; see Figure 5.3.

```
plot(dist ~ speed, data = cars, pch = 16)
abline(coef(cars))
```

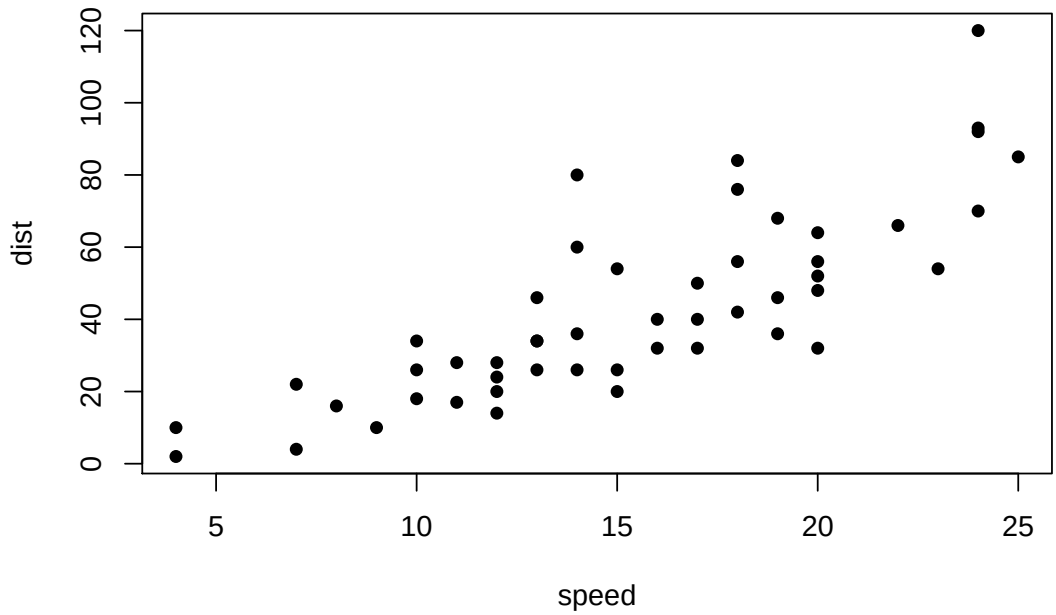

Figure 5.3: (ref:carline)

To calculate points on the regression line we may simply plug the desired x value(s) into $\hat{\mu}$, either by hand, or with the `predict` function. The inputs to `predict` are the fitted linear model object, `cars.lm`, and the desired x value(s) represented by a data frame. See the example below.

Example 5.2. Using the regression line for the `cars` data:

1. What is the meaning of $\mu(60) = \beta_0 + \beta_1(8)$? This represents the average stopping distance (in feet) for a car going 8 mph.
2. Interpret the slope β_1. The true slope β_1 represents the increase in average stopping distance for each mile per hour faster that the car drives. In this case, we estimate the car to take approximately 3.93 additional feet to stop for each additional mph increase in speed.
3. Interpret the intercept β_0. This would represent the mean stopping distance for a car traveling 0 mph (which our regression line estimates to be (-17.58. Of course, this interpretation does not make any sense for this example, because a car travelling 0 mph takes 0 ft to stop (it was not moving in the first place)! What went wrong? Looking at the data, we notice that the smallest speed for which we have measured data is 4 mph. Therefore, if we predict what would happen for slower speeds then we would be *extrapolating*, a dangerous practice which often gives nonsensical results.

5.2.2 Point Estimates of the Regression Line

We said at the beginning of the chapter that our goal was to estimate $\mu = \mathbb{E}Y$, and the arguments in Section 5.2.1 showed how to obtain an estimate $\hat{\mu}$ of μ when the regression assumptions hold. Now we will reap the benefits of our work in more ways than we previously disclosed. Given a particular value x_0, there are two values we would like to estimate:

1. the mean value of Y at x_0, and
2. a future value of Y at x_0. The first is a number, $\mu(x_0)$, and the second is a random variable, $Y(x_0)$, but our point estimate is the same for both: $\hat{\mu}(x_0)$.

Example 5.3. We may use the regression line to obtain a point estimate of the mean stopping distance for a car traveling 8 mph: $\hat{\mu}(15) = b_0 + (8)(b_1)$ which is approximately 13.88. We would also use 13.88 as a point estimate for the stopping distance of a future car traveling 8 mph.

Note that we actually have observed data for a car traveling 8 mph; its stopping distance was 16 ft as listed in the fifth row of the `cars` data (which we saw in Example **??**).

```
cars[5, ]
```

```
  speed dist
5     8   16
```

There is a special name for estimates $\hat{\mu}(x_0)$ when (x_{0}) matches an observed value x_i from the data set. They are called *fitted values*, they are denoted by $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_n$ (ignoring repetition), and they play an important role in the sections that follow.

In an abuse of notation we will sometimes write $\hat{Y}$ or $\hat{Y}(x_0)$ to denote a point on the regression line even when x_0 does not belong to the original data if the context of the statement obviates any danger of confusion.

We saw in Example 5.2 that spooky things can happen when we are cavalier about point estimation. While it is usually acceptable to predict/estimate at values of x_0 that fall within the range of the original x data, it is reckless to use $\hat{\mu}$ for point estimates at locations outside that range. Such estimates are usually worthless. *Do not extrapolate* unless there are compelling external reasons, and even then, temper it with a good deal of caution.

How to do it with R

The fitted values are automatically computed as a byproduct of the model fitting procedure and are already stored as a component of the `cars.lm` object. We may access them with the `fitted` function (we only show the first five entries):

```
fitted(cars.lm)[1:5]
```

```
        1         2         3         4         5
-1.849460 -1.849460  9.947766  9.947766 13.880175
```

Predictions at x values that are not necessarily part of the original data are done with the `predict` function. The first argument is the original `cars.lm` object and the second argument `newdata` accepts a dataframe (in the same form that was used to fit `cars.lm`) that contains the locations at which we are seeking predictions. Let us predict the average stopping distances of cars traveling 6 mph, 8 mph, and 21 mph:

```
predict(cars.lm, newdata = data.frame(speed = c(6, 8, 21)))
```

```
        1         2         3
 6.015358 13.880175 65.001489
```

Note that there were no observed cars that traveled 6 mph or 21 mph. Also note that our estimate for a car traveling 8 mph matches the value we computed by hand in Example 5.3.

5.2.3 Mean Square Error and Standard Error

To find the MLE of σ^2 we consider the partial derivative

$$\frac{\partial}{\partial \sigma^2} \ln L = \frac{n}{2\sigma^2} - \frac{1}{2(\sigma^2)^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2, \tag{5.22}$$

and after plugging in $\hat{\beta}_0$ and $\hat{\beta}_1$ and setting equal to zero we get

$$\hat{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \frac{1}{n} \sum_{i=1}^{n} [y_i - \hat{\mu}(x_i)]^2. \tag{5.23}$$

We write $\hat{Y}i = \hat{\mu}(x_i)$, and we let $E_i = Y_i - \hat{Y}_i$ be the i^{th} *residual*. We see

$$n\hat{\sigma^2} = \sum_{i=1}^{n} E_i^2 = SSE = \text{ the sum of squared errors.} \tag{5.24}$$

For a point estimate of σ^2 we use the *mean square error* S^2 defined by

$$S^2 = \frac{SSE}{n-2}, \tag{5.25}$$

and we estimate σ with the *standard error* $S = \sqrt{S^2}$[3].

How to do it with R

The residuals for the model may be obtained with the `residuals` function; we only show the first few entries in the interest of space:

```
residuals(cars.lm)[1:5]
```

```
        1         2         3         4         5 
 3.849460 11.849460 -5.947766 12.052234  2.119825 
```

In the last section, we calculated the fitted value for $x = 8$ and found it to be approximately $\hat{\mu}(8) \approx 13.88$. Now, it turns out that there was only one recorded observation at $x = 8$, and we have seen this value in the output of `head(cars)` in Example **??**; it was `dist` = 16 ft for a car with `speed` = 8 mph. Therefore, the residual should be $E = Y - \hat{Y}$ which is $E \approx 16 - 13.88$. Now take a look at the last entry of `residuals(cars.lm)`, above. It is not a coincidence.

The estimate S for σ is called the `Residual standard error` and for the `cars` data is shown a few lines up on the `summary(cars.lm)` output (see How to do it with R in Section 5.2.4). We may read it from there to be $S \approx 15.38$, or we can access it directly from the `summary` object.

```
carsumry <- summary(cars.lm)
carsumry$sigma
```

```
[1] 15.37959
```

5.2.4 Interval Estimates of the Parameters

We discussed general interval estimation in Chapter **??**. There we found that we could use what we know about the sampling distribution of certain statistics to construct confidence intervals for the parameter being estimated. We will continue in that vein, and to get started we will determine the sampling distributions of the parameter estimates, b_1 and b_0.

[3] In other words, a variable might be highly significant one moment but then fail to be significant when another variable is added to the model. When this happens it often indicates a problem with the explanatory variables, such as *multicollinearity*. See Section **??**.

To that end, we can see from Equation (5.17) (and it is made clear in Chapter **??**) that b_1 is just a linear combination of normally distributed random variables, so b_1 is normally distributed too. Further, it can be shown that

$$b_1 \sim \mathsf{norm}\,(\mathtt{mean} = \beta_1,\ \mathtt{sd} = \sigma_{b_1}) \tag{5.26}$$

where

$$\sigma_{b_1} = \frac{\sigma}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}} \tag{5.27}$$

is called *the standard error of* b_1 which unfortunately depends on the unknown value of σ. We do not lose heart, though, because we can estimate σ with the standard error S from the last section. This gives us an estimate S_{b_1} for σ_{b_1} defined by

$$S_{b_1} = \frac{S}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}}. \tag{5.28}$$

Now, it turns out that b_0, b_1, and S are mutually independent (see the footnote in Section 6.2.4). Therefore, the quantity

$$T = \frac{b_1 - \beta_1}{S_{b_1}} \tag{5.29}$$

has a $\mathsf{t}(\mathtt{df} = n - 2)$ distribution and a $100(1-\alpha)\%$ confidence interval for β_1 is given by

$$b_1 \pm \mathsf{t}_{\alpha/2}(\mathtt{df} = n - 1)\, S_{b_1}. \tag{5.30}$$

It is also sometimes of interest to construct a confidence interval for β_0 in which case we will need the sampling distribution of b_0. It is shown in Chapter **??** that

$$b_0 \sim \mathsf{norm}\,(\mathtt{mean} = \beta_0,\ \mathtt{sd} = \sigma_{b_0})\,, \tag{5.31}$$

where σ_{b_0} is given by

$$\sigma_{b_0} = \sigma\sqrt{\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}, \tag{5.32}$$

and which we estimate with the S_{b_0} defined by

$$S_{b_0} = S\sqrt{\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}. \tag{5.33}$$

Thus the quantity

$$T = \frac{b_0 - \beta_0}{S_{b_0}} \tag{5.34}$$

has a t(df $= n - 2$) distribution and a $100(1-\alpha)\%$ confidence interval for β_0 is given by

$$b_0 \pm \mathrm{t}_{\alpha/2}(\mathtt{df} = n-1)\, S_{b_0}. \tag{5.35}$$

How to do it with R

Let us take a look at the output from `summary(cars.lm)`:

```
summary(cars.lm)
```

```
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

In the `Coefficients` section we find the parameter estimates and their respective standard errors in the second and third columns; the other columns are discussed in Section 5.3. If we wanted, say, a 95% confidence interval for β_1 we could use $b_1 = 3.932$ and $S_{b_1} = 0.416$ together with a $\mathrm{t}_{0.025}(\mathtt{df} = 23)$ critical value to calculate $b_1 \pm \mathrm{t}_{0.025}(\mathtt{df} = 23) \cdot S_{b_1}$. Or, we could use the `confint` function.

```
confint(cars.lm)
```

```
                2.5 %    97.5 %
(Intercept) -31.167850 -3.990340
speed         3.096964  4.767853
```

With 95% confidence, the random interval 3.097 to 4.768 covers the parameter β_1.

5.2.5 Interval Estimates of the Regression Line

We have seen how to estimate the coefficients of regression line with both point estimates and confidence intervals. We even saw how to estimate a value $\hat{\mu}(x)$ on the regression line for a given value of x, such as $x = 15$.

But how good is our estimate $\hat{\mu}(15)$? How much confidence do we have in *this* estimate? Furthermore, suppose we were going to observe another value of Y at $x = 15$. What could we say?

Intuitively, it should be easier to get bounds on the mean (average) value of Y at x_0 – called a *confidence interval for the mean value of Y at x_0* – than it is to get bounds on a future observation of Y (called a *prediction interval for Y at x_0*). As we shall see, the intuition serves us well and confidence intervals are shorter for the mean value, longer for the individual value.

Our point estimate of $\mu(x_0)$ is of course $\hat{Y} = \hat{Y}(x_0)$, so for a confidence interval we will need to know $\hat{Y}$'s sampling distribution. It turns out (see Section) that $\hat{Y} = \hat{\mu}(x_0)$ is distributed

$$\hat{Y} \sim \texttt{norm}\left(\texttt{mean} = \mu(x_0),\ \texttt{sd} = \sigma\sqrt{\frac{1}{n} + \frac{(x_0 - \overline{x})^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}\right). \tag{5.36}$$

Since σ is unknown we estimate it with S (we should expect the appearance of a $\texttt{t}(\texttt{df} = n - 2)$ distribution in the near future).

A $100(1 - \alpha)\%$ *confidence interval (CI) for $\mu(x_0)$* is given by

$$\hat{Y} \pm \texttt{t}_{\alpha/2}(\texttt{df} = n - 2)\, S\, \sqrt{\frac{1}{n} + \frac{(x_0 - \overline{x^2})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}. \tag{5.37}$$

Prediction intervals are a little bit different. In order to find confidence bounds for a new observation of Y (we will denote it Y_{new}) we use the fact that

$$Y_{\text{new}} \sim \texttt{norm}\left(\texttt{mean} = \mu(x_0),\ \texttt{sd} = \sigma\sqrt{1 + \frac{1}{n} + \frac{(x_0 - \overline{x})^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}\right). \tag{5.38}$$

Of course, σ is unknown so we estimate it with S and a $100(1 - \alpha)\%$ prediction interval (PI) for a future value of Y at x_0 is given by

$$\hat{Y}(x_0) \pm \texttt{t}_{\alpha/2}(\texttt{df} = n - 1)\, S\, \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \overline{x})^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}. \tag{5.39}$$

We notice that the prediction interval in Equation (5.39) is wider than the confidence interval in Equation (5.37), as we expected at the beginning of the section.

How to do it with R

Confidence and prediction intervals are calculated in R with the `predict` function, which we encountered in Section 5.2.2. There we neglected to take advantage of its additional `interval` argument. The general syntax follows.

Example 5.4. We will find confidence and prediction intervals for the stopping distance of a car travelling 5, 6, and 21 mph (note from the graph that there are no collected data for these speeds). We have computed `cars.lm` earlier, and we will use this for input to the `predict` function. Also, we need to tell R the values of x_0 at which we want the predictions made, and store the x_0 values in a data frame whose variable is labeled with the correct name. *This is important*.

```
new <- data.frame(speed = c(5, 6, 21))
```

Next we instruct R to calculate the intervals. Confidence intervals are given by

```
predict(cars.lm, newdata = new, interval = "confidence")
```

```
        fit       lwr      upr
1  2.082949 -7.644150 11.81005
2  6.015358 -2.973341 15.00406
3 65.001489 58.597384 71.40559
```

Prediction intervals are given by

```
predict(cars.lm, newdata = new, interval = "prediction")
```

```
        fit       lwr      upr
1  2.082949 -30.33359 34.49948
2  6.015358 -26.18731 38.21803
3 65.001489  33.42257 96.58040
```

The type of interval is dictated by the `interval` argument (which is `none` by default), and the default confidence level is 95% (which can be changed with the `level` argument).

Example 5.5. Using the `cars` data,

1. Report a point estimate of and a 95% confidence interval for the mean stopping distance for a car travelling 5 mph. The fitted value for $x = 5$ is 2.08, so a point estimate would be 2.08 ft. The 95% CI is given by -7.64 to 11.81, so with 95% confidence the mean stopping distance lies somewhere between -7.64 ft and 11.81 ft.
2. Report a point prediction for and a 95% prediction interval for the stopping distance of a hypothetical car travelling 21 mph. The fitted value for $x = 21$ is 65, so a point prediction for the stopping distance is 65 ft. The 95% PI is 33.42 to 96.58, so with 95% confidence we may assert that the hypothetical stopping distance for a car travelling 21 mph would lie somewhere between 33.42 ft and 96.58 ft.

5.2.6 Graphing the Confidence and Prediction Bands

We earlier guessed that a bound on the value of a single new observation would be inherently less certain than a bound for an average (mean) value; therefore, we expect the CIs for the mean to be tighter than the PIs for a new observation. A close look at the standard deviations in Equations (5.37) and (5.39) confirms our guess, but we would like to see a picture to drive the point home.

We may plot the confidence and prediction intervals with one fell swoop using the `ci.plot` function from the `HH` package [58]. The graph is displayed in Figure 5.4.

```
library(HH)
ci.plot(cars.lm)
```

Notice that the bands curve outward from the regression line as the x values move away from the center. This is expected once we notice the $(x_0 - \overline{x})^2$ term in the standard deviation formulas in Equations (5.37) and (5.39).

```
print(ci.plot(cars.lm))
```

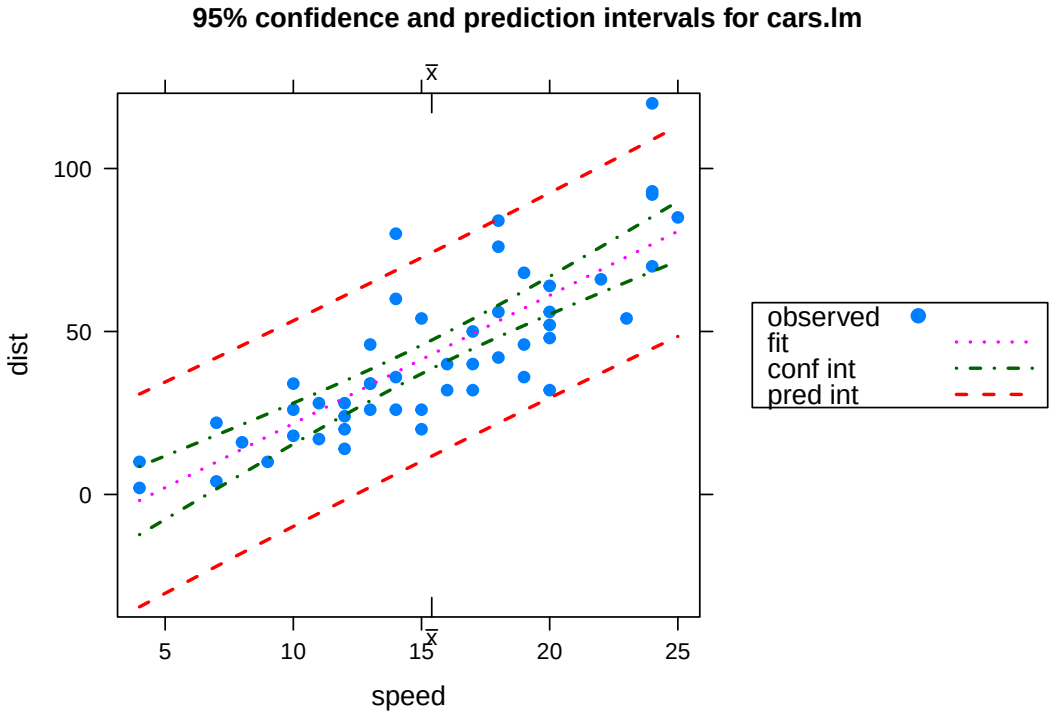

Figure 5.4: A scatterplot with confidence/prediction bands for the `cars` data.

5.3 Model Utility and Inference

5.3.1 Hypothesis Tests for the Parameters

Much of the attention of SLR is directed toward β_1 because when $\beta_1 \neq 0$ the mean value of Y increases (or decreases) as x increases. It is really boring when $\beta_1 = 0$, because in that case the mean value of Y remains the same, regardless of the value of x (when the regression assumptions hold, of course). It is thus very important to decide whether or not $\beta_1 = 0$. We address the question with a statistical test of the null hypothesis $H_0 : \beta_1 = 0$ versus the alternative hypothesis $H_1 : \beta_1 \neq 0$, and to do that we need to know the sampling distribution of b_1 when the null hypothesis is true.

To this end we already know from Section 5.2.4 that the quantity

$$T = \frac{b_1 - \beta_1}{S_{b_1}} \tag{5.40}$$

has a $\mathsf{t}(\texttt{df} = n - 2)$ distribution; therefore, when $\beta_1 = 0$ the quantity b_1/S_{b_1} has a $\mathsf{t}(\texttt{df} = n - 2)$ distribution and we can compute a p-value by comparing the observed value of b_1/S_{b_1} with values under a $\mathsf{t}(\texttt{df} = n - 2)$ curve.

Similarly, we may test the hypothesis $H_0 : \beta_0 = 0$ versus the alternative $H_1 : \beta_0 \neq 0$ with the statistic $T = b_0/S_{b_0}$, where S_{b_0} is given in Section 5.2.4. The test is conducted the same way as for β_1.

How to do it with R

Let us take another look at the output from `summary(cars.lm)`:

```
summary(cars.lm)
```

```
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

In the `Coefficients` section we find the t statistics and the p-values associated with the tests that the respective parameters are zero in the fourth and fifth columns. Since the p-values are (much) less than 0.05, we conclude that there is strong evidence that the parameters $\beta_1 \neq 0$ and $\beta_0 \neq 0$, and as such, we say that there is a statistically significant linear relationship between `dist` and `speed`.

