CHAPTER 1

Introduction

CHAPTER OBJECTIVES

- State the purpose of this book.
- Note what you will be able to do after reading this book.
- Say whom the book is for.
- Describe the book's approach.

Welcome, dear reader, to the amazing world of R!

Purpose of This Book

The R statistical analysis software (R Core Team, 2021), a remarkably powerful tool for statistical analysis, has enjoyed considerable growth in popularity in a variety of industries. RStudio (*https://rstudio.com*), an integrated development environment (IDE), is popular among R users.[1]

[1] An IDE is an application that combines activities that are common when writing computer code, easing the code writing task, and thereby improving efficiency and productivity. The RStudio IDE includes a Console; a syntax-highlighting editor that supports code completion, smart indentation, and the ability to directly execute code; integrated R help and documentation; a workspace browser and data viewer; and debugging, authoring, and other tools.

The R statistical analysis software provides powerful data analysis tools for researchers, program evaluators, and others who cannot afford (or who do not want to spend money on) the cost of other analytic software.

However, R has a steep learning curve, especially for those who have only ever used pull-down menus (rather than writing analytic syntax) to analyze data and who therefore may be concerned about the prospect of learning the software. R's remarkable functionality and flexibility contribute to this difficulty, as an analytic task often may be successfully accomplished in several different ways using the R software. As a result, users who are new to R encounter a bewildering collection of possibilities, sometimes with little or no guidance about which path to take. The purpose of this book is to overcome this difficulty by providing a clear, concise, and direct path to learning R.

What You Will Be Able to Do after Reading This Book

This book introduces the use of R to conduct fundamental data analysis tasks, including working within the RStudio IDE, reading data, data wrangling, data analysis, data visualization, and reporting in a manner that supports reproducible research. The book will be particularly helpful for those who are just getting started using R. After working through this book, you will be able to:

1. Use R to accomplish fundamental data handling, analysis, visualization, and reporting tasks, and
2. Know how to find and use additional resources to further your R capabilities.

I have written this book in a friendly manner to facilitate ease of access to the material, with an emphasis on learning how to learn to use R. The book will be of value to anyone who wants gentle guidance into the amazing world of R. Rest assured, you *can* learn and use this software, and it can even be fun.

Whom This Book Is For

I have written this book to be accessible by anyone who has a basic understanding of how to handle data analysis tasks. The book assumes no prior familiarity with R, and it is written particularly for those who are timid about learning R. The book introduces the use of R for a variety of statistical procedures, but it does not teach the procedures themselves. Thus, I assume that readers either already have an understanding of basic statistics or are simultaneously learning basic statistics elsewhere (and this book will certainly support you in the process of learning statistics).

Data analysis software is used in pretty much every field that conducts quantitative research. Thus, this book has a broad general audience. More specifically, it is intended for anyone who wants an efficient, direct, and gentle introduction to using R. The book supports those with limited resources by helping overcome the steep learning curve for this free software, thus helping to make it easily accessible to a wide variety of users.

This book can be used in an undergraduate or graduate course on analytic methods, as either a primary or supplemental text, depending on course coverage. The book can also be used to support training in workshop or other formats. Those who are learning R on their own will also find the book useful.

Approach

Using R to solve data analysis puzzles can be quite interesting, and my goal is to ease the learning curve for R by providing a user-friendly introduction to the software. My philosophy is also that learning can be fun and that by thinking about the process of becoming familiar with R, readers are empowered to further their learning beyond this book and into the vast domain of R's remarkable resources. I emphasize a direct approach to helping you learn how to use R to accomplish fundamental data analysis tasks. This differs from approaches that emphasize R itself and its many options, rather than analysis tasks immediately at hand. The text is supported by example code, screenshots, and output from analyses.

This book also emphasizes good work habits and processes that make tracking one's work easier. (For example, comments within the R code can be used to document decisions made in the process of carrying out a project and its related data analysis.) This emphasis supports research reproducibility, which is largely a matter of policy and good work habits. This is particularly important for those who wish to reproduce a previous study's results as a prelude to secondary analyses or for those who wish to replicate a study at a new time, in a new geographic location, or with a new population.

