Technology Companion

Doing Statistics in R & RStudio

A hands-on companion to

You're Smarter Than You Think: Statistics

Safaa Dabagh

Keyed chapter-by-chapter to the main book

"The computer does the arithmetic. Your job is to ask the right question and read the answer. You can absolutely do that."

2026

Before You Type a Single Thing

If the word "coding" just made your shoulders climb toward your ears, breathe. R is not coding the way a movie shows coding. R is a very good calculator that you talk to one line at a time. You type a short instruction, you press a button, it answers. If you make a mistake, nothing breaks. You fix the line and press the button again. That is the entire loop.

This companion follows the main book chapter for chapter. When you finish Chapter 4 in the book, open Chapter 4 here and watch the same ideas happen in R. You never have to guess which tool goes with which idea.

One promise: every block of code in this book is short, and every block is followed by what the answer looks like and what it means. You are never left staring at output wondering what you are seeing.

Getting Set Up (Once)

You install two free things. R is the engine. RStudio is the comfortable dashboard you actually sit in.

1. **Install R** from `cran.r-project.org` (choose your operating system, download, install with the defaults).
2. **Install RStudio Desktop** from `posit.co/download/rstudio-desktop` (the free version). Install it after R.
3. Open **RStudio**. You will never open R directly again; RStudio runs it for you.

The four panes you'll see in RStudio:

- **Source** (top left): where you write and save your code, called a *script*. This is where you live.
- **Console** (bottom left): where answers appear. You can also type here for quick throwaway questions.
- **Environment** (top right): a list of the data you have created, so you can see what R is holding.
- **Plots / Files / Help** (bottom right): your graphs and help pages show up here.

Your very first line

In the Source pane, open a new script (File → New File → R Script), type the line below, put your cursor on it, and press **Run** (or Ctrl/Cmd + Enter).

R (type this in a script, then press Run)

```
2 + 2
```

What you'll see

```
[1] 4
```

That `[1]` just means "line 1 of the answer." You are now running R. That is the whole scary

part, done.

Getting Data Into R

Three ways, from simplest to most real. You will use all three across the book.

1. Type a short list yourself with `c()` ("combine"). The arrow `<-` means "save this as."

R (type this in a script, then press Run)

```
scores <- c(14, 15, 13, 15, 16, 12, 14, 15)
scores
```

What you'll see

```
[1] 14 15 13 15 16 12 14 15
```

2. Build a small table with `data.frame()` when you have more than one variable.

R (type this in a script, then press Run)

```
study <- data.frame(
  spot  = c("Home","Library","Cafe","Home","Home"),
  hours = c(3, 5, 2, 4, 3)
)
study
```

3. Open a file (a spreadsheet saved as `.csv`) with `read.csv()`. In RStudio you can also click Import Dataset in the Environment pane and it writes this line for you.

R (type this in a script, then press Run)

```
mydata <- read.csv("class_data.csv")
```

That is the whole toolkit. Now let's walk the book.

Chapter 1

What's the Story in the Data?

▪ WHAT THIS CHAPTER DOES IN R

Get data into R, tell R which variables are *words* (categories) and which are *numbers*, and look at your table without panic.

In Chapter 1 you learned the difference between a population and a sample, and between categorical and quantitative variables. In R, that difference is a real setting you turn on, and it decides what R will let you do later.

R (type this in a script, then press Run)

```
# a categorical variable (words) and a quantitative one (numbers)
spot  <- c("Home","Library","Cafe","Home","Home","Library")
score <- c(14,15,13,15,16,12,14,15,14,13,15,16,14,15,13,14)

spot <- factor(spot)    # tell R: these are CATEGORIES, not text
str(spot)               # str() = "structure": what kind of thing is this?
str(score)
```

What you'll see

```
Factor w/ 3 levels "Cafe","Home",..: 2 3 1 2 2 3
num [1:16] 14 15 13 15 16 12 14 15 14 13 ...
```

▪ READ THE OUTPUT

`Factor w/ 3 levels` means R now treats `spot` as three categories (Cafe, Home, Library). `num` means R treats `score` as numbers it can average. That is exactly the "words vs. numbers" choice from the chapter, made official.

▪ YOUR TURN

Make a variable `major` with five majors of your choice, turn it into a factor, and run `str()` on it. How many levels does R report?

Chapter 2

Where Did This Data Come From?

▪ WHAT THIS CHAPTER DOES IN R

Take a *simple random sample* in R, and see why `set.seed()` makes randomness repeatable.

Chapter 2 is about how good data is collected. The gold standard is the simple random sample: everyone has an equal chance. R can draw one for you.

R (type this in a script, then press Run)

```
students <- paste0("S", 1:40)    # 40 student IDs: S1 ... S40

set.seed(1)                      # makes the "random" draw repeatable
sample(students, size = 5)       # pick 5 of them at random
```

What you'll see

```
[1] "S9"  "S37" "S1"  "S23" "S6"
```

▪ READ THE OUTPUT

`set.seed(1)` is a promise: "start the randomness from the same spot." Anyone who runs `set.seed(1)` then this line gets the *same* five students. That is how researchers make a random study checkable. Change the seed number, get a different sample.