5.3.2 Simple Coefficient of Determination

It would be nice to have a single number that indicates how well our linear regression model is doing, and the *simple coefficient of determination* is designed for that purpose. In what follows, we observe the values $Y_1, Y_2, \ldots, Y_n$, and the goal is to estimate $\mu(x_0)$, the mean value of Y at the location x_0.

If we disregard the dependence of Y and x and base our estimate only on the Y values then a reasonable choice for an estimator is just the MLE of μ, which is $\overline{Y}$. Then the errors incurred by the estimate are just $Y_i - \overline{Y}$ and the variation about the estimate as measured by the sample variance is proportional to

$$SSTO = \sum_{i=1}^{n} (Y_i - \overline{Y})^2. \tag{5.41}$$

The acronym $SSTO$ stands for *total sum of squares*. And we have additional information, namely, we have values x_i associated with each value of Y_i. We have seen that this information leads us to the estimate $\hat{Y}_i$ and the errors incurred are just the residuals, $E_i = Y_i - \hat{Y}_i$. The variation associated with these errors can be measured with

$$SSE = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2. \tag{5.42}$$

We have seen the SSE before, which stands for the *sum of squared errors* or *error sum of squares*. Of course, we would expect the error to be less in the latter case, since we have used more information. The improvement in our estimation as a result of the linear regression model can be measured with the difference

$$(Y_i - \overline{Y}) - (Y_i - \hat{Y}_i) = \hat{Y}_i - \overline{Y},$$

and we measure the variation in these errors with

$$SSR = \sum_{i=1}^{n} (\hat{Y}_i - \overline{Y})^2, \tag{5.43}$$

also known as the *regression sum of squares*. It is not obvious, but some algebra proved a famous result known as the *ANOVA Equality*:

$$\sum_{i=1}^{n} (Y_i - \overline{Y})^2 = \sum_{i=1}^{n} (\hat{Y}_i - \overline{Y})^2 + \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{5.44}$$

or in other words,

$$SSTO = SSR + SSE. \tag{5.45}$$

This equality has a nice interpretation. Consider $SSTO$ to be the *total variation* of the errors. Think of a decomposition of the total variation into pieces: one piece measuring the reduction of error from using the linear regression model, or *explained variation* (SSR), while the other represents what is left over, that is, the errors that the linear regression model doesn't explain, or *unexplained variation* (SSE). In this way we see that the ANOVA equality merely partitions the variation into

$$\text{total variation} = \text{explained variation} + \text{unexplained variation}.$$

For a single number to summarize how well our model is doing we use the *simple coefficient of determination* r^2, defined by

$$r^2 = 1 - \frac{SSE}{SSTO}. \tag{5.46}$$

We interpret r^2 as the proportion of total variation that is explained by the simple linear regression model. When r^2 is large, the model is doing a good job; when r^2 is small, the model is not doing a good job.

Related to the simple coefficient of determination is the sample correlation coefficient, r. As you can guess, the way we get r is by the formula $|r| = \sqrt{r^2}$. The sign of r is equal the sign of the slope estimate b_1. That is, if the regression line $\hat{\mu}(x)$ has positive slope, then $r = \sqrt{r^2}$. Likewise, if the slope of $\hat{\mu}(x)$ is negative, then $r = -\sqrt{r^2}$.

How to do it with R

The primary method to display partitioned sums of squared errors is with an *ANOVA table*. The command in R to produce such a table is `anova`. The input to `anova` is the result of an `lm` call which for the `cars` data is `cars.lm`.

```
anova(cars.lm)
```

```
Analysis of Variance Table

Response: dist
          Df Sum Sq Mean Sq F value   Pr(>F)
speed      1  21186 21185.5  89.567 1.49e-12 ***
Residuals 48  11354   236.5
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output gives

$$r^2 = 1 - \frac{SSE}{SSR + SSE} = 1 - \frac{11353.5}{21185.5 + 11353.5} \approx 0.65.$$

The interpretation should be: "The linear regression line accounts for approximately 65% of the variation of `dist` as explained by `speed`".

The value of r^2 is stored in the `r.squared` component of `summary(cars.lm)`, which we called `carsumry`.

```
carsumry$r.squared
```

```
[1] 0.6510794
```

We already knew this. We saw it in the next to the last line of the `summary(cars.lm)` output where it was called `Multiple R-squared`. Listed right beside it is the `Adjusted R-squared` which we will discuss in Chapter **??**. For the `cars` data, we find r to be

```
sqrt(carsumry$r.squared)
```

```
[1] 0.8068949
```

We choose the principal square root because the slope of the regression line is positive.

5.3.3 Overall F statistic

There is another way to test the significance of the linear regression model. In SLR, the new way also tests the hypothesis $H_0 : \beta_1 = 0$ versus $H_1 : \beta_1 \neq 0$, but it is done with a new test statistic called the *overall F statistic*. It is defined by

$$F = \frac{SSR}{SSE/(n-2)}. \tag{5.47}$$

Under the regression assumptions and when H_0 is true, the F statistic has an $\mathtt{f}(\mathtt{df1} = 1, \mathtt{df2} = n-2)$ distribution. We reject H_0 when F is large – that is, when the explained variation is large relative to the unexplained variation.

All this being said, we have not yet gained much from the overall F statistic because we already knew from Section 5.3.1 how to test $H_0 : \beta_1 = 0 \ldots$ we use the Student's t statistic. What is worse is that (in the simple linear regression model) it can be proved that the F in Equation (5.47) is exactly the Student's t statistic for β_1 squared,

$$F = \left(\frac{b_1}{S_{b_1}}\right)^2. \tag{5.48}$$

So why bother to define the F statistic? Why not just square the t statistic and be done with it? The answer is that the F statistic has a more complicated interpretation and plays a more important role in the multiple linear regression model which we will study in Chapter **??**. See Section 6.3.2 for details.

How to do it with R

The overall F statistic and p-value are displayed in the bottom line of the `summary(cars.lm)` output. It is also shown in the final columns of `anova(cars.lm)`:

```
anova(cars.lm)
```

```
Analysis of Variance Table

Response: dist
          Df Sum Sq Mean Sq F value   Pr(>F)
speed      1  21186 21185.5  89.567 1.49e-12 ***
Residuals 48  11354   236.5
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we see that the F statistic is 89.57 with a p-value very close to zero. The conclusion: there is very strong evidence that $H_0 : \beta_1 = 0$ is false, that is, there is strong evidence that $\beta_1 \neq 0$. Moreover, we conclude that the regression relationship between `dist` and `speed` is significant.

Note that the value of the F statistic is the same as the Student's t statistic for `speed` squared.

5.4 Residual Analysis

We know from our model that $Y = \mu(x) + \epsilon$, or in other words, $\epsilon = Y - \mu(x)$. Further, we know that $\epsilon \sim \text{norm}(\texttt{mean} = 0, \texttt{sd} = \sigma)$. We may estimate ϵ_i with the *residual* $E_i = Y_i - \hat{Y}_i$, where $\hat{Y}_i = \hat{\mu}(x_i)$. If the regression assumptions hold, then the residuals should be normally distributed. We check this in Section 5.4.1. Further, the residuals should have mean zero with constant variance σ^2, and we check this in Section 5.4.2. Last, the residuals should be independent, and we check this in Section 5.4.3.

In every case, we will begin by looking at residual plots – that is, scatterplots of the residuals E_i versus index or predicted values $\hat{Y}_i$ – and follow up with hypothesis tests.

5.4.1 Normality Assumption

We can assess the normality of the residuals with graphical methods and hypothesis tests. To check graphically whether the residuals are normally distributed we may look at histograms or q-q plots. We first examine a histogram in Figure 5.5, which was produced by the following code.

```
hist(residuals(cars.lm))
```

There we see that the distribution of the residuals appears to be mound shaped, for the most part. We can plot the order statistics of the studentized residuals versus quantiles from a $\text{t}(\texttt{mean} = 0, \texttt{sd} = 1)$ distribution with the command `qqPlot(cars.lm)` from the `RcmdrMisc` package [**RcmdrMisc**],

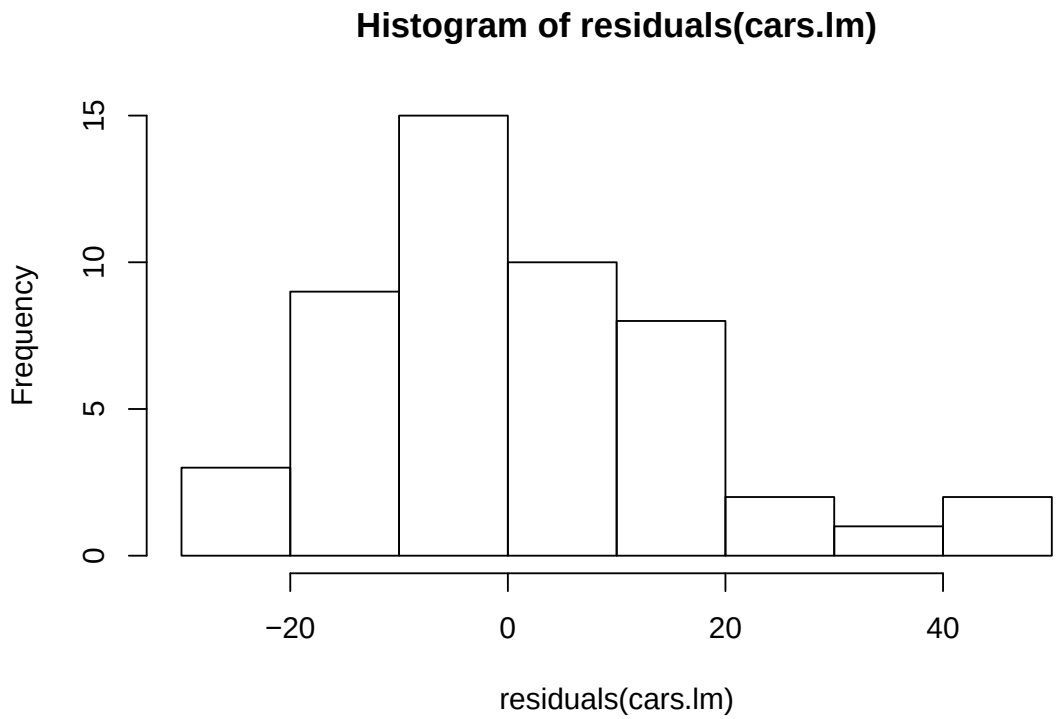

Figure 5.5: A histogram of the residuals from the linear regression model for the `cars` data. Used for checking the normality assumption. Look out for any skewness or extreme values; hopefully the graph is mound shaped and symmetric about zero, with no outliers.

and the results are in Figure 5.6. If the assumption of normality were true, then we would expect points randomly scattered about the dotted straight line displayed in the figure. In this case, we see a slight departure from normality in that the dots show a bit of clustering on one side or the other of the line. The points on the upper end of the plot also begin to stray from the line. We would say there is some evidence that the residuals are not perfectly normal.

```
library(RcmdrMisc)
qqPlot(cars.lm)
```

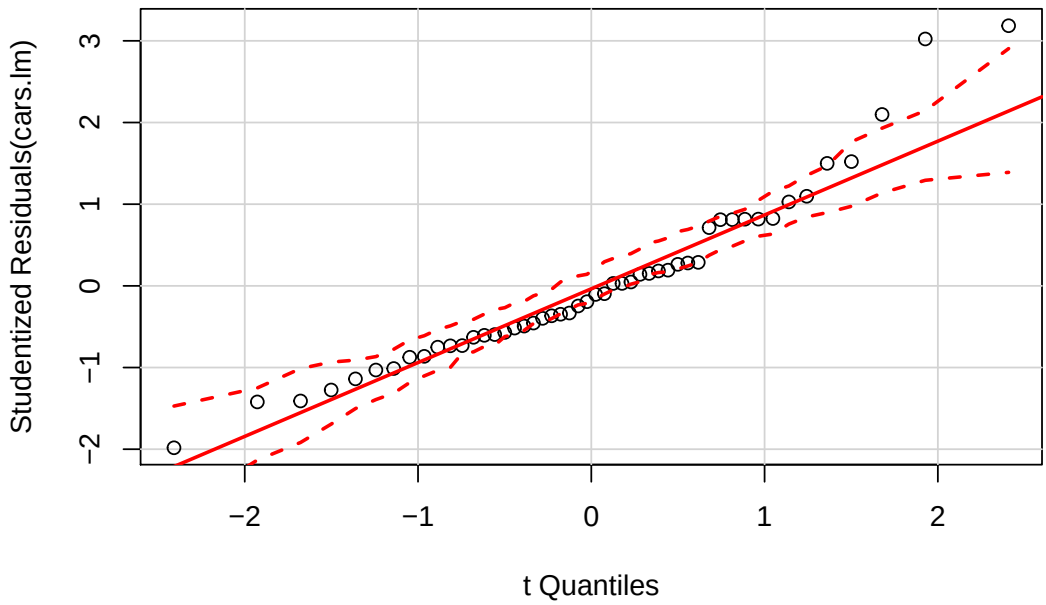

Figure 5.6: A quantile-quantile plot of the studentized residuals from the linear regression model for the `cars` data. Used for checking the normality assumption. Look out for any curvature or substantial departures from the straight line; hopefully the dots hug the line closely and stay within the bands.

Testing the Normality Assumption

Even though we may be concerned about the plots, we can use tests to determine if the evidence present is statistically significant, or if it could have happened merely by chance. There are many statistical tests of normality. We will use the Shapiro-Wilk test, since it is known to be a good test and to be quite powerful. However, there are many other fine tests of normality including the Anderson-Darling test and the Lillefors test, just to mention two of them.

The Shapiro-Wilk test is based on the statistic

$$W = \frac{\left(\sum_{i=1}^{n} a_i E_{(i)}\right)^2}{\sum_{j=1}^{n} E_j^2}, \tag{5.49}$$

where the $E_{(i)}$ are the ordered residuals and the a_i are constants derived from the order statistics of a sample of size n from a normal distribution. See Section **??**. We perform the Shapiro-Wilk test below, using the `shapiro.test` function from the `stats` package [115]. The hypotheses are

$$H_0: \text{ the residuals are normally distributed}$$

versus

$$H_1: \text{ the residuals are not normally distributed.}$$

The results from R are

```
shapiro.test(residuals(cars.lm))
```

```
    Shapiro-Wilk normality test

data:  residuals(cars.lm)
W = 0.94509, p-value = 0.02152
```

For these data we would reject the assumption of normality of the residuals at the $\alpha = 0.05$ significance level, but do not lose heart, because the regression model is reasonably robust to departures from the normality assumption. As long as the residual distribution is not highly skewed, then the regression estimators will perform reasonably well. Moreover, departures from constant variance and independence will sometimes affect the quantile plots and histograms, therefore it is wise to delay final decisions regarding normality until all diagnostic measures have been investigated.

5.4.2 Constant Variance Assumption

We will again go to residual plots to try and determine if the spread of the residuals is changing over time (or index). However, it is unfortunately not that easy because the residuals do not have constant variance! In fact, it can be shown that the variance of the residual E_i is

$$\text{Var}(E_i) = \sigma^2(1 - h_{ii}), \quad i = 1, 2, \ldots, n, \tag{5.50}$$

where h_{ii} is a quantity called the *leverage* which is defined below. Consequently, in order to check the constant variance assumption we must standardize the residuals before plotting. We estimate the standard error of E_i with $s_{E_i} = s\sqrt{(1 - h_{ii})}$ and define the *standardized residuals* R_i, $i = 1, 2, \ldots, n$, by

$$R_i = \frac{E_i}{s\sqrt{1 - h_{ii}}}, \quad i = 1, 2, \ldots, n. \tag{5.51}$$

For the constant variance assumption we do not need the sign of the residual so we will plot $\sqrt{|R_i|}$ versus the fitted values. As we look at a scatterplot of $\sqrt{|R_i|}$ versus $\hat{Y}_i$ we would expect under the regression assumptions to see a constant band of observations, indicating no change in the magnitude of the observed distance from the line. We want to watch out for a fanning-out of the residuals, or a less common funneling-in of the residuals. Both patterns indicate a change in the residual variance

and a consequent departure from the regression assumptions, the first an increase, the second a decrease.

In this case, we plot the standardized residuals versus the fitted values. The graph may be seen in Figure 5.7. For these data there does appear to be somewhat of a slight fanning-out of the residuals.

```
plot(cars.lm, which = 3)
```

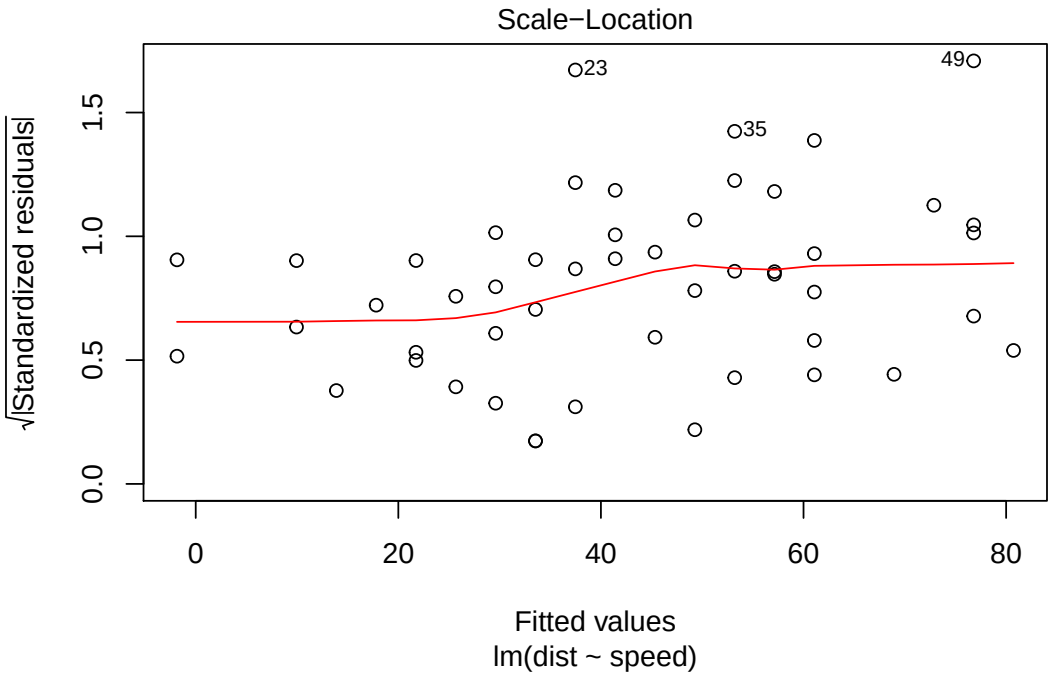

Figure 5.7: Plot of standardized residuals against the fitted values for the `cars` data. Used for checking the constant variance assumption. Watch out for any fanning out (or in) of the dots; hopefully they fall in a constant band.