The book begins with instructions on downloading and installing R and RStudio. It then continues with an overview of RStudio and how to download and install R packages, which add functionality to the initial Base R installation. Packages include code, documentation, and other information that can be shared among users and provide tools for accomplishing tasks in R. Packages support reproducibility through sharing code that users develop to solve data handling and analysis problems (an advantage of R is that we, too, can develop and share our own packages if we are interested in doing so). From there, the book introduces the use of R for data handling, analysis, visualization, and reporting. This content describes both *Base R* and *Tidyverse* approaches to working in R. It will help you develop a sense for both because some capabilities are available using one approach but not the other, and also because you may need to work with people who adopt one or the other approach. Finally, the book briefly introduces user-written functions and offers ideas about next steps in learning R.

Keep in mind throughout this book that you *can* learn R. As is true about learning anything new, especially a new language, it will take time and practice. As you work through the book, be sure to enter and run the R code, so that you take a hands-on approach to actively engage with the material. Be gentle, remember to breathe, take a break whenever you need one (especially if you find yourself stuck or frustrated), and have some fun. Let's go!

KEY TAKEAWAYS

- R is a powerful tool for statistical analysis that is also free.
- The purpose of this book is to provide a clear, concise, and direct path to learning R and to help you learn how to learn more about using R.
- After reading this book you will be able to:
  - Use R for fundamental data handling, analysis, visualization, and reporting, and
  - Know to find and use additional resources to further your R capabilities.
- It is important to be able to work from both Base R and Tidyverse approaches.
- The book is written in a friendly manner to facilitate ease of access to the material.
- You *can* learn and use R software, and it *can* be fun.

Guilford Publications
370 Seventh Avenue
New York, NY 10001
212-431-9800
800-365-7006
www.guilford.com

CHAPTER 7

R Markdown

CHAPTER OBJECTIVES

- Become familiar with elements of an R Markdown file.
- Use R Markdown to create various types of output (for example, HTML document, Word document, and PowerPoint presentation).
- Understand how R Markdown integrates analysis and reporting.

In addition to performing data analysis tasks, R can output the results of analyses to various formats (for example, a Word document, HTML, a PowerPoint presentation) along with accompanying text. This capability eases report writing and facilitates research reproducibility. As defined by Bollen et al. (2015, pp. 3–4), reproducibility

> refers to the ability of a researcher to duplicate the results of a prior study using the same materials and procedures as were used by the original investigator. So in an attempt to reproduce a published statistical analysis, a second researcher might use the same raw data to build the same analysis files and implement the same statistical analysis to determine whether they yield the same results. . . . Reproducibility is a minimum necessary condition for a finding to be believable and informative.

The National Academies of Sciences, Engineering, and Medicine (2019) has considered reproducibility and replicability in detail. They define these terms as follows (pp. 6–7):

> *Reproducibility* is obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis. This definition is synonymous with "computational reproducibility," and the terms are used interchangeably in this report.
>
> *Replicability* is obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data. Two studies may be considered to have replicated if they obtain consistent results given the level of uncertainty inherent in the system under study.
>
> Generalizability, another term frequently used in science, refers to the extent that results of a study apply in other contexts or populations that differ from the original one.[1] A single scientific study may include elements or any combination of these concepts. In short, reproducibility involves the *original* data and code; replicability involves *new* data collection to test for consistency with previous results of a similar study.

In this chapter we will begin an exploration of the world of R Markdown to integrate data analysis and reporting. Not only is R Markdown useful in promoting research reproducibility, it also makes it easier (and less error-prone) to update a report when incorporating new data, rerunning analyses, or making other changes that affect the analyses or reporting.

Markdown Template

We begin by creating a new Markdown file from the pull-down menu: **File → New File → R Markdown . . .** This opens the New R Markdown dialog box shown in Figure 7.1. In this dialog box, we have options to create a document (in HTML, PDF, and Word formats), presentation (in HTML, PDF, and PowerPoint formats), and an interactive document or presentation using Shiny[2], or we can use some other template (we can even create our own reusable template).