▪ YOUR TURN

Draw a random sample of 8 students. Then run the same two lines again. Same eight? Now delete the `set.seed` line and run `sample()` twice. What changes?

Chapter 3

Show Me a Picture

▪ WHAT THIS CHAPTER DOES IN R

Make the three core graphs from the chapter: a bar chart for categories and a histogram for numbers (plus a boxplot).

R uses your "words vs. numbers" choice to give you the right picture. Bar charts for categories, histograms for numbers.

R (type this in a script, then press Run)

```
# a bar chart of a categorical variable
barplot(table(spot),
        col = "#3A7CA5", main = "Where students study")

# a histogram of a quantitative variable
hist(score, breaks = 5,
     col = "#3A7CA5", main = "Quiz scores", xlab = "Score")

# a boxplot shows the middle, the spread, and outliers
boxplot(score, horizontal = TRUE, col = "#C8860D")
```

▪ READ THE OUTPUT

`table(spot)` counts each category first; `barplot()` draws one bar per count, with gaps. `hist()` bins the numbers into touching bars, so you see where scores pile up (here, around 14 to 15, just like the book). Each graph opens in the Plots pane, bottom right. Click Export to save it.

▪ YOUR TURN

Change `breaks = 5` to `breaks = 10` in the histogram. More bars or fewer? Does the "pile" still sit near 14 to 15?

Chapter 4

The Typical and the Spread

▪ WHAT THIS CHAPTER DOES IN R

Compute the center (mean, median) and the spread (standard deviation, IQR) with one word each.

Chapter 4's formulas are one-word commands in R. You do not compute them by hand here; you check your by-hand work against R.

R (type this in a script, then press Run)

```
mean(score)      # the average
median(score)    # the middle value
sd(score)        # standard deviation (typical distance from the mean)
IQR(score)       # the middle 50% (Q3 minus Q1)

summary(score)   # min, Q1, median, mean, Q3, max, all at once
```

What you'll see

```
[1] 14
[1] 14
[1] 1.153916
[1] 1.25
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   13.75   14.00   14.00   15.00   16.00
```

▪ READ THE OUTPUT

`summary()` is your five-number summary plus the mean, in one line. Here the mean and median are both 14, which says the scores are fairly symmetric. If the mean were pulled well above the median, that would warn you of a right skew or a high outlier, exactly the reasoning from the chapter.

▪ YOUR TURN

Add one unusually high score, say `score <- c(score, 40)`, and run `mean()` and `median()` again. Which one moved more? That is why we report the median for skewed data.

Chapter 5

What Are the Chances?

▪ WHAT THIS CHAPTER DOES IN R

Build a two-way table, add its margins, and read probabilities and conditional probabilities straight off it.

Chapter 5's two-way tables are where probability gets concrete. R turns counts into proportions for you.

R (type this in a script, then press Run)

```
# 100 people, cross-classified: had caffeine? and slept well?
tab <- as.table(matrix(c(30,20, 10,40), nrow = 2, byrow = TRUE,
        dimnames = list(Caffeine = c("Yes","No"),
                        Slept    = c("Yes","No"))))
addmargins(tab)              # add row, column, and grand totals
prop.table(tab)             # every cell as a share of all 100
prop.table(tab, margin = 1) # CONDITIONAL: within each caffeine row
```

What you'll see

```
        Slept
Caffeine Yes No Sum
     Yes  30 20  50
     No   10 40  50
     Sum  40 60 100
```

▪ READ THE OUTPUT

`prop.table(tab)` answers "what fraction of *everyone*?" `prop.table(tab, margin = 1)` answers "*given* they had caffeine, what fraction slept well?" That second one is conditional probability, the trickiest idea in the chapter, made into a single argument: `margin = 1` means "per row."

▪ YOUR TURN

Run `prop.table(tab, margin = 2)`. Now you are conditioning on the *columns* (slept well or not). In plain words, what question does that table answer?

Chapter 6

Predictable Randomness

▪ WHAT THIS CHAPTER DOES IN R

Get exact binomial and normal probabilities without a table in the back of the book.

Chapter 6's distribution tables are replaced by four small commands. For the normal curve: `pnorm` for "probability below," `qnorm` for "the value at a percentile."

R (type this in a script, then press Run)

```
# BINOMIAL: 10 quizzes, each correct with probability 0.7
dbinom(8, size = 10, prob = 0.7)   # P(exactly 8 correct)
pbinom(8, size = 10, prob = 0.7)   # P(8 or fewer correct)

# NORMAL: exam scores ~ Normal(mean = 70, sd = 8)
pnorm(80, mean = 70, sd = 8)       # P(score below 80)
qnorm(0.90, mean = 70, sd = 8)     # the 90th-percentile score
```

What you'll see

```
[1] 0.2334744
[1] 0.8506917
[1] 0.8943502
[1] 80.25326
```

▪ READ THE OUTPUT

`pnorm(80, 70, 8) = 0.894` says about 89% of scores fall below 80. `qnorm(0.90, 70, 8) = 80.25` runs it backward: to be in the top 10%, you need about 80.25. `p` takes a score and gives a probability; `q` takes a probability and gives a score. They are opposites.