Testing the Constant Variance Assumption

We will use the Breusch-Pagan test to decide whether the variance of the residuals is nonconstant. The null hypothesis is that the variance is the same for all observations, and the alternative hypothesis is that the variance is not the same for all observations. The test statistic is found by fitting a linear model to the centered squared residuals,

$$W_i = E_i^2 - \frac{SSE}{n}, \quad i = 1, 2, \ldots, n. \tag{5.52}$$

By default the same explanatory variables are used in the new model which produces fitted values $\hat{W}_i$, $i = 1, 2, \ldots, n$. The Breusch-Pagan test statistic in R is then calculated with

$$BP = n \sum_{i=1}^{n} \hat{W}_i^2 \div \sum_{i=1}^{n} W_i^2. \tag{5.53}$$

We reject the null hypothesis if BP is too large, which happens when the explained variation i the new model is large relative to the unexplained variation in the original model. We do it in R with the `bptest` function from the `lmtest` package [189].

```
bptest(cars.lm)
```

```
    studentized Breusch-Pagan test

data:  cars.lm
BP = 3.2149, df = 1, p-value = 0.07297
```

For these data we would not reject the null hypothesis at the $\alpha = 0.05$ level. There is relatively weak evidence against the assumption of constant variance.

5.4.3 Independence Assumption

One of the strongest of the regression assumptions is the one regarding independence. Departures from the independence assumption are often exhibited by correlation (or autocorrelation, literally, self-correlation) present in the residuals. There can be positive or negative correlation.

Positive correlation is displayed by positive residuals followed by positive residuals, and negative residuals followed by negative residuals. Looking from left to right, this is exhibited by a cyclical feature in the residual plots, with long sequences of positive residuals being followed by long sequences of negative ones.

On the other hand, negative correlation implies positive residuals followed by negative residuals, which are then followed by positive residuals, *etc*. Consequently, negatively correlated residuals are often associated with an alternating pattern in the residual plots. We examine the residual plot in Figure 5.8. There is no obvious cyclical wave pattern or structure to the residual plot.

```
plot(cars.lm, which = 1)
```

Testing the Independence Assumption

We may statistically test whether there is evidence of autocorrelation in the residuals with the Durbin-Watson test. The test is based on the statistic

$$D = \frac{\sum_{i=2}^{n}(E_i - E_{i-1})^2}{\sum_{j=1}^{n} E_j^2}. \tag{5.54}$$

Exact critical values are difficult to obtain, but R will calculate the p-value to great accuracy. It is performed with the `dwtest` function from the `lmtest` package [189]. We will conduct a two

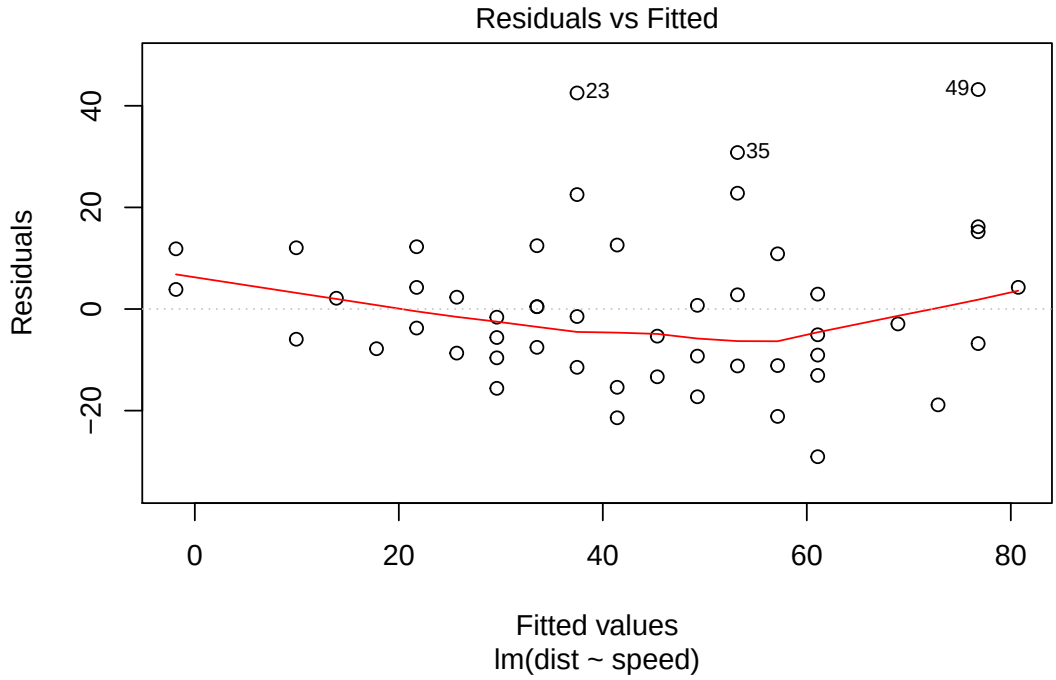

Figure 5.8: Plot of the residuals versus the fitted values for the `cars` data. Used for checking the independence assumption. Watch out for any patterns or structure; hopefully the points are randomly scattered on the plot.

sided test that the correlation is not zero, which is not the default (the default is to test that the autocorrelation is positive).

```
dwtest(cars.lm, alternative = "two.sided")
```

```
    Durbin-Watson test

data:  cars.lm
DW = 1.6762, p-value = 0.1904
alternative hypothesis: true autocorrelation is not 0
```

In this case we do not reject the null hypothesis at the $\alpha = 0.05$ significance level; there is very little evidence of nonzero autocorrelation in the residuals.

5.4.4 Remedial Measures

We often find problems with our model that suggest that at least one of the three regression assumptions is violated. What do we do then? There are many measures at the statistician's disposal, and we mention specific steps one can take to improve the model under certain types of violation.

- **Mean response is not linear** We can directly modify the model to better approximate the mean response. In particular, perhaps a polynomial regression function of the form

  $$\mu(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$$

  would be appropriate. Alternatively, we could have a function of the form

  $$\mu(x) = \beta_0 e^{\beta_1 x}.$$

  Models like these are studied in nonlinear regression courses.
- **Error variance is not constant** Sometimes a transformation of the dependent variable will take care of the problem. There is a large class of them called *Box-Cox transformations*. They take the form

  $$Y^* = Y^\lambda, \tag{5.55}$$

  where λ is a constant. (The method proposed by Box and Cox will determine a suitable value of λ automatically by maximum likelihood). The class contains the transformations

  $$\begin{aligned} \lambda = 2, &\quad Y^* = Y^2 \\ \lambda = 0.5, &\quad Y^* = \sqrt{Y} \\ \lambda = 0, &\quad Y^* = \ln Y \\ \lambda = -1, &\quad Y^* = 1/Y \end{aligned}$$

  Alternatively, we can use the method of *weighted least squares*. This is studied in more detail in later classes.

- **Error distribution is not normal** The same transformations for stabilizing the variance are equally appropriate for smoothing the residuals to a more Gaussian form. In fact, often we will kill two birds with one stone.
- **Errors are not independent** There is a large class of autoregressive models to be used in this situation which occupy the latter part of Chapter **??**.

5.5 Other Diagnostic Tools

There are two types of observations with which we must be especially careful:

- **Influential observations** are those that have a substantial effect on our estimates, predictions, or inferences. A small change in an influential observation is followed by a large change in the parameter estimates or inferences.
- **Outlying observations** are those that fall fall far from the rest of the data. They may be indicating a lack of fit for our regression model, or they may just be a mistake or typographical error that should be corrected. Regardless, special attention should be given to these observations. An outlying observation may or may not be influential.

We will discuss outliers first because the notation builds sequentially in that order.

5.5.1 Outliers

There are three ways that an observation (x_i, y_i) might be identified as an outlier: it can have an x_i value which falls far from the other x values, it can have a y_i value which falls far from the other y values, or it can have both its x_i and y_i values falling far from the other x and y values.

5.5.2 Leverage

Leverage statistics are designed to identify observations which have x values that are far away from the rest of the data. In the simple linear regression model the leverage of x_i is denoted by h_{ii} and defined by

$$h_{ii} = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{k=1}^{n}(x_k - \overline{x})^2}, \quad i = 1, 2, \ldots, n. \tag{5.56}$$

The formula has a nice interpretation in the SLR model: if the distance from x_i to $\overline{x}$ is large relative to the other x's then h_{ii} will be close to 1.

Leverages have nice mathematical properties; for example, they satisfy

$$0 \leq h_{ii} \leq 1, \tag{5.57}$$

and their sum is

$$\sum_{i=1}^{n} h_{ii} = \sum_{i=1}^{n} \left[\frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{k=1}^{n} (x_k - \overline{x})^2} \right], \tag{5.58}$$

$$= \frac{n}{n} + \frac{\sum_i (x_i - \overline{x})^2}{\sum_k (x_k - \overline{x})^2}, \tag{5.59}$$

$$= 2. \tag{5.60}$$

A rule of thumb is to consider leverage values to be large if they are more than double their average size (which is $2/n$ according to Equation (5.58)). So leverages larger than $4/n$ are suspect. Another rule of thumb is to say that values bigger than 0.5 indicate high leverage, while values between 0.3 and 0.5 indicate moderate leverage.

5.5.3 Standardized and Studentized Deleted Residuals

We have already encountered the *standardized residuals* r_i in Section 5.4.2; they are merely residuals that have been divided by their respective standard deviations:

$$R_i = \frac{E_i}{S\sqrt{1 - h_{ii}}}, \quad i = 1, 2, \ldots, n. \tag{5.61}$$

Values of $|R_i| > 2$ are extreme and suggest that the observation has an outlying y-value.

Now delete the i^{th} case and fit the regression function to the remaining $n - 1$ cases, producing a fitted value $\hat{Y}_{(i)}$ with *deleted residual* $D_i = Y_i - \hat{Y}_{(i)}$. It is shown in later classes that

$$\text{Var}\,(D_i) = \frac{S^2_{(i)}}{1 - h_{ii}}, \quad i = 1, 2, \ldots, n, \tag{5.62}$$

so that the *studentized deleted residuals* t_i defined by

$$t_i = \frac{D_i}{S_{(i)}/(1 - h_{ii})}, \quad i = 1, 2, \ldots, n, \tag{5.63}$$

have a $\mathsf{t}(\texttt{df} = n - 3)$ distribution and we compare observed values of t_i to this distribution to decide whether or not an observation is extreme.

The folklore in regression classes is that a test based on the statistic in Equation (5.63) can be too liberal. A rule of thumb is if we suspect an observation to be an outlier *before* seeing the data then we say it is significantly outlying if its two-tailed p-value is less than α, but if we suspect an observation to be an outlier *after* seeing the data then we should only say it is significantly outlying if its two-tailed p-value is less than α/n. The latter rule of thumb is called the *Bonferroni approach* and can be overly conservative for large data sets. The responsible statistician should look at the data and use his/her best judgement, in every case.

How to do it with R

We can calculate the standardized residuals with the `rstandard` function. The input is the `lm` object, which is `cars.lm`.

```
sres <- rstandard(cars.lm)
sres[1:5]
```

```
         1          2          3          4          5
 0.2660415  0.8189327 -0.4013462  0.8132663  0.1421624
```

We can find out which observations have studentized residuals larger than two with the command

```
sres[which(abs(sres) > 2)]
```

```
      23       35       49
2.795166 2.027818 2.919060
```

In this case, we see that observations 23, 35, and 49 are potential outliers with respect to their y-value. We can compute the studentized deleted residuals with `rstudent`:

```
sdelres <- rstudent(cars.lm)
sdelres[1:5]
```

```
         1          2          3          4          5
 0.2634500  0.8160784 -0.3978115  0.8103526  0.1407033
```

We should compare these values with critical values from a t(`df` $= n - 3$) distribution, which in this case is t(`df` $= 50 - 3 = 47$). We can calculate a 0.005 quantile and check with

```
t0.005 <- qt(0.005, df = 47, lower.tail = FALSE)
sdelres[which(abs(sdelres) > t0.005)]
```

```
      23       49
3.022829 3.184993
```

This means that observations 23 and 49 have a large studentized deleted residual. The leverages can be found with the `hatvalues` function:

```
leverage <- hatvalues(cars.lm)
leverage[which(leverage > 4/50)]
```

```
         1          2         50
0.11486131 0.11486131 0.08727007
```

Here we see that observations 1, 2, and 50 have leverages bigger than double their mean value. These observations would be considered outlying with respect to their x value (although they may or may not be influential).

5.5.4 Influential Observations

DFBETAS **and** *DFFITS*

Any time we do a statistical analysis, we are confronted with the variability of data. It is always a concern when an observation plays too large a role in our regression model, and we would not like or procedures to be overly influenced by the value of a single observation. Hence, it becomes desirable to check to see how much our estimates and predictions would change if one of the observations were not included in the analysis. If an observation changes the estimates/predictions a large amount, then the observation is influential and should be subjected to a higher level of scrutiny.

We measure the change in the parameter estimates as a result of deleting an observation with *DFBETAS*. The *DFBETAS* for the intercept b_0 are given by

$$(DFBETAS)_{0(i)} = \frac{b_0 - b_{0(i)}}{S_{(i)}\sqrt{\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}}, \quad i = 1, 2, \ldots, n. \tag{5.64}$$

and the *DFBETAS* for the slope b_1 are given by

$$(DFBETAS)_{1(i)} = \frac{b_1 - b_{1(i)}}{S_{(i)}\left[\sum_{i=1}^{n}(x_i - \overline{x})^2\right]^{-1/2}}, \quad i = 1, 2, \ldots, n. \tag{5.65}$$

See Section 6.6 for a better way to write these. The signs of the *DFBETAS* indicate whether the coefficients would increase or decrease as a result of including the observation. If the *DFBETAS* are large, then the observation has a large impact on those regression coefficients. We label observations as suspicious if their *DFBETAS* have magnitude greater 1 for small data or $2/\sqrt{n}$ for large data sets. We can calculate the *DFBETAS* with the `dfbetas` function (some output has been omitted):

```
dfb <- dfbetas(cars.lm)
head(dfb)
```

```
  (Intercept)       speed
1  0.09440188 -0.08624563
2  0.29242487 -0.26715961
3 -0.10749794  0.09369281
4  0.21897614 -0.19085472
5  0.03407516 -0.02901384
6 -0.11100703  0.09174024
```

We see that the inclusion of the first observation slightly increases the `Intercept` and slightly decreases the coefficient on `speed`.

We can measure the influence that an observation has on its fitted value with *DFFITS*. These are calculated by deleting an observation, refitting the model, recalculating the fit, then standardizing. The formula is

$$(DFFITS)_i = \frac{\hat{Y}_i - \hat{Y}_{(i)}}{S_{(i)}\sqrt{h_{ii}}}, \quad i = 1, 2, \ldots, n. \tag{5.66}$$

The value represents the number of standard deviations of $\hat{Y}_i$ that the fitted value $\hat{Y}_i$ increases or decreases with the inclusion of the i^{th} observation. We can compute them with the `dffits` function.

```
dff <- dffits(cars.lm)
dff[1:5]
```

```
          1           2           3           4           5
 0.09490289  0.29397684 -0.11039550  0.22487854  0.03553887
```

A rule of thumb is to flag observations whose $DFFIT$ exceeds one in absolute value, but there are none of those in this data set.

Cook's Distance

The $DFFITS$ are good for measuring the influence on a single fitted value, but we may want to measure the influence an observation has on all of the fitted values simultaneously. The statistics used for measuring this are Cook's distances which may be calculated[4] by the formula

$$D_i = \frac{E_i^2}{(p+1)S^2} \cdot \frac{h_{ii}}{(1-h_{ii})^2}, \quad i = 1, 2, \ldots, n. \tag{5.67}$$

It shows that Cook's distance depends both on the residual E_i and the leverage h_{ii} and in this way D_i contains information about outlying x and y values.

To assess the significance of D, we compare to quantiles of an $\mathsf{f}(\texttt{df1} = 2, \texttt{df2} = n - 2)$ distribution. A rule of thumb is to classify observations falling higher than the 50^{th} percentile as being extreme.

How to do it with R

We can calculate the Cook's Distances with the `cooks.distance` function.

```
cooksD <- cooks.distance(cars.lm)
cooksD[1:4]
```

```
          1           2           3           4
0.004592312 0.043513991 0.006202350 0.025467338
```

[4] Rescaling the data gets the job done but a better way to avoid the multicollinearity introduced by the higher order terms is with *orthogonal polynomials*, whose coefficients are chosen just right so that the polynomials are not correlated with each other. This is beginning to linger outside the scope of this book, however, so we will content ourselves with a brief mention and then stick with the rescaling approach in the discussion that follows. A nice example of orthogonal polynomials in action can be run with `example(cars)`.

We can look at a plot of the Cook's distances with the command `plot(cars.lm, which = 4)`.

```
plot(cars.lm, which = 4)
```

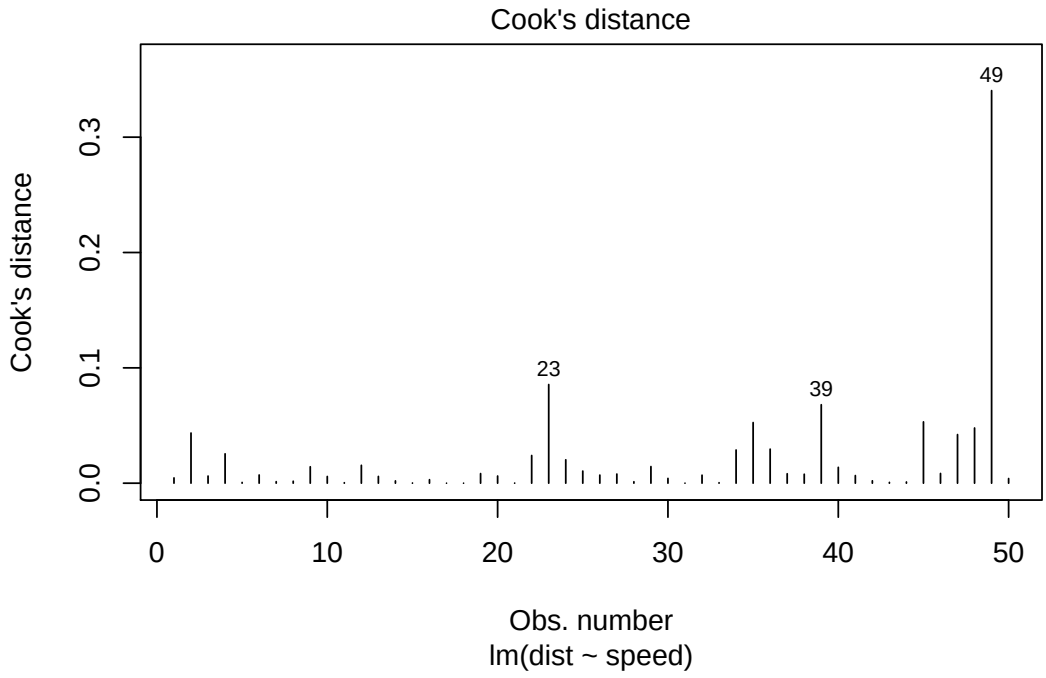

Figure 5.9: Cook's distances for the `cars` data. Used for checking for influential and/our outlying observations. Values with large Cook's distance merit further investigation.