Let's choose to create a Word document. As noted in the dialog box, we can later change to another format. This opens the document

[1] The same definition of generalizability as used by NSF (Bollen et al., 2015).

[2] To learn more about Shiny, visit: *https://shiny.rstudio.com*.

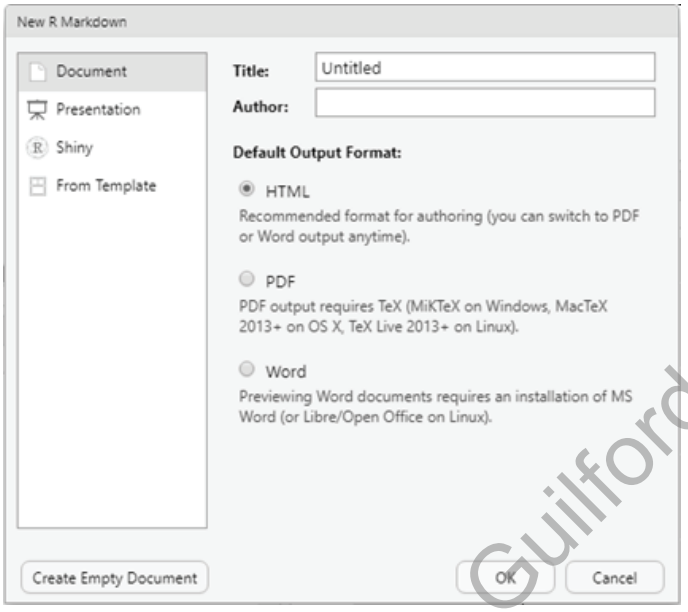

FIGURE 7.1. New R Markdown dialog box.

template (clicking the Create *Empty Document* button will open a blank file rather than a template).

Let's examine this file closely. Lines 1 through 4 contain the YAML (YAML Ain't Markup Language) header that presents options for how the document will be rendered. The YAML code appears between two lines of three hyphens (" --- "). Next are alternating chunks of white and gray areas. The white areas include text and text formatting commands. The gray areas contain R code.

Each chunk of R code begins with " ``` {r} " and ends with " ``` " (the backticks are located on the same keyboard key as the tilde). The chunks of R code can be named for convenience (that is, *setup*). They can also contain options; for example, *echo* = *FALSE* shows the results in the output document but does not show the code in the output document, and *include* = *FALSE* keeps both the code and the results from being displayed in the output document. The first chunk of R code includes setup options that apply to the entire document unless different options are specified for a later chunk of code. Each chunk of text contains text, and associated formatting, that will appear in the output document. For

example, the two hashmarks (" ## ") at the beginning of a line cause that line to appear in the output document as a level two header.

We can run individual chunks of R code by placing the cursor on the desired part(s) in the code and running it in the usual way (for example, by pressing Ctrl+Enter). We can also set chunk options and run chunks of code, using the icons at the upper right-hand corner of a code chunk. We can run the entire file, rendering the output document, by clicking the **Knit** button (the one with the ball of yarn and needles) just above the Markdown script, or from the pull-down menu from **File → Knit Document** or with **Cntrl + Shift + K**.

By looking at the file, we can anticipate what the output document will include: the title *Untitled*, a level two header that says *R Markdown* followed by some text, summary statistics for the two variables in the *cars* data frame (which is already available as part of Base R), another level two header followed by text, a graph plotting the two variables in the *pressure* dataframe, and then some final text. Go ahead and **Knit** the document, watch the progress in the Console, and then compare the document that is produced to the Markdown file that produced it.

Working in Markdown

Now let's create our own Markdown document. We will use the example from Terrell (2021) that we used for the chi-square analysis in Chapter 4.

Open Markdown Template, Enter First Lines of Code, and Knit

Open a new Word document template, either by clicking the *Create Empty Document* button in the bottom left corner of the dialog box or by opening the template without clicking that button and deleting the code it contains. We will enter our code part by part, knitting the file along the way to see how it develops and how we can control the output. Along the way, be aware that on occasion you may need to close an output file, or even exit and re-launch RStudio (remember to save your work first), just to reset.