Observations with the largest Cook's D values are labeled, hence we see that observations 23, 39, and 49 are suspicious. However, we need to compare to the quantiles of an f(`df1` = 2, `df2` = 48) distribution:

```
F0.50 <- qf(0.5, df1 = 2, df2 = 48)
any(cooksD > F0.50)
```

```
[1] FALSE
```

We see that with this data set there are no observations with extreme Cook's distance, after all.

5.5.5 All Influence Measures Simultaneously

We can display the result of diagnostic checking all at once in one table, with potentially influential points displayed. We do it with the command `influence.measures(cars.lm)`:

```
influence.measures(cars.lm)
```

The output is a huge matrix display, which we have omitted in the interest of brevity. A point is identified if it is classified to be influential with respect to any of the diagnostic measures. Here we see that observations 2, 11, 15, and 18 merit further investigation.

We can also look at all diagnostic plots at once with the commands

```
plot(cars.lm)
```

The `par` command is used so that $2 \times 2 = 4$ plots will be shown on the same display. The diagnostic plots for the `cars` data are shown in Figure 5.10.

```
par(mfrow = c(2,2))
plot(cars.lm)
```

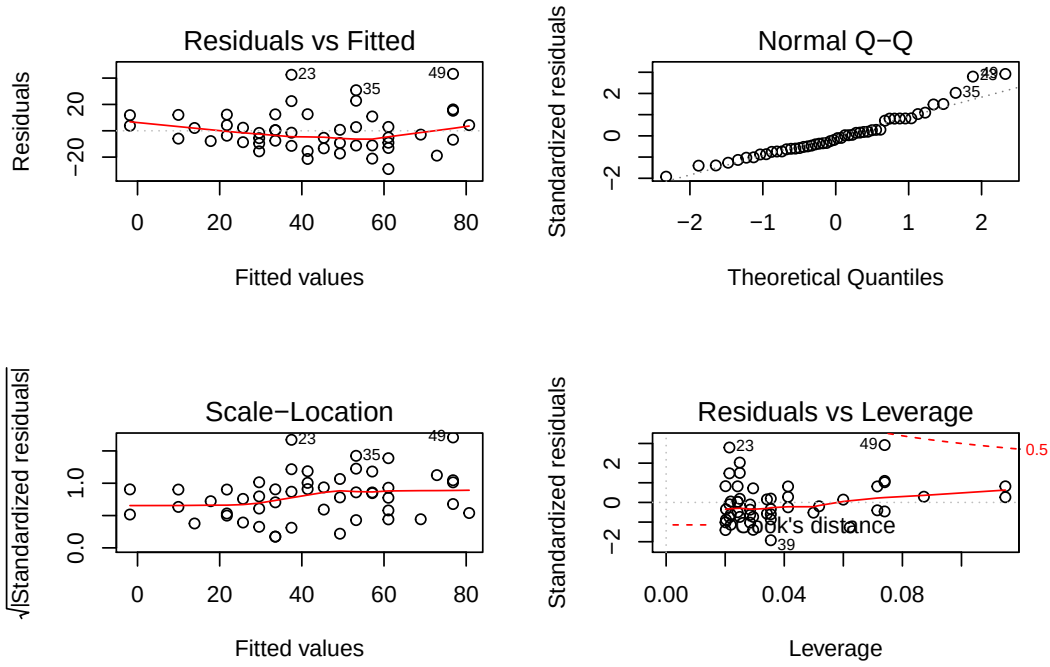

Figure 5.10: Diagnostic plots for the `cars` data.

```
par(mfrow = c(1,1))
```

We have discussed all of the plots except the last, which is possibly the most interesting. It shows Residuals vs. Leverage, which will identify outlying y values versus outlying x values. Here we see that observation 23 has a high residual, but low leverage, and it turns out that observations 1 and 2 have relatively high leverage but low/moderate leverage (they are on the right side of the plot, just above the horizontal line). Observation 49 has a large residual with a comparatively large leverage.

We can identify the observations with the `identify` command; it allows us to display the observation number of dots on the plot. First, we plot the graph, then we call `identify`:

```
plot(cars.lm, which = 5)           # std'd resids vs lev plot
identify(leverage, sres, n = 4)    # identify 4 points
```

The graph with the identified points is omitted (but the plain plot is shown in the bottom right corner of Figure 5.10). Observations 1 and 2 fall on the far right side of the plot, near the horizontal axis.

5.6 Exercises

Exercise 5.1. Prove the ANOVA equality, Equation (5.44). *Hint*: show that

$$\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(\hat{Y}_i - \overline{Y}) = 0.$$

Exercise 5.2. Solve the following system of equations for β_1 and β_0 to find the MLEs for slope and intercept in the simple linear regression model.

$$\begin{aligned} n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} y_i \\ \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} x_i y_i \end{aligned}$$

Exercise 5.3. Show that the formula given in Equation (5.21) is equivalent to

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)\Big/ n}{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \Big/ n}.$$

Chapter 6

Multiple Linear Regression

We know a lot about simple linear regression models, and a next step is to study multiple regression models that have more than one independent (explanatory) variable. In the discussion that follows we will assume that we have p explanatory variables, where $p > 1$.

The language is phrased in matrix terms – for two reasons. First, it is quicker to write and (arguably) more pleasant to read. Second, the matrix approach will be required for later study of the subject; the reader might as well be introduced to it now.

Most of the results are stated without proof or with only a cursory justification. Those yearning for more should consult an advanced text in linear regression for details, such as *Applied Linear Regression Models* [99] or *Linear Models: Least Squares and Alternatives* [121].

What do I want them to know?

- the basic MLR model, and how it relates to the SLR
- how to estimate the parameters and use those estimates to make predictions
- basic strategies to determine whether or not the model is doing a good job
- a few thoughts about selected applications of the MLR, such as polynomial, interaction, and dummy variable models
- some of the uses of residuals to diagnose problems
- hints about what will be coming later

6.1 The Multiple Linear Regression Model

The first thing to do is get some better notation. We will write

$$\mathbf{Y}_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \text{and} \quad \mathbf{X}_{n\times(p+1)} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{p1} \\ 1 & x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{pn} \end{bmatrix}. \tag{6.1}$$

The vector $\mathbf{Y}$ is called the *response vector* and the matrix $\mathbf{X}$ is called the *model matrix* . As in Chapter 5, the most general assumption that relates $\mathbf{Y}$ to $\mathbf{X}$ is

$$\mathbf{Y} = \mu(\mathbf{X}) + \epsilon, \tag{6.2}$$

where μ is some function (the *signal*) and ϵ is the *noise* (everything else). We usually impose some structure on μ and ϵ. In particular, the standard multiple linear regression model assumes

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon, \tag{6.3}$$

where the parameter vector β looks like

$$\beta_{(p+1)\times 1} = \begin{bmatrix} \beta_0 & \beta_1 & \cdots & \beta_p \end{bmatrix}^{\mathrm{T}}, \tag{6.4}$$

and the random vector $\epsilon_{n\times 1} = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \cdots & \epsilon_n \end{bmatrix}^{\mathrm{T}}$ is assumed to be distributed

$$\epsilon \sim \mathsf{mvnorm}\left(\mathtt{mean} = \mathbf{0}_{n\times 1},\ \mathtt{sigma} = \sigma^2 \mathbf{I}_{n\times n}\right). \tag{6.5}$$

The assumption on ϵ is equivalent to the assumption that $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are IID $\mathsf{norm}(\mathtt{mean} = 0,\ \mathtt{sd} = \sigma)$. It is a linear model because the quantity $\mu(\mathbf{X}) = \mathbf{X}\beta$ is linear in the parameters $\beta_0, \beta_1, \ldots, \beta_p$. It may be helpful to see the model in expanded form; the above matrix formulation is equivalent to the more lengthy

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \epsilon_i, \quad i = 1, 2, \ldots, n. \tag{6.6}$$

Example 6.1 (Girth, Height, and Volume for Black Cherry trees)**.** Measurements were made of the girth, height, and volume of timber in 31 felled black cherry trees. Note that girth is the diameter of the tree (in inches) measured at 4 ft 6 in above the ground. The variables are

1. `Girth`: tree diameter in inches (denoted x_1)
2. `Height`: tree height in feet (x_2).
3. `Volume`: volume of the tree in cubic feet. (y)

The data are in the `datasets` package [108] and are already on the search path; they can be viewed with

```
head(trees)
```

```
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

Let us take a look at a visual display of the data. For multiple variables, instead of a simple scatterplot we use a scatterplot matrix which is made with the `splom` function in the `lattice` package [135] as shown below. The plot is shown in Figure 6.1.

```
splom(trees)
```

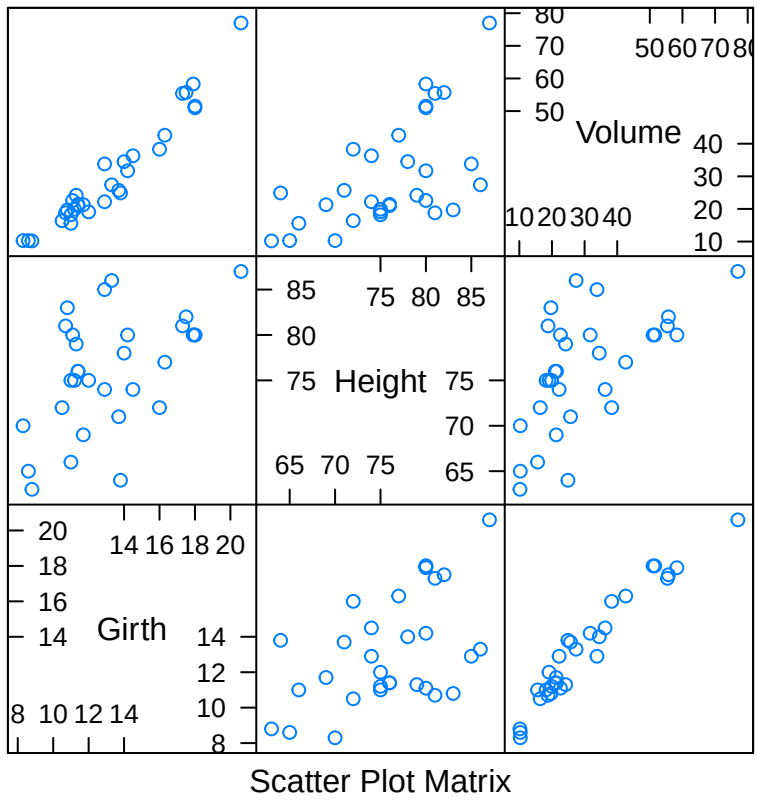

Figure 6.1: A scatterplot matrix of the `trees` data.

The dependent (response) variable `Volume` is listed in the first row of the scatterplot matrix. Moving from left to right, we see an approximately linear relationship between `Volume` and the independent (explanatory) variables `Height` and `Girth`. A first guess at a model for these data might be

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon, \tag{6.7}$$

in which case the quantity $\mu(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ would represent the mean value of Y at the point (x_1, x_2).

6.1.1 What does it mean?

The interpretation is simple. The intercept β_0 represents the mean `Volume` when all other independent variables are zero. The parameter β_i represents the change in mean `Volume` when there is a unit increase in x_i, while the other independent variable is held constant. For the `trees` data, β_1 represents the change in average `Volume` as `Girth` increases by one unit when the `Height` is held constant, and β_2 represents the change in average `Volume` as `Height` increases by one unit when the `Girth` is held constant.

In simple linear regression, we had one independent variable and our linear regression surface was 1D, simply a line. In multiple regression there are many independent variables and so our linear regression surface will be many-D... in general, a hyperplane. But when there are only two explanatory variables the hyperplane is just an ordinary plane and we can look at it with a 3D scatterplot.

One way to do this is with the R Commander in the `Rcmdr` package [45]. It has a 3D scatterplot option under the `Graphs` menu. It is especially great because the resulting graph is dynamic; it can be moved around with the mouse, zoomed, *etc*. But that particular display does not translate well to a printed book.

Another way to do it is with the `scatterplot3d` function in the `scatterplot3d` package [87]. The code follows, and the result is shown in Figure 6.2.

```
s3d <- with(trees, scatterplot3d(Girth, Height, Volume, pch = 16,
                                 highlight.3d = TRUE, angle = 60))
```

Looking at the graph we see that the data points fall close to a plane in three dimensional space. (The plot looks remarkably good. In the author's experience it is rare to see points fit so well to the plane without some additional work.)

6.2 Estimation and Prediction

6.2.1 Parameter estimates

We will proceed exactly like we did in Section 5.2. We know

$$\epsilon \sim \text{mvnorm}\left(\texttt{mean} = \mathbf{0}_{n\times 1},\ \texttt{sigma} = \sigma^2 \mathbf{I}_{n\times n}\right), \tag{6.8}$$

which means that $\mathbf{Y} = \mathbf{X}\beta + \epsilon$ has an $\text{mvnorm}\left(\texttt{mean} = \mathbf{X}\beta,\ \texttt{sigma} = \sigma^2 \mathbf{I}_{n\times n}\right)$ distribution. Therefore, the likelihood function is

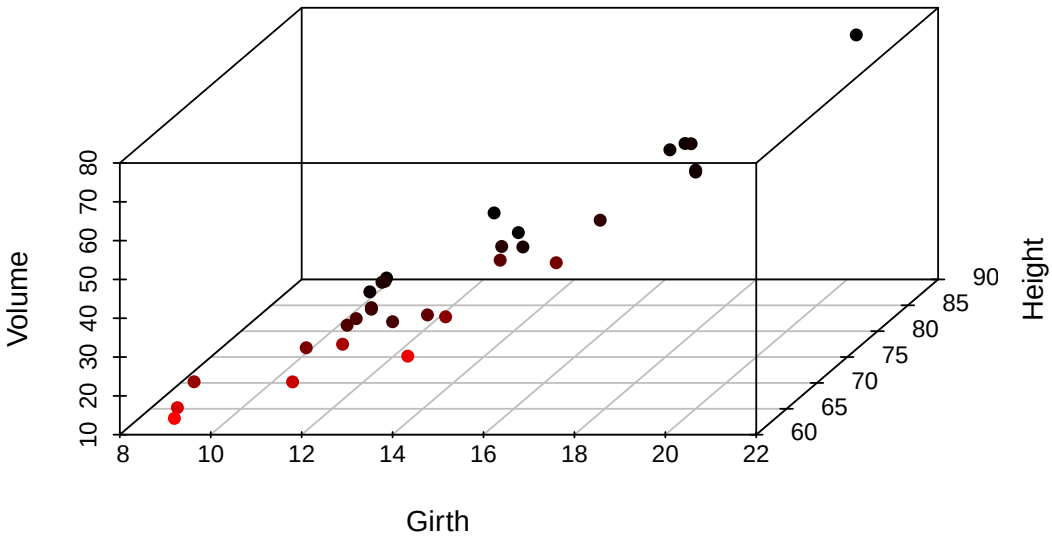

Figure 6.2: (ref:3D-scatterplot-trees)

$$L(\beta, \sigma) = \frac{1}{2\pi^{n/2}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{Y} - \mathbf{X}\beta)^{\mathrm{T}}(\mathbf{Y} - \mathbf{X}\beta)\right\}. \tag{6.9}$$

To *maximize* the likelihood in β, we need to *minimize* the quantity $g(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^{\mathrm{T}}(\mathbf{Y} - \mathbf{X}\beta)$. We do this by differentiating g with respect to β. (It may be a good idea to brush up on the material in Appendices **??** and **??**.) First we will rewrite g:

$$g(\beta) = \mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \mathbf{Y}^{\mathrm{T}}\mathbf{X}\beta - \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Y} + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta, \tag{6.10}$$

which can be further simplified to $g(\beta) = \mathbf{Y}^{\mathrm{T}}\mathbf{Y} - 2\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Y} + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta$ since $\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ is 1×1 and thus equal to its transpose. Now we differentiate to get

$$\frac{\partial g}{\partial \beta} = \mathbf{0} - 2\mathbf{X}^{\mathrm{T}}\mathbf{Y} + 2\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta, \tag{6.11}$$

since $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is symmetric. Setting the derivative equal to the zero vector yields the so called "normal equations"

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta = \mathbf{X}^{\mathrm{T}}\mathbf{Y}. \tag{6.12}$$

In the case that $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is invertible[1], we may solve the equation for β to get the maximum likelihood estimator of β which we denote by $\mathbf{b}$:

[1] We can find solutions of the normal equations even when $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is not of full rank, but the topic falls outside the scope of this book. The interested reader can consult an advanced text such as Rao [121].

$$\mathbf{b} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}. \tag{6.13}$$

Remark 6.1. The formula in Equation (6.13) is convenient for mathematical study but is inconvenient for numerical computation. Researchers have devised much more efficient algorithms for the actual calculation of the parameter estimates, and we do not explore them here.

Remark 6.2. We have only found a critical value, and have not actually shown that the critical value is a minimum. We omit the details and refer the interested reader to [121].

How to do it with R

We do all of the above just as we would in simple linear regression. The powerhouse is the `lm` function. Everything else is based on it. We separate explanatory variables in the model formula by a plus sign.

```
trees.lm <- lm(Volume ~ Girth + Height, data = trees)
trees.lm
```

```
Call:
lm(formula = Volume ~ Girth + Height, data = trees)

Coefficients:
(Intercept)        Girth       Height
   -57.9877       4.7082       0.3393
```

We see from the output that for the `trees` data our parameter estimates are

$$\mathbf{b} = \begin{bmatrix} -58.0 & 4.7 & 0.3 \end{bmatrix},$$

and consequently our estimate of the mean response is $\hat{\mu}$ given by

$$\begin{aligned} \hat{\mu}(x_1, x_2) &= b_0 + b_1 x_1 + b_2 x_2, && (6.14)\\ &\approx -58.0 + 4.7 x_1 + 0.3 x_2. && (6.15) \end{aligned}$$

We could see the entire model matrix $\mathbf{X}$ with the `model.matrix` function, but in the interest of brevity we only show the first few rows.