Let's begin with the following lines of code. The first four lines are the YAML header, stating the title and author, and choosing to output

a Word document. The next set of lines (between the two sets of three backticks (“ ``` ”) contains the first chunk of R code. The first line of this chunk begins with a curly brace, is followed by the letter “r,” an optional name for the chunk (here it is *setup*), a comma, and the *include = FALSE* instruction to keep the chunk from being included in the output. Next comes an overall setting for the file, which is *echo = FALSE*, so that our default setting is to not include the R code in the output. We then load the packages we will use in this program (it is good to load all the packages we need at the beginning of the file). In the line that reads *# Load Packages*, the “ # ” at the beginning of the line indicates that it is a comment since it is inside a chunk of R code.

After the three backticks that end the R code chunk (in gray), we are in the text area (in white). In the line that reads *## Introduction*, the “ ## ” at the beginning of the line instructs R Markdown that this is a level two header. We then have a line of text, followed by an indented block indicated by the “ > ” at the beginning of the text.

````
---
title: "Graduation Status by Learning Style"
author: "Your Name Here"
output: word_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)

# Load packages
library(tidyverse)
library(rstatix)
library(flextable)

```

## Introduction

In this document we look at an example from Terrell (2021):

> Throughout the world, more and more colleges and universities
are introducing distance learning programs... Some people are
critical of these programs because they feel they may not equally
support all learning styles and therefore may lead to failure in
such a learning environment... By analyzing the interaction of
learning style and attrition, we will be able to determine if the
concerns about distance education are warranted. (p.327)
````

Go ahead and knit this file. The first time it is knitted we are prompted for a name for the file, since it gets saved at this point. Knitting progress is displayed in the Console. Once done, a Word document is opened, and you will see the title, author, level two header, and text that we wrote.

Create and Look at a Data Frame

Let's now add the next R code chunk. The first line begins the chunk, with three backticks, an open curly brace, the letter "r," and an optional name for the chunk. We then have groups of lines that create the dataframe and a second dataframe in long format, look at the first and last few lines of the data frame (commented out), create a third data frame with summary counts and percentages, and create an object that contains the results of the chi-square test. The code chunk ends with another set of three backticks.

````
```{r dataframe}

# Create data frame
dist_learn <- as.table(
  rbind(c(9, 16, 12, 10), c(5, 12, 6, 10)))

dimnames(dist_learn) <- list(
  Status = c("Graduate", "Non-graduate"),
  Learning_style = c("Accommodator", "Assimilator", "Diverger",
                     "Converger"))

# Convert counts to cases
dist_learn_long <- counts_to_cases(dist_learn)

# Look at data
#dist_learn_long %>%
#  slice(1:5, 76:80)

# Get counts to use in division later
summarized <- dist_learn_long %>%
  group_by(Status) %>%
  summarize(n = n()) %>%
  mutate(Percent = prop.table(n)) %>%
  mutate(Percent = Percent * 100) %>%
  mutate(Percent = round(Percent, 1))

# Conduct chi_square test
csq <- chisq.test(dist_learn_long$Status, dist_learn_long$Learning_style)

```
````

We can run the R code from within the R Markdown document by placing our cursor on the part of interest and then running it as usual. The result appears just below the code chunk (you may need to scroll down to see it). This is good to do, as once the file gets longer it is easier to test the code that way (and debug if necessary) than by knitting (rendering) the document. Go ahead and test the code, and once it is working, knit the file again. Since this code chunk did not produce anything that is displayed, none appears in the output document. We do see a message, but we know from Chapter 4 that in this instance the message is not important for us. Therefore, we can edit the first line of this code chunk to read:

```
```{r dataframe, message = FALSE}
```

Create the Crosstabulation Table

Let's now add the following lines of code. First, we have another level two header, followed by a line of text. Then we have an R code chunk (again, beginning and ending with three backticks). This chunk contains a comment, and then it uses the *proc_freq()* function from the *rstatix* package to create a *flextable* that displays a table in our Word document. The first line in the *proc_freq()* function names the variables to include in the table and provides a title for the table, the next two lines suppress the column and cell percentages, the two lines after that relabel a variable and suppress a column header, and the final line fits the table on the page.

```
## Crosstabulation table

Next, we create the crosstabulation table.