```
head(model.matrix(trees.lm))
```

```
  (Intercept) Girth Height
1           1   8.3     70
2           1   8.6     65
3           1   8.8     63
4           1  10.5     72
5           1  10.7     81
6           1  10.8     83
```

6.2.2 Point Estimates of the Regression Surface

The parameter estimates **b** make it easy to find the fitted values , $\hat{\mathbf{Y}}$. We write them individually as $\hat{Y}_i$, $i = 1, 2, \ldots, n$, and recall that they are defined by

$$\begin{aligned} \hat{Y}_i &= \hat{\mu}(x_{1i}, x_{2i}), && (6.16) \\ &= b_0 + b_1 x_{1i} + b_2 x_{2i}, \quad i = 1, 2, \ldots, n. && (6.17) \end{aligned}$$

They are expressed more compactly by the matrix equation

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{b}. \tag{6.18}$$

From Equation (6.13) we know that $\mathbf{b} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$, so we can rewrite

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{X}\left[\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}\right], && (6.19) \\ &= \mathbf{H}\mathbf{Y}, && (6.20) \end{aligned}$$

where $\mathbf{H} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}$ is appropriately named *the hat matrix* because it "puts the hat on **Y**". The hat matrix is very important in later courses. Some facts about **H** are

- **H** is a symmetric square matrix, of dimension n × n.
- The diagonal entries h_{ii} satisfy $0 \leq h_{ii} \leq 1$ (compare to Equation (5.57)).
- The trace is $\mathrm{tr}(\mathbf{H}) = p$.
- **H** is *idempotent* (also known as a *projection matrix*) which means that $\mathbf{H}^2 = \mathbf{H}$. The same is true of $\mathbf{I} - \mathbf{H}$.

Now let us write a column vector $\mathbf{x}_0 = (x_{10}, x_{20})^{\mathrm{T}}$ to denote given values of the explanatory variables `Girth` $== x_{10}$ and `Height` $== x_{20}$. These values may match those of the collected data, or they may be completely new values not observed in the original data set. We may use the parameter estimates to find $\hat{Y}(\mathbf{x}_0)$, which will give us 1. an estimate of $\mu(\mathbf{x}_0)$, the mean value of a future observation at $\mathbf{x}_0$, and 2. a prediction for $Y(\mathbf{x}_0)$, the actual value of a future observation at $\mathbf{x}_0$.

We can represent $\hat{Y}(\mathbf{x}_0)$ by the matrix equation

$$\hat{Y}(\mathbf{x}_0) = \mathbf{x}_0^{\mathrm{T}}\mathbf{b}, \tag{6.21}$$

which is just a fancy way to write

$$\hat{Y}(x_{10}, x_{20}) = b_0 + b_1 x_{10} + b_2 x_{20}. \tag{6.22}$$

Example 6.2. If we wanted to predict the average volume of black cherry trees that have `Girth` = 15 in and are `Height` = 77 ft tall then we would use the estimate

$$\begin{aligned}\hat{\mu}(15, 77) &= -58 + 4.7(15) + 0.3(77),\\ &\approx 35.6 \text{ ft}^3.\end{aligned}$$

We would use the same estimate $\hat{Y} = 35.6$ to predict the measured `Volume` of another black cherry tree – yet to be observed – that has `Girth` = 15 in and is `Height` = 77 ft tall.

How to do it with R

The fitted values are stored inside `trees.lm` and may be accessed with the `fitted` function. We only show the first five fitted values.

```
fitted(trees.lm)[1:5]
```

```
        1         2         3         4         5 
 4.837660  4.553852  4.816981 15.874115 19.869008 
```

The syntax for general prediction does not change much from simple linear regression. The computations are done with the `predict` function as described below.

The only difference from SLR is in the way we tell R the values of the explanatory variables for which we want predictions. In SLR we had only one independent variable but in MLR we have many (for the `trees` data we have two). We will store values for the independent variables in the data frame `new`, which has two columns (one for each independent variable) and three rows (we shall make predictions at three different locations).

```
new <- data.frame(Girth = c(9.1, 11.6, 12.5), Height = c(69, 74, 87))
```

We can view the locations at which we will predict:

```
new
```

```
  Girth Height
1   9.1     69
2  11.6     74
3  12.5     87
```

We continue just like we would have done in SLR.

```
predict(trees.lm, newdata = new)
```

```
        1         2         3 
 8.264937 21.731594 30.379205 
```

Example. Using the `trees` data,

1. Report a point estimate of the mean `Volume` of a tree of `Girth` 9.1 in and `Height` 69 ft. The fitted value for $x_1 = 9.1$ and $x_2 = 69$ is 8.3, so a point estimate would be 8.3 cubic feet.

2. Report a point prediction for and a 95% prediction interval for the `Volume` of a hypothetical tree of `Girth` 12.5 in and `Height` 87 ft. The fitted value for $x_1 = 12.5$ and $x_2 = 87$ is 30.4, so a point prediction for the `Volume` is 30.4 cubic feet.

6.2.3 Mean Square Error and Standard Error

The residuals are given by

$$\mathbf{E} = \mathbf{Y} - \hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{H}\mathbf{Y} = (\mathbf{I} - \mathbf{H})\mathbf{Y}. \tag{6.23}$$

Now we can use Theorem **??** to see that the residuals are distributed

$$\mathbf{E} \sim \text{mvnorm}(\texttt{mean} = \mathbf{0},\ \texttt{sigma} = \sigma^2(\mathbf{I} - \mathbf{H})), \tag{6.24}$$

since $(\mathbf{I} - \mathbf{H})\mathbf{X}\beta = \mathbf{X}\beta - \mathbf{X}\beta = \mathbf{0}$ and $(\mathbf{I} - \mathbf{H})\,(\sigma^2\mathbf{I})\,(\mathbf{I} - \mathbf{H})^{\mathrm{T}} = \sigma^2(\mathbf{I} - \mathbf{H})^2 = \sigma^2(\mathbf{I} - \mathbf{H})$. Thesum of squared errors SSE is just

$$SSE = \mathbf{E}^{\mathrm{T}}\mathbf{E} = \mathbf{Y}^{\mathrm{T}}(\mathbf{I} - \mathbf{H})(\mathbf{I} - \mathbf{H})\mathbf{Y} = \mathbf{Y}^{\mathrm{T}}(\mathbf{I} - \mathbf{H})\mathbf{Y}. \tag{6.25}$$

Recall that in SLR we had two parameters (β_0 and β_1) in our regression model and we estimated σ^2 with $s^2 = SSE/(n-2)$. In MLR, we have $p+1$ parameters in our regression model and we might guess that to estimate σ^2 we would use the *mean square error* S^2 defined by

$$S^2 = \frac{SSE}{n - (p+1)}. \tag{6.26}$$

That would be a good guess. The *residual standard error* is $S = \sqrt{S^2}$.

How to do it with R

The residuals are also stored with `trees.lm` and may be accessed with the `residuals` function. We only show the first five residuals.

```
residuals(trees.lm)[1:5]
```

```
        1         2         3         4         5
5.4623403 5.7461484 5.3830187 0.5258848 -1.0690084
```

The `summary` function output (shown later) lists the `Residual Standard Error` which is just $S = \sqrt{S^2}$. It is stored in the `sigma` component of the `summary` object.

```
treesumry <- summary(trees.lm)
treesumry$sigma
```

```
[1] 3.881832
```

For the `trees` data we find $s \approx 3.882$.

6.2.4 Interval Estimates of the Parameters

We showed in Section 6.2.1 that $\mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$, which is really just a big matrix – namely $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ – multiplied by $\mathbf{Y}$. It stands to reason that the sampling distribution of $\mathbf{b}$ would be intimately related to the distribution of $\mathbf{Y}$, which we assumed to be

$$\mathbf{Y} \sim \mathsf{mvnorm}\left(\mathtt{mean} = \mathbf{X}\beta,\ \mathtt{sigma} = \sigma^2\mathbf{I}\right). \tag{6.27}$$

Now recall Theorem **??** that we said we were going to need eventually (the time is now). That proposition guarantees that

$$\mathbf{b} \sim \mathsf{mvnorm}\left(\mathtt{mean} = \beta,\ \mathtt{sigma} = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right), \tag{6.28}$$

since

$$\mathbb{E}\mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T(\mathbf{X}\beta) = \beta, \tag{6.29}$$

and

$$\mathrm{Var}(\mathbf{b}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T(\sigma^2\mathbf{I})\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}, \tag{6.30}$$

the first equality following because the matrix $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ is symmetric.

There is a lot that we can glean from Equation (6.28). First, it follows that the estimator $\mathbf{b}$ is unbiased (see Section **??**). Second, the variances of $b_0, b_1, \ldots, b_n$ are exactly the diagonal elements of $\sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$, which is completely known except for that pesky parameter σ^2. Third, we can estimate the standard error of b_i (denoted S_{b_i}) with the mean square error S (defined in the previous section) multiplied by the corresponding diagonal element of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$. Finally, given estimates of the standard errors we may construct confidence intervals for β_i with an interval that looks like

$$b_i \pm \mathsf{t}_{\alpha/2}(\mathtt{df} = n - p - 1)S_{b_i}. \tag{6.31}$$

The degrees of freedom for the Student's t distribution[2] are the same as the denominator of S^2.

[2]We are taking great leaps over the mathematical details. In particular, we have yet to show that s^2 has a chi-square distribution and we have not even come close to showing that b_i and s_{b_i} are independent. But these are entirely outside the scope of the present book and the reader may rest assured that the proofs await in later classes. See C.R. Rao for more.

How to do it with R

To get confidence intervals for the parameters we need only use `confint` :

```
confint(trees.lm)
```

```
                  2.5 %      97.5 %
(Intercept) -75.68226247 -40.2930554
Girth         4.16683899   5.2494820
Height        0.07264863   0.6058538
```

For example, using the calculations above we say that for the regression model `Volume ~ Girth + Height` we are 95% confident that the parameter β_1 lies somewhere in the interval 4.2 to 5.2.

6.2.5 Confidence and Prediction Intervals

We saw in Section 6.2.2 how to make point estimates of the mean value of additional observations and predict values of future observations, but how good are our estimates? We need confidence and prediction intervals to gauge their accuracy, and lucky for us the formulas look similar to the ones we saw in SLR.

In Equation (6.21) we wrote $\hat{Y}(\mathbf{x}_0) = \mathbf{x}_0^{\mathrm{T}}\mathbf{b}$, and in Equation (6.28) we saw that

$$\mathbf{b} \sim \mathsf{mvnorm}\left(\mathtt{mean} = \beta,\ \mathtt{sigma} = \sigma^2\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\right). \tag{6.32}$$

The following is therefore immediate from Theorem **??**:

$$\hat{Y}(\mathbf{x}_0) \sim \mathsf{mvnorm}\left(\mathtt{mean} = \mathbf{x}_0^{\mathrm{T}}\beta,\ \mathtt{sigma} = \sigma^2\mathbf{x}_0^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{x}_0\right). \tag{6.33}$$

It should be no surprise that confidence intervals for the mean value of a future observation at the location $\mathbf{x}_0 = \begin{bmatrix} x_{10} & x_{20} & \dots & x_{p0} \end{bmatrix}^{\mathrm{T}}$ are given by

$$\hat{Y}(\mathbf{x}_0) \pm \mathsf{t}_{\alpha/2}(\mathtt{df} = n - p - 1)\, S\, \sqrt{\mathbf{x}_0^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{x}_0}. \tag{6.34}$$

Intuitively, $\mathbf{x}_0^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{x}_0$ measures the distance of $\mathbf{x}_0$ from the center of the data. The degrees of freedom in the Student's t critical value are $n - (p + 1)$ because we need to estimate $p + 1$ parameters.

Prediction intervals for a new observation at $\mathbf{x}_0$ are given by

$$\hat{Y}(\mathbf{x}_0) \pm \mathsf{t}_{\alpha/2}(\mathtt{df} = n - p - 1)\, S\, \sqrt{1 + \mathbf{x}_0^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{x}_0}. \tag{6.35}$$

The prediction intervals are wider than the confidence intervals, just as in Section 5.2.5.

How to do it with R

The syntax is identical to that used in SLR, with the proviso that we need to specify values of the independent variables in the data frame `new` as we did in Section 5.2.5 (which we repeat here for illustration).

```
new <- data.frame(Girth = c(9.1, 11.6, 12.5), Height = c(69, 74, 87))
```

Confidence intervals are given by

```
predict(trees.lm, newdata = new, interval = "confidence")
```

```
        fit      lwr      upr
1  8.264937  5.77240 10.75747
2 21.731594 20.11110 23.35208
3 30.379205 26.90964 33.84877
```

Prediction intervals are given by

```
predict(trees.lm, newdata = new, interval = "prediction")
```

```
        fit         lwr      upr
1  8.264937 -0.06814444 16.59802
2 21.731594 13.61657775 29.84661
3 30.379205 21.70364103 39.05477
```

As before, the interval type is decided by the `interval` argument and the default confidence level is 95% (which can be changed with the `level` argument).

Example. Using the `trees` data,

1. Report a 95% confidence interval for the mean `Volume` of a tree of `Girth` 9.1 in and `Height` 69 ft. The 95% CI is given by 5.8 to 10.8, so with 95% confidence the mean `Volume` lies somewhere between 5.8 cubic feet and 10.8 cubic feet.
2. Report a 95% prediction interval for the `Volume` of a hypothetical tree of `Girth` 12.5 in and `Height` 87 ft. The 95% prediction interval is given by 26.9 to 33.8, so with 95% confidence we may assert that the hypothetical `Volume` of a tree of `Girth` 12.5 in and `Height` 87 ft would lie somewhere between 26.9 cubic feet and 33.8 feet.

6.3 Model Utility and Inference

6.3.1 Multiple Coefficient of Determination

We saw in Section 6.2.3 that the error sum of squares SSE can be conveniently written in MLR as

$$SSE = \mathbf{Y}^{\mathrm{T}}(\mathbf{I} - \mathbf{H})\mathbf{Y}. \tag{6.36}$$

It turns out that there are equally convenient formulas for the total sum of squares $SSTO$ and the regression sum of squares SSR. They are:

$$SSTO = \mathbf{Y}^{\mathrm{T}}\left(\mathbf{I} - \frac{1}{n}\mathbf{J}\right)\mathbf{Y} \tag{6.37}$$

and

$$SSR = \mathbf{Y}^{\mathrm{T}}\left(\mathbf{H} - \frac{1}{n}\mathbf{J}\right)\mathbf{Y}. \tag{6.38}$$

(The matrix $\mathbf{J}$ is defined in Appendix **??**.) Immediately from Equations (6.36), (6.37), and (6.38) we get the *Anova Equality*

$$SSTO = SSE + SSR. \tag{6.39}$$

(See Exercise **??**.) We define the *multiple coefficient of determination* by the formula

$$R^2 = 1 - \frac{SSE}{SSTO}. \tag{6.40}$$

We interpret R^2 as the proportion of total variation that is explained by the multiple regression model. In MLR we must be careful, however, because the value of R^2 can be artificially inflated by the addition of explanatory variables to the model, regardless of whether or not the added variables are useful with respect to prediction of the response variable. In fact, it can be proved that the addition of a single explanatory variable to a regression model will increase the value of R^2, *no matter how worthless* the explanatory variable is. We could model the height of the ocean tides, then add a variable for the length of cheetah tongues on the Serengeti plain, and our R^2 would inevitably increase.

This is a problem, because as the philosopher, Occam, once said: "causes should not be multiplied beyond necessity". We address the problem by penalizing R^2 when parameters are added to the model. The result is an *adjusted* R^2 which we denote by $\overline{R}^2$.

$$\overline{R}^2 = \left(R^2 - \frac{p}{n-1}\right)\left(\frac{n-1}{n-p-1}\right). \tag{6.41}$$

It is good practice for the statistician to weigh both R^2 and $\overline{R}^2$ during assessment of model utility. In many cases their values will be very close to each other. If their values differ substantially, or if one changes dramatically when an explanatory variable is added, then (s)he should take a closer look at the explanatory variables in the model.

How to do it with R

For the `trees` data, we can get R^2 and $\overline{R}^2$ from the `summary` output or access the values directly by name as shown (recall that we stored the `summary` object in `treesumry`).

```
treesumry$r.squared
```

```
[1] 0.94795
```

```
treesumry$adj.r.squared
```

```
[1] 0.9442322
```

High values of R^2 and $\overline{R}^2$ such as these indicate that the model fits very well, which agrees with what we saw in Figure 6.2.

6.3.2 Overall F-Test

Another way to assess the model's utility is to to test the hypothesis

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0 \text{ versus } H_1 : \text{ at least one } \beta_i \neq 0.$$

The idea is that if all β_i's were zero, then the explanatory variables $X_1, \ldots, X_p$ would be worthless predictors for the response variable Y. We can test the above hypothesis with the overall F statistic, which in MLR is defined by

$$F = \frac{SSR/p}{SSE/(n-p-1)}. \tag{6.42}$$

When the regression assumptions hold and under H_0, it can be shown that $F \sim \mathsf{f}(\texttt{df1} = p, \texttt{df2} = n - p - 1)$. We reject H_0 when F is large, that is, when the explained variation is large relative to the unexplained variation.

How to do it with R

The overall F statistic and its associated p-value is listed at the bottom of the `summary` output, or we can access it directly by name; it is stored in the `fstatistic` component of the `summary` object.

```
treesumry$fstatistic
```

```
   value    numdf    dendf
254.9723   2.0000  28.0000
```

For the `trees` data, we see that (F =) 254.9723374 with a p-value =< `2.2e-16`. Consequently we reject H_0, that is, the data provide strong evidence that not all β_i's are zero.

6.3.3 Student's t Tests

We know that

$$\mathbf{b} \sim \text{mvnorm}\left(\texttt{mean} = \beta,\ \texttt{sigma} = \sigma^2 \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\right) \tag{6.43}$$

and we have seen how to test the hypothesis $H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$, but let us now consider the test

$$H_0 : \beta_i = 0 \text{ versus } H_1 : \beta_i \neq 0, \tag{6.44}$$

where β_i is the coefficient for the i^{th} independent variable. We test the hypothesis by calculating a statistic, examining it's null distribution, and rejecting H_0 if the p-value is small. If H_0 is rejected, then we conclude that there is a significant relationship between Y and x_i *in the regression model* $Y \sim (x_1, \ldots, x_p)$. This last part of the sentence is very important because the significance of the variable x_i sometimes depends on the presence of other independent variables in the model[3].

To test the hypothesis we go to find the sampling distribution of (b_{i}), the estimator of the corresponding parameter β_i, when the null hypothesis is true. We saw in Section 6.2.4 that

$$T_i = \frac{b_i - \beta_i}{S_{b_i}} \tag{6.45}$$

has a Student's t distribution with $n - (p + 1)$ degrees of freedom. (Remember, we are estimating $p + 1$ parameters.) Consequently, under the null hypothesis $H_0 : \beta_i = 0$ the statistic $t_i = b_i / S_{b_i}$ has a t($\texttt{df} = n - p - 1$) distribution.

How to do it with R

The Student's t tests for significance of the individual explanatory variables are shown in the `summary` output.

```
treesumry
```

```
Call:
lm(formula = Volume ~ Girth + Height, data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
```

[3] In other words, a variable might be highly significant one moment but then fail to be significant when another variable is added to the model. When this happens it often indicates a problem with the explanatory variables, such as *multicollinearity*. See Section **??**.

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
Girth         4.7082     0.2643  17.816  < 2e-16 ***
Height        0.3393     0.1302   2.607   0.0145 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

We see from the *p-values* that there is a significant linear relationship between `Volume` and `Girth` and between `Volume` and `Height` in the regression model `Volume ~ Girth + Height`. Further, it appears that the `Intercept` is significant in the aforementioned model.

6.4 Polynomial Regression

6.4.1 Quadratic Regression Model

In each of the previous sections we assumed that μ was a linear function of the explanatory variables. For example, in SLR we assumed that $\mu(x) = \beta_0 + \beta_1 x$, and in our previous MLR examples we assumed $\mu(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. In every case the scatterplots indicated that our assumption was reasonable. Sometimes, however, plots of the data suggest that the linear model is incomplete and should be modified.