```{r crosstab, echo = FALSE}

# Create the crosstabulation table
dist_learn_long %>%
  proc_freq("Learning_style", "Status", "Status by Learning Style",
            include.column_percent = FALSE,
            include.table_percent = FALSE) %>%
  set_header_labels(Learning_style = "Learning Style",
                    label = "") %>%
  autofit()

```
```
```

Go ahead and knit the document again, and notice that the output now includes the *flextable*. Now is a good time to explore what this code is doing by altering it, re-knitting, and comparing the output to what it was before. I suggest trying these things one at a time and re-knitting with each change before returning to the original:

- Set *echo* to *TRUE*, delete "*Status by Learning Style,*"
- Set *include.column_percent* to *TRUE*,
- Sset *include.table_percent* to *TRUE*,
- Remove the *set_header_labels(. . .)* and note that this causes two differences, and
- Remove *autofit()* and the pipe (" %>% ") that precedes it.

This sort of practice is a good way to get familiar with writing code. It is especially beneficial if you work on predicting how the change will affect the result and then checking the accuracy of your prediction. If your prediction was not correct, think about why and how you would predict differently next time. This kind of exercise will help you in the future, when you want to accomplish something with your R code and you are figuring out the puzzle of how to do it.

Conduct a Chi-Square Test and Embed Results within Text

Now let's add some more code, all of which is in the R Markdown text area. The first line instructs R Markdown to begin a new output page (using the *\newpage* instruction). The next line is another level two header, which is followed by a line of text. We then use the asterisk at the beginning of the line to instruct R Markdown to create a bullet point. This bullet point includes embedded R code, which begins with a backtick followed by the letter "r" and ends with another backtick (*'r . . .'*). This bullet point uses the *nrow()* function to include the number of cases in the bullet point, so that we expect that in the output it will read "The results are based on the 80 cases in our data frame." One great thing about this is that if the data frame gets changed (for example, if more cases are added to it), this bullet point will automatically be updated the next time the file is knit. We then have two bullet points that draw results from our summarized data frame. The last bullet

point uses information from the object named *csq*, which we created to contain the results of the chi-square analysis. We use the Base R naming convention *object$variable* to refer to the items that we want. You may find it helpful to use the *view()* function to display the *csq* object so that you can take a look at it by entering and running *view(csq)* inside one of the R code chunks. Thus, this bullet point includes:

- The degrees of freedom *csq$parameter*,
- The number of cases *nrow(dist_learn_long)* (note that the asterisk on each side of the "*N*" in the following text input causes the "*N*" to be italicized),
- The chi-square statistic rounded to three decimal places *round(csq$statistic, 3)*, and
- The probability associated with the chi-square statistic rounded to two decimal places *round(csq$p.value, 2)* (note that the asterisk on each side of the "*p*" in the following text input causes the "*p*" to be italicized).

Following the bullet points are two lines that illustrate how to include comments within the text area, which is done by using the format " *<!-- . . . text . . . -->* " As noted in the last line, to indicate that R code embedded within text should be considered a comment, we need to include both the " *<!-- . . . -->* " for the text and the " # " for the R code.

```
\newpage

## Embed Results Into Text

As we can see from this table:

* The results are based on the `r nrow(dist_learn_long)` cases in
our data frame.

* The overall percentage who graduated was: `r summarized %>%
    filter(Status == "Graduate") %>%
    select(Percent)`.

* The overall percentage who graduated was: `r summarized %>%
    filter(Status == "Non-graduate") %>%
    select(Percent)`.
```

```
* The percentage who graduated was not different among the
different learning styles. Pearson's chi-square statistic
(`r csq$parameter`, *N* = `r nrow(dist_learn_long)`) =
`r round(csq$statistic, 3)`, *p* = `r round(csq$p.value, 2)`.