```
plot(Volume ~ Girth, data = trees)
```

For example, let us examine a scatterplot of `Volume` versus `Girth` a little more closely. See Figure 6.3. There might be a slight curvature to the data; the volume curves ever so slightly upward as the girth increases. After looking at the plot we might try to capture the curvature with a mean response such as

$$\mu(x_1) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2. \tag{6.46}$$

The model associated with this choice of μ is

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \epsilon. \tag{6.47}$$

The regression assumptions are the same. Almost everything indeed is the same. In fact, it is still called a "linear regression model", since the mean response μ is linear *in the parameters* β_0, β_1, and β_2.

However, there is one important difference. When we introduce the squared variable in the model we inadvertently also introduce strong dependence between the terms which can cause significant

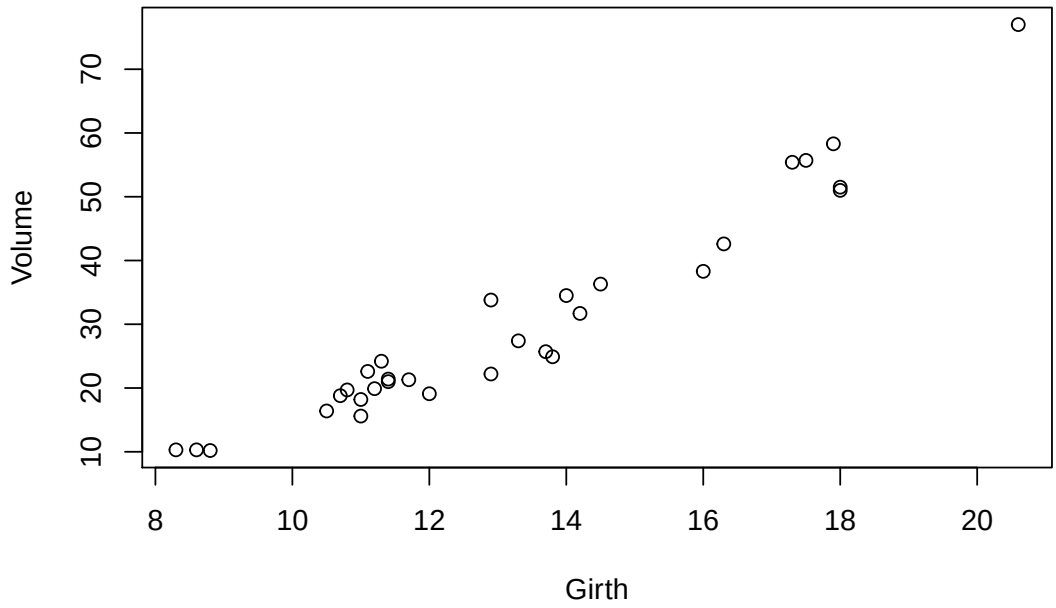

Figure 6.3: Scatterplot of `Volume` versus `Girth` for the `trees` data.

numerical problems when it comes time to calculate the parameter estimates. Therefore, we should usually rescale the independent variable to have mean zero (and even variance one if we wish) *before* fitting the model. That is, we replace the x_i's with $x_i - \overline{x}$ (or $(x_i - \overline{x})/s$) before fitting the model[4].

How to do it with R

There are multiple ways to fit a quadratic model to the variables `Volume` and `Girth` using R. 1. One way would be to square the values for `Girth` and save them in a vector `Girthsq`. Next, fit the linear model `Volume ~ Girth + Girthsq`. 2. A second way would be to use the *insulate* function in R, denoted by `I`: : Volume ~ Girth + I(Girthˆ2) The second method is shorter than the first but the end result is the same. And once we calculate and store the fitted model (in, say, `treesquad.lm`) all of the previous comments regarding R apply. 3. A third and "right" way to do it is with orthogonal polynomials: : Volume ~ poly(Girth, degree = 2) See `?poly` and `?cars` for more information. Note that we can recover the approach in 2 with `poly(Girth, degree = 2, raw = TRUE)`.

Example 6.3. We will fit the quadratic model to the `trees` data and display the results with `summary`, being careful to rescale the data before fitting the model. We may rescale the `Girth` variable to have zero mean and unit variance on-the-fly with the `scale` function.

[4]Rescaling the data gets the job done but a better way to avoid the multicollinearity introduced by the higher order terms is with *orthogonal polynomials*, whose coefficients are chosen just right so that the polynomials are not correlated with each other. This is beginning to linger outside the scope of this book, however, so we will content ourselves with a brief mention and then stick with the rescaling approach in the discussion that follows. A nice example of orthogonal polynomials in action can be run with `example(cars)`.

```
treesquad.lm <- lm(Volume ~ scale(Girth) + I(scale(Girth)^2), data = trees)
summary(treesquad.lm)
```

```
Call:
lm(formula = Volume ~ scale(Girth) + I(scale(Girth)^2), data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4889 -2.4293 -0.3718  2.0764  7.6447

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        27.7452     0.8161  33.996  < 2e-16 ***
scale(Girth)       14.5995     0.6773  21.557  < 2e-16 ***
I(scale(Girth)^2)   2.5067     0.5729   4.376 0.000152 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.335 on 28 degrees of freedom
Multiple R-squared:  0.9616,    Adjusted R-squared:  0.9588
F-statistic: 350.5 on 2 and 28 DF,  p-value: < 2.2e-16
```

We see that the F statistic indicates the overall model including `Girth` and `Girth^2` is significant. Further, there is strong evidence that both `Girth` and `Girth^2` are significantly related to `Volume`. We may examine a scatterplot together with the fitted quadratic function using the `lines` function, which adds a line to the plot tracing the estimated mean response.

```
plot(Volume ~ scale(Girth), data = trees)
lines(fitted(treesquad.lm) ~ scale(Girth), data = trees)
```

The plot is shown in Figure 6.4. Pay attention to the scale on the x-axis: it is on the scale of the transformed `Girth` data and not on the original scale.

Remark 6.3. When a model includes a quadratic term for an independent variable, it is customary to also include the linear term in the model. The principle is called *parsimony*. More generally, if the researcher decides to include x^m as a term in the model, then (s)he should also include all lower order terms $x, x^2, \ldots, x^{m-1}$ in the model.

We do estimation/prediction the same way that we did in Section 6.2.2, except we do not need a `Height` column in the dataframe =new= since the variable is not included in the quadratic model.

```
new <- data.frame(Girth = c(9.1, 11.6, 12.5))
predict(treesquad.lm, newdata = new, interval = "prediction")
```

```
       fit       lwr      upr
1 11.56982  4.347426 18.79221
2 20.30615 13.299050 27.31325
3 25.92290 18.972934 32.87286
```

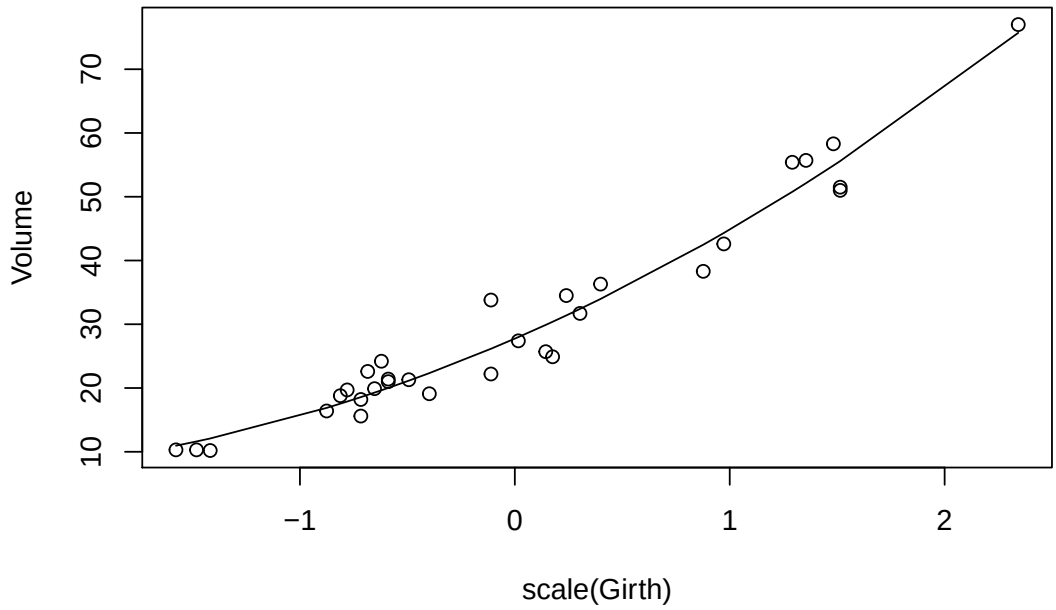

Figure 6.4: A quadratic model for the `trees` data.

The predictions and intervals are slightly different from what they were previously. Notice that it was not necessary to rescale the `Girth` prediction data before input to the `predict` function; the model did the rescaling for us automatically.

Remark 6.4. We have mentioned on several occasions that it is important to rescale the explanatory variables for polynomial regression. Watch what happens if we ignore this advice:

```
summary(lm(Volume ~ Girth + I(Girth^2), data = trees))
```

```
Call:
lm(formula = Volume ~ Girth + I(Girth^2), data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4889 -2.4293 -0.3718  2.0764  7.6447

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.78627   11.22282   0.961 0.344728
Girth       -2.09214    1.64734  -1.270 0.214534
I(Girth^2)   0.25454    0.05817   4.376 0.000152 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.335 on 28 degrees of freedom
Multiple R-squared:  0.9616,    Adjusted R-squared:  0.9588
F-statistic: 350.5 on 2 and 28 DF,  p-value: < 2.2e-16
```

Now nothing is significant in the model except `Girth^2`. We could delete the `Intercept` and `Girth` from the model, but the model would no longer be *parsimonious*. A novice may see the output and be confused about how to proceed, while the seasoned statistician recognizes immediately that `Girth` and `Girth^2` are highly correlated (see Section 6.7.3). The only remedy to this ailment is to rescale `Girth`, which we should have done in the first place.

In Example **??** of Section 6.5.

Note: The `trees` data do not have any qualitative explanatory variables, so we will construct one for illustrative purposes[5]. We will leave the `Girth` variable alone, but we will replace the variable `Height` by a new variable `Tall` which indicates whether or not the cherry tree is taller than a certain threshold (which for the sake of argument will be the sample median height of 76 ft). That is, `Tall` will be defined by

$$\texttt{Tall} = \begin{cases} \texttt{yes}, & \text{if } \texttt{Height} > 76, \\ \texttt{no}, & \text{if } \texttt{Height} \leq 76. \end{cases} \tag{6.48}$$

We can construct `Tall` very quickly in R with the `cut` function:

```
trees$Tall <- cut(trees$Height, breaks = c(-Inf, 76, Inf),
                  labels = c("no","yes"))
trees$Tall[1:5]
```

```
[1] no  no  no  no  yes
Levels: no yes
```

Note that `Tall` is automatically generated to be a factor with the labels in the correct order. See `?cut` for more.

Once we have `Tall`, we include it in the regression model just like we would any other variable. It is handled internally in a special way. Define a "dummy variable" `Tallyes` that takes values

$$\texttt{Tallyes} = \begin{cases} 1, & \text{if } \texttt{Tall} = \texttt{yes}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.49}$$

That is, `Tallyes` is an *indicator variable* which indicates when a respective tree is tall. The model may now be written as

$$\texttt{Volume} = \beta_0 + \beta_1 \texttt{Girth} + \beta_2 \texttt{Tallyes} + \epsilon. \tag{6.50}$$

[5]This procedure of replacing a continuous variable by a discrete/qualitative one is called *binning*, and is almost *never* the right thing to do. We are in a bind at this point, however, because we have invested this chapter in the `trees` data and I do not want to switch mid-discussion. I am currently searching for a data set with pre-existing qualitative variables that also conveys the same points present in the trees data, and when I find it I will update this chapter accordingly.

Let us take a look at what this definition does to the mean response. Trees with `Tall = yes` will have the mean response

$$\mu(\texttt{Girth}) = (\beta_0 + \beta_2) + \beta_1 \texttt{Girth}, \tag{6.51}$$

while trees with `Tall = no` will have the mean response

$$\mu(\texttt{Girth}) = \beta_0 + \beta_1 \texttt{Girth}. \tag{6.52}$$

In essence, we are fitting two regression lines: one for tall trees, and one for short trees. The regression lines have the same slope but they have different y intercepts (which are exactly $|\beta_2|$ far apart).

6.4.2 How to do it with R

The important thing is to double check that the qualitative variable in question is stored as a factor. The way to check is with the `class` command. For example,

```
class(trees$Tall)
```

```
[1] "factor"
```

If the qualitative variable is not yet stored as a factor then we may convert it to one with the `factor` command. See Section 3.1.3. Other than this we perform MLR as we normally would.

```
treesdummy.lm <- lm(Volume ~ Girth + Tall, data = trees)
summary(treesdummy.lm)
```

```
Call:
lm(formula = Volume ~ Girth + Tall, data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-5.7788 -3.1710  0.4888  2.6737 10.0619

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -34.1652     3.2438  -10.53 3.02e-11 ***
Girth         4.6988     0.2652   17.72  < 2e-16 ***
Tallyes       4.3072     1.6380    2.63   0.0137 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.875 on 28 degrees of freedom
Multiple R-squared:  0.9481,    Adjusted R-squared:  0.9444
F-statistic: 255.9 on 2 and 28 DF,  p-value: < 2.2e-16
```

From the output we see that all parameter estimates are statistically significant and we conclude that the mean response differs for trees with `Tall = yes` and trees with `Tall = no`.

Remark 6.5. We were somewhat disingenuous when we defined the dummy variable `Tallyes` because, in truth, R defines `Tallyes` automatically without input from the user[6]. Indeed, the author fit the model beforehand and wrote the discussion afterward with the knowledge of what R would do so that the output the reader saw would match what (s)he had previously read. The way that R handles factors internally is part of a much larger topic concerning *contrasts*, which falls outside the scope of this book. The interested reader should see Neter et al [99] or Fox [43] for more.

Remark 6.6. In general, if an explanatory variable `foo` is qualitative with n levels `bar1`, `bar2`, ..., `barn` then R will by default automatically define $n - 1$ indicator variables in the following way:

$$\texttt{foobar2} = \begin{cases} 1, & \text{if } \texttt{foo} = \text{"}\texttt{bar2}\text{"}, \\ 0, & \text{otherwise.} \end{cases}, \ldots, \texttt{foobarn} = \begin{cases} 1, & \text{if } \texttt{foo} = \text{"}\texttt{barn}\text{"}, \\ 0, & \text{otherwise.} \end{cases}$$

The level `bar1` is represented by $\texttt{foobar2} = \cdots = \texttt{foobarn} = 0$. We just need to make sure that `foo` is stored as a factor and R will take care of the rest.

6.4.3 Graphing the Regression Lines

We can see a plot of the two regression lines with the following mouthful of code.

```
treesTall <- split(trees, trees$Tall)
treesTall[["yes"]]$Fit <- predict(treesdummy.lm, treesTall[["yes"]])
treesTall[["no"]]$Fit <- predict(treesdummy.lm, treesTall[["no"]])
plot(Volume ~ Girth, data = trees)
points(Volume ~ Girth, data = treesTall[["yes"]], pch = 1)
points(Volume ~ Girth, data = treesTall[["no"]], pch = 2)
lines(Fit ~ Girth, data = treesTall[["yes"]])
lines(Fit ~ Girth, data = treesTall[["no"]])
```

It may look intimidating but there is reason to the madness. First we `split` the `trees` data into two pieces, with groups determined by the `Tall` variable. Next we add the fitted values to each piece via `predict`. Then we set up a `plot` for the variables `Volume` versus `Girth`, but we do not plot anything yet (`type = n`) because we want to use different symbols for the two groups. Next we add `points` to the plot for the `Tall = yes` trees and use an open circle for a plot character (`pch = 1`), followed by `points` for the `Tall = no` trees with a triangle character (`pch = 2`). Finally, we add regression `lines` to the plot, one for each group.

There are other – shorter – ways to plot regression lines by groups, namely the `scatterplot` function in the `car` package [48] and the `xyplot` function in the `lattice` package [135]. We

[6] That is, R by default handles contrasts according to its internal settings which may be customized by the user for fine control. Given that we will not investigate contrasts further in this book it does not serve the discussion to delve into those settings, either. The interested reader should check `?contrasts` for details.

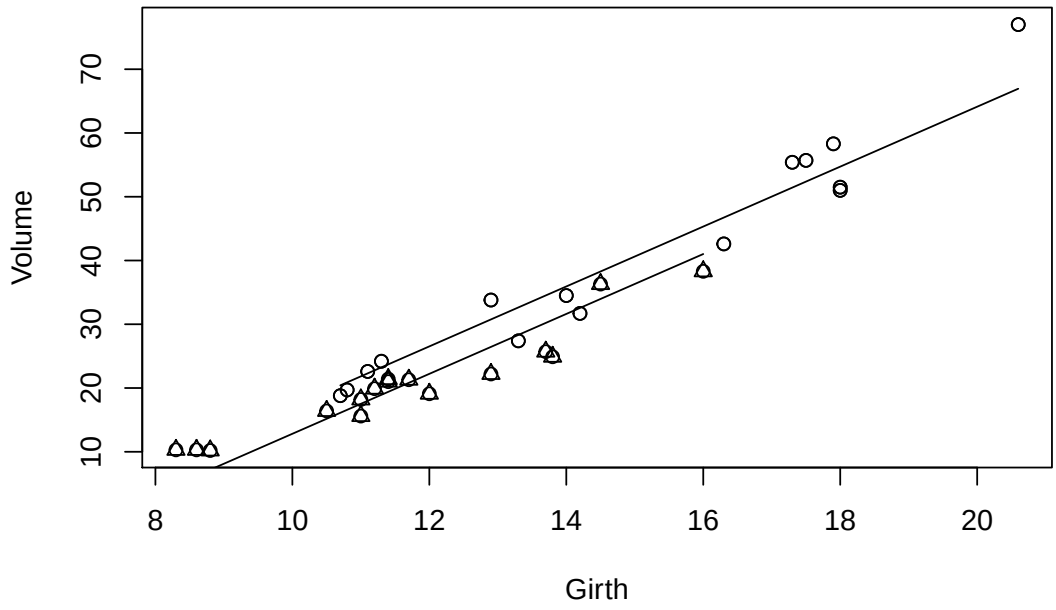

Figure 6.5: A dummy variable model for the `trees` data.

elected to introduce the reader to the above approach since many advanced plots in R are done in a similar, consecutive fashion.

6.5 Partial F Statistic

We saw in Section 6.3.2 how to test $H_0 : \beta_0 = \beta_1 = \cdots = \beta_p = 0$ with the overall F statistic and we saw in Section 6.3.3 how to test $H_0 : \beta_i = 0$ that a particular coefficient β_i is zero. Sometimes, however, we would like to test whether a certain part of the model is significant. Consider the regression model

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_j x_j + \beta_{j+1} x_{j+1} + \cdots + \beta_p x_p + \epsilon, \tag{6.53}$$

where $j \geq 1$ and $p \geq 2$. Now we wish to test the hypothesis

$$H_0 : \beta_{j+1} = \beta_{j+2} = \cdots = \beta_p = 0 \tag{6.54}$$

versus the alternative

$$H_1 : \text{at least one of } \beta_{j+1},\ \beta_{j+2},\ , \ldots, \beta_p \neq 0. \tag{6.55}$$

The interpretation of H_0 is that none of the variables $x_{j+1}, \ldots, x_p$ is significantly related to Y and the interpretation of H_1 is that at least one of $x_{j+1}, \ldots, x_p$ is significantly related to Y. In essence, for this hypothesis test there are two competing models under consideration:

$$\text{the full model:} \quad y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon, \tag{6.56}$$

$$\text{the reduced model:} \quad y = \beta_0 + \beta_1 x_1 + \cdots + \beta_j x_j + \epsilon, \tag{6.57}$$

Of course, the full model will always explain the data *better* than the reduced model, but does the full model explain the data *significantly better* than the reduced model? This question is exactly what the partial F statistic is designed to answer.

We first calculate SSE_f, the unexplained variation in the full model, and SSE_r, the unexplained variation in the reduced model. We base our test on the difference $SSE_r - SSE_f$ which measures the reduction in unexplained variation attributable to the variables $x_{j+1}, \ldots, x_p$. In the full model there are $p+1$ parameters and in the reduced model there are $j+1$ parameters, which gives a difference of $p-j$ parameters (hence degrees of freedom). The partial F statistic is

$$F = \frac{(SSE_r - SSE_f)/(p-j)}{SSE_f/(n-p-1)}. \tag{6.58}$$

It can be shown when the regression assumptions hold under H_0 that the partial F statistic has an f(df1 $= p-j$, df2 $= n-p-1$) distribution. We calculate the p-value of the observed partial F statistic and reject H_0 if the p-value is small.

6.5.1 How to do it with R

The key ingredient above is that the two competing models are *nested* in the sense that the reduced model is entirely contained within the complete model. The way to test whether the improvement is significant is to compute `lm` objects both for the complete model and the reduced model then compare the answers with the `anova` function.

Example 6.4 (The Trees data)**.** For the `trees` data, let us fit a polynomial regression model and for the sake of argument we will ignore our own good advice and fail to rescale the explanatory variables.

```
treesfull.lm <- lm(Volume ~ Girth + I(Girth^2) + Height +
                   I(Height^2), data = trees)
summary(treesfull.lm)
```

```
Call:
lm(formula = Volume ~ Girth + I(Girth^2) + Height + I(Height^2),
    data = trees)

Residuals:
   Min     1Q Median     3Q    Max
```

```
-4.368 -1.670 -0.158  1.792  4.358

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.955101  63.013630  -0.015    0.988
Girth       -2.796569   1.468677  -1.904    0.068 .
I(Girth^2)   0.265446   0.051689   5.135 2.35e-05 ***
Height       0.119372   1.784588   0.067    0.947
I(Height^2)  0.001717   0.011905   0.144    0.886
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.674 on 26 degrees of freedom
Multiple R-squared:  0.9771,    Adjusted R-squared:  0.9735
F-statistic:   277 on 4 and 26 DF,  p-value: < 2.2e-16
```

In this ill-formed model nothing is significant except `Girth` and `Girth^2`. Let us continue down this path and suppose that we would like to try a reduced model which contains nothing but `Girth` and `Girth^2` (not even an `Intercept`). Our two models are now

$$\begin{aligned} \text{the full model:} \quad & Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \epsilon, \\ \text{the reduced model:} \quad & Y = \beta_1 x_1 + \beta_2 x_1^2 + \epsilon, \end{aligned}$$

We fit the reduced model with `lm` and store the results:

```
treesreduced.lm <- lm(Volume ~ -1 + Girth + I(Girth^2), data = trees)
```

To delete the intercept from the model we used `-1` in the model formula. Next we compare the two models with the `anova` function. The convention is to list the models from smallest to largest.

```
anova(treesreduced.lm, treesfull.lm)
```

```
Analysis of Variance Table

Model 1: Volume ~ -1 + Girth + I(Girth^2)
Model 2: Volume ~ Girth + I(Girth^2) + Height + I(Height^2)
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     29 321.65
2     26 185.86  3    135.79 6.3319 0.002279 **
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see from the output that the complete model is highly significant compared to the model that does not incorporate `Height` or the `Intercept`. We wonder (with our tongue in our cheek) if the

Height^2 term in the full model is causing all of the trouble. We will fit an alternative reduced model that only deletes Height^2.

```
treesreduced2.lm <- lm(Volume ~ Girth + I(Girth^2) + Height,
                       data = trees)
anova(treesreduced2.lm, treesfull.lm)
```

```
Analysis of Variance Table

Model 1: Volume ~ Girth + I(Girth^2) + Height
Model 2: Volume ~ Girth + I(Girth^2) + Height + I(Height^2)
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     27 186.01
2     26 185.86  1   0.14865 0.0208 0.8865
```

In this case, the improvement to the reduced model that is attributable to Height^2 is not significant, so we can delete Height^2 from the model with a clear conscience. We notice that the p-value for this latest partial F test is 0.8865, which seems to be remarkably close to the p-value we saw for the univariate t test of Height^2 at the beginning of this example. In fact, the *p-values* are *exactly* the same. Perhaps now we gain some insight into the true meaning of the univariate tests.

6.6 Residual Analysis and Diagnostic Tools

We encountered many, many diagnostic measures for simple linear regression in Sections 5.4 and 5.5. All of these are valid in multiple linear regression, too, but there are some slight changes that we need to make for the multivariate case. We list these below, and apply them to the trees example.

- **Shapiro-Wilk, Breusch-Pagan, Durbin-Watson:** unchanged from SLR, but we are now equipped to talk about the Shapiro-Wilk test statistic for the residuals. It is defined by the formula

$$W = \frac{\mathbf{a}^T\mathbf{E}^*}{\mathbf{E}^T\mathbf{E}}, \tag{6.59}$$

  where $\mathbf{E}^*$ is the sorted residuals and $\mathbf{a}_{1\times n}$ is defined by

$$\mathbf{a} = \frac{\mathbf{m}^T\mathbf{V}^{-1}}{\sqrt{\mathbf{m}^T\mathbf{V}^{-1}\mathbf{V}^{-1}\mathbf{m}}}, \tag{6.60}$$

  where $\mathbf{m}_{n\times 1}$ and $\mathbf{V}_{n\times n}$ are the mean and covariance matrix, respectively, of the order statistics from an mvnorm(mean = $\mathbf{0}$, sigma = $\mathbf{I}$) distribution.
- **Leverages:** are defined to be the diagonal entries of the hat matrix $\mathbf{H}$ (which is why we called them h_{ii} in Section 6.2.2). The sum of the leverages is $\text{tr}(\mathbf{H}) = p + 1$. One rule of thumb considers a leverage extreme if it is larger than double the mean leverage value, which is $2(p + 1)/n$, and another rule of thumb considers leverages bigger than 0.5 to indicate high leverage, while values between 0.3 and 0.5 indicate moderate leverage.
- **Standardized residuals:** unchanged. Considered extreme if $|R_i| > 2$.
- **Studentized residuals:** compared to a t(df = $n - p - 2$) distribution.

- **DFBETAS:** The formula is generalized to

$$(DFBETAS)_{j(i)} = \frac{b_j - b_{j(i)}}{S_{(i)}\sqrt{c_{jj}}}, \quad j = 0, \ldots p, \; i = 1, \ldots, n, \tag{6.61}$$

  where c_{jj} is the j^{th} diagonal entry of $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$. Values larger than one for small data sets or $2/\sqrt{n}$ for large data sets should be investigated.
- **DFFITS:** unchanged. Larger than one in absolute value is considered extreme.
- **Cook's D:** compared to an f(`df1` $= p + 1$, `df2` $= n - p - 1$) distribution. Observations falling higher than the 50^{th} percentile are extreme. Note that plugging the value $p = 1$ into the formulas will recover all of the ones we saw in Chapter 5.

6.7 Additional Topics

6.7.1 Nonlinear Regression

We spent the entire chapter talking about the `trees` data, and all of our models looked like `Volume ~ Girth + Height` or a variant of this model. But let us think again: we know from elementary school that the volume of a rectangle is $V = lwh$ and the volume of a cylinder (which is closer to what a black cherry tree looks like) is

$$V = \pi r^2 h \quad \text{or} \quad V = 4\pi dh, \tag{6.62}$$

where r and d represent the radius and diameter of the tree, respectively. With this in mind, it would seem that a more appropriate model for μ might be

$$\mu(x_1, x_2) = \beta_0 x_1^{\beta_1} x_2^{\beta_2}, \tag{6.63}$$

where β_1 and β_2 are parameters to adjust for the fact that a black cherry tree is not a perfect cylinder.

How can we fit this model? The model is not linear in the parameters any more, so our linear regression methods will not work... or will they? In the `trees` example we may take the logarithm of both sides of Equation (6.63) to get

$$\mu^*(x_1, x_2) = \ln\left[\mu(x_1, x_2)\right] = \ln\beta_0 + \beta_1 \ln x_1 + \beta_2 \ln x_2, \tag{6.64}$$

and this new model μ^* is linear in the parameters $\beta_0^* = \ln\beta_0$, $\beta_1^* = \beta_1$ and $\beta_2^* = \beta_2$. We can use what we have learned to fit a linear model `log(Volume) ~ log(Girth) + log(Height)`, and everything will proceed as before, with one exception: we will need to be mindful when it comes time to make predictions because the model will have been fit on the log scale, and we will need to transform our predictions back to the original scale (by exponentiating with `exp`) to make sense.

```
treesNonlin.lm <- lm(log(Volume) ~ log(Girth) + log(Height), data = trees)
summary(treesNonlin.lm)
```

```
Call:
lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)

Residuals:
      Min        1Q    Median        3Q       Max
-0.168561 -0.048488  0.002431  0.063637  0.129223

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
log(Girth)   1.98265    0.07501  26.432  < 2e-16 ***
log(Height)  1.11712    0.20444   5.464 7.81e-06 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.08139 on 28 degrees of freedom
Multiple R-squared:  0.9777,    Adjusted R-squared:  0.9761
F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

This is our best model yet (judging by R^2 and $\overline{R}^2$), all of the parameters are significant, it is simpler than the quadratic or interaction models, and it even makes theoretical sense. It rarely gets any better than that.

We may get confidence intervals for the parameters, but remember that it is usually better to transform back to the original scale for interpretation purposes:

```
exp(confint(treesNonlin.lm))
```

```
                   2.5 %      97.5 %
(Intercept) 0.0002561078 0.006783093
log(Girth)  6.2276411645 8.468066317
log(Height) 2.0104387829 4.645475188
```

(Note that we did not update the row labels of the matrix to show that we exponentiated and so they are misleading as written.) We do predictions just as before. Remember to transform the response variable back to the original scale after prediction.

```
new <- data.frame(Girth = c(9.1, 11.6, 12.5), Height = c(69, 74, 87))
exp(predict(treesNonlin.lm, newdata = new, interval = "confidence"))
```

```
       fit      lwr      upr
1 11.90117 11.25908 12.57989
2 20.82261 20.14652 21.52139
3 28.93317 27.03755 30.96169
```

The predictions and intervals are slightly different from those calculated earlier, but they are close. Note that we did not need to transform the `Girth` and `Height` arguments in the dataframe `new`. All transformations are done for us automatically.

6.7.2 Real Nonlinear Regression

We saw with the `trees` data that a nonlinear model might be more appropriate for the data based on theoretical considerations, and we were lucky because the functional form of μ allowed us to take logarithms to transform the nonlinear model to a linear one. The same trick will not work in other circumstances, however. We need techniques to fit general models of the form

$$\mathbf{Y} = \mu(\mathbf{X}) + \epsilon, \tag{6.65}$$

where μ is some crazy function that does not lend itself to linear transformations.

There are a host of methods to address problems like these which are studied in advanced regression classes. The interested reader should see Neter *et al* [99] or Tabachnick and Fidell [149].

It turns out that John Fox has posted an Appendix to his book [42] which discusses some of the methods and issues associated with nonlinear regression; see http://cran.r-project.org/doc/contrib/Fox-Companion/appendix.html for more. Here is an example of how it works, based on a question from R-help.

```
# fake data
set.seed(1)
x <- seq(from = 0, to = 1000, length.out = 200)
y <- 1 + 2*(sin((2*pi*x/360) - 3))^2 + rnorm(200, sd = 2)
# plot(x, y)
acc.nls <- nls(y ~ a + b*(sin((2*pi*x/360) - c))^2,
               start = list(a = 0.9, b = 2.3, c = 2.9))
summary(acc.nls)
```

```
Formula: y ~ a + b * (sin((2 * pi * x/360) - c))^2

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  0.95884    0.23097   4.151 4.92e-05 ***
b  2.22868    0.37114   6.005 9.07e-09 ***
c  3.04343    0.08434  36.084  < 2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.865 on 197 degrees of freedom

Number of iterations to convergence: 3
Achieved convergence tolerance: 6.407e-08
```