<!--    This is a comment in the RMarkdown text -->

<!-- To comment out a line with embedded R code, both the line and
        the R code need to be commented out, for example: -->
<!--    Some text here `r #some embedded R code here` then some
more text here. -->
```

Go ahead and knit the file again and study the output document side by side with the code that produced it. If necessary, debug the R Markdown file as needed.

Visualize the Results

Once the file is running correctly, let's add our last lines of code. We first have another level two header, which is followed by a line of text. We then have a new R code chunk, which uses the *ggplot()* function to create a graph. We next create a bar chart with *learning style* on the X axis and the *percent graduated* on the Y axis, set the limits of the Y axis, add a main label and labels for the two axes as well as data labels above the bars, and set the look of the background.

```
## Visualize the Results

These results are illustrated in the following graph.

```{r, graph, echo = FALSE}

# Create graph
ggplot(dist_learn_long,
       aes(x = Learning_style, y = ..prop.. * 100, group = 1)) +
  geom_bar() +
  ylim(0, 100) +
  labs(title = "Status by Learning Style",
       x = "Learning Style",
       y = "Percent Graduated") +
  geom_text(aes(label = round(..prop.. * 100, 0)),
            stat = "count", vjust = -0.5) +
  theme_bw()

```
```

Go ahead and again run the R code from within the R Markdown file. Once it is working, knit the file and examine the output file side by side with the code that produced it.

Before we end this chapter, let's look at one other output format, a PowerPoint presentation. Open a new R Markdown file from the pull-down menus (**File → New File → R Markdown . . .**) and in the **New R Markdown** dialog box select **Presentation** and **PowerPoint** (see Figure 7.2).

By now you are familiar enough with R Markdown that you can understand what is in the template (that is, the YAML header, R setup code chunk, text, R code chunk to produce a summary table, more text, and then another R code chunk). Go ahead and **knit** the template as is, and then compare the PowerPoint presentation output to the Markdown file that produced it. This exercise will help make even clearer how to work with Markdown as well as its potential applications. You can now use R Markdown to create your own presentations.

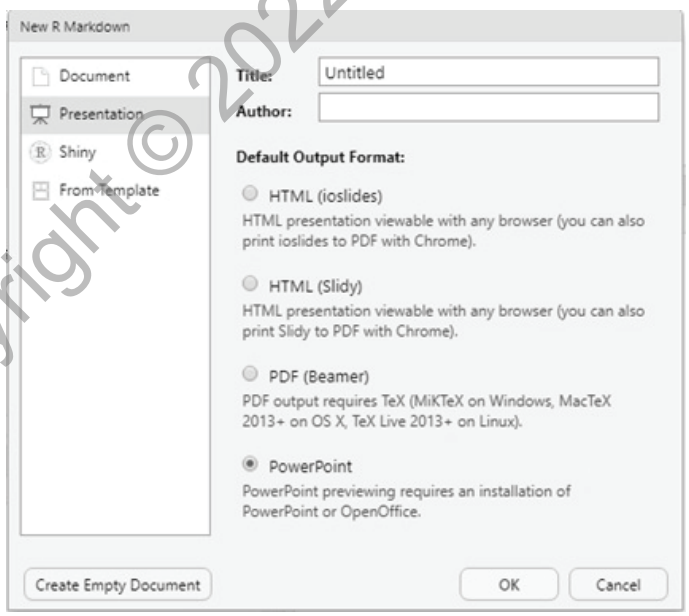

FIGURE 7.2. New R Markdown dialog box (presentation, PowerPoint).

KEY TAKEAWAYS

- R Markdown is a powerful tool for facilitating research reproducibility.
- It does this by integrating analysis code and reporting text.
- R Markdown can also facilitate creation of reports and presentations, and eases the task of updating them as necessary.
- Results of analyses and R objects or their elements can be integrated between blocks of text or inline with text.

EXERCISES

1. Using the data from Terrell (2021) on absences from class and scores on a final exam presented in Chapter 5, use R Markdown to create a Word document that provides descriptive statistics for the two variables, computes and tests the correlation coefficient, and displays a scatterplot with regression line (use *ggplot()* to create the scatterplot).
2. Repeat Exercise 1, but this time create an HTML document rather than a Word document as the output.

Guilford Publications
370 Seventh Avenue
New York, NY 10001
212-431-9800
800-365-7006
www.guilford.com