```
#plot(x, fitted(acc.nls))
```

6.7.3 Multicollinearity

A multiple regression model exhibits *multicollinearity* when two or more of the explanatory variables are substantially correlated with each other. We can measure multicollinearity by having one of the explanatory play the role of "dependent variable" and regress it on the remaining explanatory variables. The the R^2 of the resulting model is near one, then we say that the model is multicollinear or shows multicollinearity.

Multicollinearity is a problem because it causes instability in the regression model. The instability is a consequence of redundancy in the explanatory variables: a high R^2 indicates a strong dependence between the selected independent variable and the others. The redundant information inflates the variance of the parameter estimates which can cause them to be statistically insignificant when they would have been significant otherwise. To wit, multicollinearity is usually measured by what are called *variance inflation factors*.

Once multicollinearity has been diagnosed there are several approaches to remediate it. Here are a couple of important ones.

- **Principal Components Analysis.** This approach casts out two or more of the original explanatory variables and replaces them with new variables, derived from the original ones, that are by design uncorrelated with one another. The redundancy is thus eliminated and we may proceed as usual with the new variables in hand. Principal Components Analysis is important for other reasons, too, not just for fixing multicollinearity problems.
- **Ridge Regression.** The idea of this approach is to replace the original parameter estimates with a different type of parameter estimate which is more stable under multicollinearity. The estimators are not found by ordinary least squares but rather a different optimization procedure which incorporates the variance inflation factor information.

We decided to omit a thorough discussion of multicollinearity because we are not equipped to handle the mathematical details. Perhaps the topic will receive more attention in a later edition.

- What to do when data are not normal
- Bootstrap (see Chapter **??**).

6.7.4 Akaike's Information Criterion

$$AIC = -2 \ln L + 2(p + 1)$$

6.8 Exercises

Exercise. Use Equations (6.36), (6.37), and (6.38) to prove the Anova Equality:

$$SSTO = SSE + SSR.$$

Bibliography

[1] Daniel Adler and Duncan Murdoch. *rgl: 3D visualization device system (OpenGL)*. R package version 0.93.952. 2013. URL: http://CRAN.R-project.org/package=rgl.

[2] A. Agresti and B. A. Coull. “Approximate is better than "exact" for interval estimation of binomial proportions”. In: *The American Statistician* 52 (1998), pp. 119–126.

[3] Alan Agresti. *Categorical Data Analysis*. Wiley, 2002.

[4] Martin Maechler et al. *sfsmisc: Utilities from Seminar fuer Statistik ETH Zurich*. R package version 1.0-24. 2013. URL: http://CRAN.R-project.org/package=sfsmisc.

[5] Fortran code by Alan Genz and R code by Adelchi Azzalini. *mnormt: The multivariate normal and t distributions*. R package version 1.4-5. 2012. URL: http://CRAN.R-project.org/package=mnormt.

[6] Jim Albert. *LearnBayes: Functions for Learning Bayesian Inference*. R package version 2.12. 2012. URL: http://CRAN.R-project.org/package=LearnBayes.

[7] Dirk Eddelbuettel with contributions by Antoine Lucas et al. *digest: Create cryptographic hash digests of R objects*. R package version 0.6.3. 2013. URL: http://CRAN.R-project.org/package=digest.

[8] Tom M. Apostol. *Calculus*. Second. Vol. I. Wiley, 1967.

[9] Tom M. Apostol. *Calculus*. Second. Vol. II. Wiley, 1967.

[10] Robert B. Ash and Catherine Doleans-Dade. *Probability & Measure Theory*. Harcourt Academic Press, 2000.

[11] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.0-12. 2013. URL: http://CRAN.R-project.org/package=Matrix.

[12] S original by Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weingessel@ci.tuwien.ac.at>. *quadprog: Functions to solve Quadratic Programming Problems.* R package version 1.5-5. 2013. URL: http://CRAN.R-project.org/package=quadprog.

[13] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics*. Vol. I. Prentice Hall, 2001.

[14] Patrick Billingsley. *Probability and Measure*. Wiley Interscience, 1995.

[15] Ben Bolker. *emdbook: Ecological Models and Data in R*. R package version 1.3.4. 2013.

[16] Bruce L. Bowerman, Richard O’Connell, and Anne Koehler. *Forecasting, Time Series, and Regression: An Applied Approach*. South-Western College Pub, 2004.

[17] P. J. Brockwell and R. A. Davis. *Time Series and Forecasting Methods*. Second. Springer, 1991, p. 555.

[18] Angelo Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-9. 2013.

[19] Neal L. Carothers. *Real Analysis*. Cambridge University Press, 2000.

[20] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, 2002.

[21] Kung-Sik Chan and Brian Ripley. *TSA: Time Series Analysis*. R package version 1.01. 2012. URL: http://CRAN.R-project.org/package=TSA.

[22] Scott Chasalow. *combinat: combinatorics utilities*. R package version 0.0-8. 2012. URL: http://CRAN.R-project.org/package=combinat.

[23] S original by Chris Fraley et al. *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. 2012. URL: http://CRAN.R-project.org/package=fracdiff.

[24] Erhan Cinlar. *Introduction to Stochastic Processes*. Prentice Hall, 1975.

[25] William S. Cleveland. *The Elements of Graphing Data*. Hobart Press, 1994.

[26] Yves Croissant and Giovanni Millo. "Panel Data Econometrics in R: The plm Package". In: *Journal of Statistical Software* 27.2 (2008). URL: http://www.jstatsoft.org/v27/i02/.

[27] David B. Dahl. *xtable: Export tables to LaTeX or HTML*. R package version 1.7-1. 2013. URL: http://CRAN.R-project.org/package=xtable.

[28] Peter Dalgaard. *Introductory Statistics with R*. Springer, 2008. URL: http://staff.pubhealth.ku.dk/~pd/ISwR.html.

[29] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. ISBN 0-521-57391-2. Cambridge: Cambridge University Press, 1997. URL: http://statwww.epfl.ch/davison/BMA/.

[30] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. Cambridge University Press, 1997.

[31] Thomas J. DiCiccio and Bradley Efron. "Bootstrap Confidence Intervals". In: *Statistical Science* 11 (1996), pp. 189–228.

[32] M Dowle, T Short, and S Lianoglou. *data.table: Extension of data.frame for fast indexing, fast ordered joins, fast assignment, fast grouping and list columns.* R package version 1.8.8. 2013. URL: http://CRAN.R-project.org/package=data.table.

[33] Richard Durrett. *Probability: Theory and Examples*. Duxbury Press, 1996.

[34] Rick Durrett. *Essentials of Stochastic Processes*. Springer, 1999.

[35] Christophe Dutang, Vincent Goulet, and Mathieu Pigeon. "actuar: An R Package for Actuarial Science". In: *Journal of Statistical Software* 25.7 (2008), p. 38. URL: http://www.jstatsoft.org/v25/i07.

[36] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. ISBN 978-1-4614-6867-7. New York: Springer, 2013.

[37] Dirk Eddelbuettel and Romain François. "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8 (2011), pp. 1–18. URL: http://www.jstatsoft.org/v40/i08/.

[38] Dirk Eddelbuettel and Conrad Sanderson. "RcppArmadillo: Accelerating R with high-performance C++ linear algebra". In: *Computational Statistics and Data Analysis* in press (2013). URL: http://dx.doi.org/10.1016/j.csda.2013.02.005.

[39] Brian Everitt. *An R and S-Plus Companion to Multivariate Analysis*. Springer, 2007.

[40] Gerald B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley, 1999.

[41] Thomas Lumley using Fortran code by Alan Miller. *leaps: regression subset selection*. R package version 2.9. 2009. URL: http://CRAN.R-project.org/package=leaps.

[42] John Fox. *An R and S Plus Companion to Applied Regression*. Sage, 2002.

[43] John Fox. *Applied Regression Analysis, Linear Models, and Related Methods*. Sage, 1997.

[44] John Fox. "Effect Displays in R for Generalised Linear Models". In: *Journal of Statistical Software* 8.15 (2003), pp. 1–27. URL: http://www.jstatsoft.org/v08/i15/.

[45] John Fox. "The R Commander: A Basic Statistics Graphical User Interface to R". In: *Journal of Statistical Software* 14.9 (2005), pp. 1–42. URL: http://www.jstatsoft.org/v14/i09.

[46] John Fox and Jangman Hong. "Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the effects Package". In: *Journal of Statistical Software* 32.1 (2009), pp. 1–24. URL: http://www.jstatsoft.org/v32/i01/.

[47] John Fox, Zhenghua Nie, and Jarrett Byrnes. *sem: Structural Equation Models*. R package version 3.1-3. 2013. URL: http://CRAN.R-project.org/package=sem.

[48] John Fox and Sanford Weisberg. *An R Companion to Applied Regression*. Second. Thousand Oaks CA: Sage, 2011. URL: http://socserv.socsci.mcmaster.ca/jfox/Books/Companion.

[49] Michael Friendly. *Visualizing Categorical Data*. SAS Publishing, 2000.

[50] Andrew Gelman et al. *Bayesian Data Analysis*. CRC Press, 2004.

[51] Alan Genz and Frank Bretz. *Computation of Multivariate Normal and t Probabilities*. Lecture Notes in Statistics. Heidelberg: Springer-Verlag, 2009. ISBN: 978-3-642-01688-2.

[52] Alan Genz et al. *mvtnorm: Multivariate Normal and t Distributions*. R package version 0.9-9995. 2013. URL: http://CRAN.R-project.org/package=mvtnorm.

[53] Rob J Hyndman with contributions from George Athanasopoulos et al. *forecast: Forecasting functions for time series and linear models*. R package version 4.06. 2013. URL: http://CRAN.R-project.org/package=forecast.

[54] Gregor Gorjanc. "Using Sweave with LyX". In: *R News* 1 (2008), pp. 2–9.

[55] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 1997. URL: http://www.dartmouth.edu/~chance/.

[56] Bettina Grün and Achim Zeileis. "Automatic Generation of Exams in R". In: *Journal of Statistical Software* 29.10 (2009), pp. 1–14. URL: http://www.jstatsoft.org/v29/i10/.

[57] David Firth with contributions from Heather Turner. *relimp: Relative Contribution of Effects in a Regression Model*. R package version 1.0-3. 2011. URL: http://CRAN.R-project.org/package=relimp.

[58] Richard M. Heiberger. *HH: Statistical Analysis and Data Display: Heiberger and Holland*. R package version 2.3-37. 2013. URL: http://CRAN.R-project.org/package=HH.

[59] Richard M. Heiberger and Burt Holland. *Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS*. Springer, 2004. URL: http://astro.temple.edu/~rmh/HH/.

[60] Richard M. Heiberger and Erich Neuwirth. *R Through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics*. Springer, 2009. URL: http://www.springer.com/statistics/computanional+statistics/book/978-1-4419-0051-7.

[61] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to Mathematical Statistics*. Pearson Prentice Hall, 2005.

[62] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Pearson Prentice Hall, 2006.

[63] Torsten Hothorn, Frank Bretz, and Peter Westfall. "Simultaneous Inference in General Parametric Models". In: *Biometrical Journal* 50.3 (2008), pp. 346–363.

[64] Torsten Hothorn and Kurt Hornik. *exactRankTests: Exact Distributions for Rank and Permutation Tests*. R package version 0.8-25. 2013. URL: http://CRAN.R-project.org/package=exactRankTests.

[65] Torsten Hothorn, Friedrich Leisch, and Achim Zeileis. *modeltools: Tools and Classes for Statistical Models*. R package version 0.2-19. 2012. URL: http://CRAN.R-project.org/package=modeltools.

[66] Torsten Hothorn et al. "A Lego System for Conditional Inference". In: *The American Statistician* 60.3 (2006), pp. 257–263.

[67] Torsten Hothorn et al. "Implementing a Class of Permutation Tests: The coin Package". In: *Journal of Statistical Software* 28.8 (2008), pp. 1–23. URL: http://www.jstatsoft.org/v28/i08/.

[68] Ross Ihaka et al. *colorspace: Color Space Manipulation*. R package version 1.2-2. 2013. URL: http://CRAN.R-project.org/package=colorspace.

[69] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Second. Vol. 1. Wiley, 1994.

[70] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Second. Vol. 2. Wiley, 1995.

[71] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Discrete Multivariate Distributions*. Wiley, 1997.

[72] Norman L. Johnson, Samuel Kotz, and Adrienne W. Kemp. *Univariate Discrete Distributions*. Second. Wiley, 1993.

[73] Roger W. Johnson. *How Many Fish are in the Pond?* URL: http://www.rsscse.org.uk/ts/gtb/johnson3.pdf.

[74] Frank E Harrell Jr, with contributions from Charles Dupont, and many others. *Hmisc: Harrell Miscellaneous*. R package version 3.12-2. 2013. URL: http://CRAN.R-project.org/package=Hmisc.

[75] Louis Kates and Thomas Petzoldt. *proto: Prototype object-based programming*. R package version 0.3-10. 2012. URL: http://CRAN.R-project.org/package=proto.

[76] G. Jay Kerns. *IPSUR: Introduction to Probability and Statistics Using R*. R package version 1.4. 2012. URL: http://CRAN.R-project.org/package=IPSUR.

[77] G. Jay Kerns. *prob: Elementary Probability on Finite Sample Spaces*. R package version 0.9-2. 2009. URL: http://CRAN.R-project.org/package=prob.

[78] Samuel Kotz, N. Balakrishnan, and Norman L. Johnson. *Continuous Multivariate Distributions*. Second. Vol. 1: Models and Applications. Wiley, 2000.

[79] Duncan Temple Lang. *XML: Tools for parsing and generating XML within R and S-Plus.* R package version 3.98-1.1. 2013. URL: http://CRAN.R-project.org/package=XML.

[80] Michael Lavine. *Introduction to Statistical Thought*. Lavine, Michael, 2009. URL: http://www.math.umass.edu/~lavine/Book/book.html.

[81] Peter M. Lee. *Bayesian Statistics: An Introduction*. Wiley, 1997.

[82] E. L. Lehmann. *Testing Statistical Hypotheses*. Springer-Verlag, 1986.

[83] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer, 1998.

[84] Jim Lemon and Philippe Grosjean. *prettyR: Pretty descriptive stats.* R package version 2.0-7. 2013. URL: http://CRAN.R-project.org/package=prettyR.

[85] Andy Liaw and Matthew Wiener. "Classification and Regression by randomForest". In: *R News* 2.3 (2002), pp. 18–22. URL: http://CRAN.R-project.org/doc/Rnews/.

[86] Uwe Ligges. "Accessing the Sources". In: *R News* 6 (2006), pp. 43–45.

[87] Uwe Ligges and Martin Mächler. "Scatterplot3d - an R Package for Visualizing Multivariate Data". In: *Journal of Statistical Software* 8.11 (2003), pp. 1–20. URL: http://www.jstatsoft.org.

[88] Catherine Loader. *locfit: Local Regression, Likelihood and Density Estimation.* R package version 1.5-9.1. 2013. URL: http://CRAN.R-project.org/package=locfit.

[89] Thomas Lumley. *dichromat: Color Schemes for Dichromats*. R package version 2.0-0. 2013. URL: http://CRAN.R-project.org/package=dichromat.

[90] Martin Maechler et al. *cluster: Cluster Analysis Basics and Extensions*. R package version 1.14.4 — For new features, see the 'Changelog' file (in the package source). 2013.

[91] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 1999.

[92] John Maindonald and John Braun. *Data Analysis and Graphics Using R*. Cambridge University Press, 2003.

[93] John Maindonald and W. John Braun. *DAAG: Data Analysis And Graphics data and functions*. R package version 1.16. 2013. URL: http://CRAN.R-project.org/package=DAAG.

[94] David Meyer, Achim Zeileis, and Kurt Hornik. "The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd". In: *Journal of Statistical Software* 17.3 (2006), pp. 1–48. URL: http://www.jstatsoft.org/v17/i03/.

[95] David Meyer, Achim Zeileis, and Kurt Hornik. *vcd: Visualizing Categorical Data*. R package version 1.2-13. 2012.

[96] David Meyer et al. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.6-1. 2012. URL: http://CRAN.R-project.org/package=e1071.

[97] Ben Mezrich. *Bringing Down the House: The Inside Story of Six M.I.T. Students Who Took Vegas for Millions*. Free Press, 2003.

[98] Jeff Miller. *Earliest Known Uses of Some of the Words of Mathematics*. URL: http://jeff560.tripod.com/mathword.html.

[99] John Neter et al. *Applied Linear Regression Models*. Third. McGraw Hill, 1996.

[100] Erich Neuwirth. *RColorBrewer: ColorBrewer palettes*. R package version 1.0-5. 2011. URL: http://CRAN.R-project.org/package=RColorBrewer.

[101] Frederick Novomestky. *matrixcalc: Collection of functions for matrix calculations*. R package version 1.0-3. 2012. URL: http://CRAN.R-project.org/package=matrixcalc.

[102] Jari Oksanen et al. *vegan: Community Ecology Package*. R package version 2.0-8. 2013. URL: http://CRAN.R-project.org/package=vegan.

[103] Jose Pinheiro et al. *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-110. 2013.

[104] Tony Plate and Richard Heiberger. *abind: Combine multi-dimensional arrays*. R package version 1.4-0. 2011. URL: http://CRAN.R-project.org/package=abind.

[105] R Core Team. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...* R package version 0.8-54. 2013. URL: http://CRAN.R-project.org/package=foreign.

[106] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[107] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[108] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[109] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[110] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[111] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[112] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[113] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[114] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[115] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[116] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[117] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[118] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[119] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[120] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2009. URL: http://www.R-project.org.

[121] C. Radhakrishna Rao and Helge Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer, 1999.

[122] Sidney I. Resnick. *A Probability Path*. Birkhauser, 1999.

[123] Brian Ripley and from 1999 to Oct 2002 Michael Lapsley. *RODBC: ODBC Database Access*. R package version 1.3-7. 2013. URL: http://CRAN.R-project.org/package=RODBC.

[124] Maria L. Rizzo. *Statistical Computing with R*. Chapman & Hall/CRC, 2008.

[125] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[126] Kenneth A. Ross. *Elementary Calculus: The Theory of Calculus*. Springer, 1980.

[127] RStudio. *manipulate: Interactive Plots for RStudio*. R package version 0.97.551. 2011.

[128] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA, 2012. URL: http://www.rstudio.com/.

[129] Peter Ruckdeschel. *startupmsg: Utilities for start-up messages*. R package version 0.8. 2013. URL: http://CRAN.R-project.org/package=startupmsg.

[130] Peter Ruckdeschel. *SweaveListingUtils: Utilities for Sweave together with TeX listings package*. R package version 0.6.1. 2013. URL: http://CRAN.R-project.org/package=SweaveListingUtils.

[131] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[132] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[133] P. Ruckdeschel et al. "S4 Classes for Distributions". English. In: *R News* 6.2 (May 2006), pp. 2–6. URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/distr.pdf.

[134] P. Ruckdeschel et al. *S4 Classes for Distributions—a manual for packages distr, distrSim, distrTEst, distrEx, distrMod, and distrTeach*. English. Tech. rep. Kaiserslautern, Germany: Fraunhofer ITWM, July 2008. URL: http://r-forge.r-project.org/plugins/scmsvn/viewcvs.php/*checkout*/pkg/distrDoc/inst/doc/distr.pdf?root=distr.

[135] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R*. ISBN 978-0-387-75968-5. New York: Springer, 2008. URL: http://lmdvr.r-forge.r-project.org.

[136] Deepayan Sarkar and Felix Andrews. *latticeExtra: Extra Graphical Utilities Based on Lattice*. R package version 0.6-24. 2012. URL: http://CRAN.R-project.org/package=latticeExtra.

[137] F. E. Satterthwaite. "An Approximate Distribution of Estimates of Variance Components". In: *Biometrics Bulletin* 2 (1946), pp. 110–114.

[138] Luca Scrucca. “qcc: an R package for quality control charting and statistical process control”. In: *R News* 4/1 (2004), pp. 11–17. URL: http://CRAN.R-project.org/doc/Rnews/.

[139] Robert J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, 1980.

[140] S. S. Shapiro and M. B. Wilk. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52 (1965), pp. 591–611.

[141] Gavin L. Simpson. *permute: Functions for generating restricted permutations of data*. R package version 0.7-0. 2012. URL: http://CRAN.R-project.org/package=permute.

[142] Greg Snow. *TeachingDemos: Demonstrations for teaching and learning*. R package version 2.9. 2013. URL: http://CRAN.R-project.org/package=TeachingDemos.

[143] Karline Soetaert. *diagram: Functions for visualising simple graphs (networks), plotting flow diagrams*. R package version 1.6.1. 2013. URL: http://CRAN.R-project.org/package=diagram.

[144] Karline Soetaert. *shape: Functions for plotting graphical shapes, colors*. R package version 1.4.0. 2012. URL: http://CRAN.R-project.org/package=shape.

[145] Max Kuhn. Contributions from Steve Weston et al. *odfWeave: Sweave processing of Open Document Format (ODF) files*. R package version 0.8.2. 2012. URL: http://CRAN.R-project.org/package=odfWeave.

[146] James Stewart. *Calculus*. Thomson Brooks/Cole, 2008.

[147] Stephen M. Stigler. *The History of Statistics: The Measurement of Uncertainty before 1900*. Harvard University Press, 1986.

[148] Gilbert Strang. *Linear Algebra and Its Applications*. Harcourt, 1988.

[149] Barbara G. Tabachnick and Linda S. Fidell. *Using Multivariate Statistics*. Allyn and Bacon, 2006.

[150] Justin Talbot. *labeling: Axis Labeling*. R package version 0.2. 2013. URL: http://CRAN.R-project.org/package=labeling.

[151] Terry M. Therneau and Patricia M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. New York: Springer, 2000. ISBN: 0-387-98784-3.

[152] G. Jay Kerns with contributions by Theophilius Boye, Tyler Drombosky, and adapted from the work of John Fox et al. *RcmdrPlugin.IPSUR: An IPSUR Plugin for the R Commander*. R package version 0.1-8. 2012. URL: http://CRAN.R-project.org/package=RcmdrPlugin.IPSUR.

[153] Terry Therneau. *bdsmatrix: Routines for Block Diagonal Symmetric matrices*. R package version 1.3-1. 2013. URL: http://CRAN.R-project.org/package=bdsmatrix.

[154] Terry M Therneau. *A Package for Survival Analysis in S*. R package version 2.37-4. 2013. URL: http://CRAN.R-project.org/package=survival.

[155] Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning*. R package version 4.1-1. 2013.

[156] Luke Tierney. *codetools: Code Analysis Tools for R*. R package version 0.2-8. 2011. URL: http://CRAN.R-project.org/package=codetools.

[157] Luke Tierney. *tkrplot: TK Rplot*. R package version 0.0-23. 2011. URL: http://CRAN.R-project.org/package=tkrplot.

[158] Adrian Trapletti and Kurt Hornik. *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-32. 2013. URL: http://CRAN.R-project.org/package=tseries.

[159] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[160] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[161] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[162] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[163] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.

[164] William N. Venables and David M. Smith. *An Introduction to R*. 2010. URL: http://www.r-project.org/Manuals.

[165] John Verzani. *Using R for Introductory Statistics*. CRC Press, 2005. URL: http://www.math.csi.cuny.edu/UsingR/.

[166] John Verzani. *UsingR: Data sets for the text "Using R for Introductory Statistics"*. R package version 0.1-18. 2012. URL: http://CRAN.R-project.org/package=UsingR.

[167] Matt Wand. *KernSmooth: Functions for kernel smoothing for Wand and Jones (1995)*. R package version 2.23-10. 2013. URL: http://CRAN.R-project.org/package=KernSmooth.

[168] B. L. Welch. “The generalization of "Student's" problem when several different population variances are involved”. In: *Biometrika* 34 (1947), pp. 28–35.

[169] Charlotte Wickham. *munsell: Munsell colour system*. R package version 0.4.2. 2013. URL: http://CRAN.R-project.org/package=munsell.

[170] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

[171] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book.

[172] Hadley Wickham. *gtable: Arrange grobs in tables.* R package version 0.1.2. 2012. URL: http://CRAN.R-project.org/package=gtable.

[173] Hadley Wickham. “Reshaping Data with the reshape Package”. In: *Journal of Statistical Software* 21.12 (2007), pp. 1–20. URL: http://www.jstatsoft.org/v21/i12/.

[174] Hadley Wickham. *scales: Scale functions for graphics.* R package version 0.2.3. 2012. URL: http://CRAN.R-project.org/package=scales.

[175] Hadley Wickham. *stringr: Make it easier to work with strings.* R package version 0.6.2. 2012. URL: http://CRAN.R-project.org/package=stringr.

[176] Hadley Wickham. “The Split-Apply-Combine Strategy for Data Analysis”. In: *Journal of Statistical Software* 40.1 (2011), pp. 1–29. URL: http://www.jstatsoft.org/v40/i01/.

[177] Wickham and Hadley. “Reshaping data with the reshape package”. In: *Journal of Statistical Software* 21.12 (2007). URL: http://www.jstatsoft.org/v21/i12/paper.

[178] Peter Wolf and Uni Bielefeld. *aplpack: Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, and some slider functions*. R package version 1.2.7. 2012. URL: http://CRAN.R-project.org/package=aplpack.

[179] S. N. Wood. “Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models”. In: *Journal of the Royal Statistical Society (B)* 73.1 (2011), pp. 3–36.

[180] S. N. Wood. “Modelling and smoothing parameter estimation with multiple quadratic penalties”. In: *Journal of the Royal Statistical Society (B)* 62.2 (2000), pp. 413–428.

[181] S. N. Wood. “Stable and efficient multiple smoothing parameter estimation for generalized additive models”. In: *Journal of the American Statistical Association* 99.467 (2004), pp. 673–686.

[182] S. N. Wood. “Thin-plate regression splines”. In: *Journal of the Royal Statistical Society (B)* 65.1 (2003), pp. 95–114.

[183] S.N Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006.

[184] Achim Zeileis. “Econometric Computing with HC and HAC Covariance Matrix Estimators”. In: *Journal of Statistical Software* 11.10 (2004), pp. 1–17. URL: http://www.jstatsoft.org/v11/i10/.

[185] Achim Zeileis. “Object-oriented Computation of Sandwich Estimators”. In: *Journal of Statistical Software* 16.9 (2006), pp. 1–16. URL: http://www.jstatsoft.org/v16/i09/..

[186] Achim Zeileis and Yves Croissant. “Extended Model Formulas in R: Multiple Parts and Multiple Responses”. In: *Journal of Statistical Software* 34.1 (2010), pp. 1–13. URL: http://www.jstatsoft.org/v34/i01/.

[187] Achim Zeileis and Gabor Grothendieck. “zoo: S3 Infrastructure for Regular and Irregular Time Series”. In: *Journal of Statistical Software* 14.6 (2005), pp. 1–27. URL: http://www.jstatsoft.org/v14/i06/.

[188] Achim Zeileis, Kurt Hornik, and Paul Murrell. “Escaping RGBland: Selecting Colors for Statistical Graphics”. In: *Computational Statistics & Data Analysis* 53 (2009), pp. 3259–3270. DOI: 10.1016/j.csda.2008.11.033.

[189] Achim Zeileis and Torsten Hothorn. “Diagnostic Checking in Regression Relationships”. In: *R News* 2.3 (2002), pp. 7–10. URL: http://CRAN.R-project.org/doc/Rnews/.

[190] Achim Zeileis, David Meyer, and Kurt Hornik. “Residual-based Shadings for Visualizing (Conditional) Independence”. In: *Journal of Computational and Graphical Statistics* 16.3 (2007), pp. 507–525.

Index