▪ YOUR TURN

Find the probability a score is *above* 85. (Hint: R gives you "below," so compute `1 - pnorm(85, 70, 8)`.)

Chapter 7

From Sample to Truth

▪ WHAT THIS CHAPTER DOES IN R

Watch the Central Limit Theorem happen by simulating many samples and graphing their means.

Chapter 7 claims something wild: even from a lopsided population, the averages of samples pile up into a bell. In R you can see it, not just trust it.

R (type this in a script, then press Run)

```
set.seed(7)
pop <- rexp(100000, rate = 1)          # a very SKEWED population

# take 1000 samples of size 30, keep each sample's MEAN
means <- replicate(1000, mean(sample(pop, 30)))

hist(means, col = "#3A7CA5",
     main = "1000 sample means (n = 30)")
mean(means)       # centers near the population mean (which is 1)
sd(means)         # this is the standard error
```

What you'll see

```
[1] 0.9994
[1] 0.1804
```

▪ READ THE OUTPUT

The population is strongly skewed, yet the histogram of the 1000 *means* looks bell-shaped. That is the Central Limit Theorem, live. The `sd` of those means (about 0.18) is the *standard error*, and it matches the formula from the chapter: population sd over the square root of 30.

▪ YOUR TURN

Change the sample size from 30 to 5 (edit both spots). Does the bell get wider or narrower? Bigger samples give smaller standard error, exactly what the chapter promised.

Chapter 8

How Sure Are We?

▪ WHAT THIS CHAPTER DOES IN R

Build a confidence interval for a mean and for a proportion, and read it the way the chapter taught.

A confidence interval is a range of believable values for the truth. R hands you the whole interval.

R (type this in a script, then press Run)

```
# 95% confidence interval for the MEAN of the quiz scores
t.test(score)$conf.int

# 95% confidence interval for a PROPORTION: 18 of 40 said "yes"
prop.test(18, 40)$conf.int
```

What you'll see

```
[1] 13.38503 14.61497
attr(,"conf.level")
[1] 0.95
[1] 0.2963  0.6104
```

▪ READ THE OUTPUT

The mean interval, about `13.4 to 14.6`, says: we are 95% confident the true average score sits in there. The proportion interval, about `0.30 to 0.61`, does the same for the true "yes" rate. Want 90% instead of 95%? Add `conf.level = 0.90` inside the parentheses and watch the interval shrink.

▪ YOUR TURN

Re-run the mean interval with `t.test(score, conf.level = 0.99)$conf.int`. Wider or narrower than 95%? Why does being *more* confident cost you a *wider* net?

Chapter 9

Testing What We Believe

▪ WHAT THIS CHAPTER DOES IN R

Run a one-sample hypothesis test for a mean, a proportion, and a variance, and find the p-value.

Chapter 9 is the logic of the p-value. R runs the test and prints the p-value plainly; your job is to decide what it means.

R (type this in a script, then press Run)

```
# H0: the true mean score is 13
t.test(score, mu = 13)

# H0: the true "yes" proportion is 0.5   (18 of 40 said yes)
prop.test(18, 40, p = 0.5)

# H0: the true variance is 4 (a chi-square test, done directly)
n <- length(score); s2 <- var(score)
chi <- (n - 1) * s2 / 4
c(statistic = chi, df = n - 1, p_value = 1 - pchisq(chi, df = n - 1))
```

What you'll see

```
        One Sample t-test
t = 3.4641, df = 15, p-value = 0.003438
mean of x
       14
```

▪ READ THE OUTPUT

Find the `p-value` in the output. Here it is `0.003`, which is smaller than 0.05, so we reject the claim that the mean is 13; the evidence says the true mean is higher. Same move for the proportion and variance tests: read the p-value, compare it to your significance level, decide. R never decides for you, and that is the point of the chapter.

▪ CHECK YOUR VOICE

When the output scrolls by, the anxious voice says "I don't know what any of this means." You do. You are looking for one number, the p-value, and asking one question: smaller than 0.05 or not? Everything else is supporting detail.

Chapter 10

Comparing Two Groups

▪ WHAT THIS CHAPTER DOES IN R

Compare two group means (independent and paired) and two proportions.

Chapter 10 asks: is the difference between two groups real, or just noise? One command per situation.

R (type this in a script, then press Run)

```
groupA <- c(72,75,68,90,85,79)
groupB <- c(65,70,60,72,68,64)
t.test(groupA, groupB)              # two INDEPENDENT groups

before <- c(70,68,75,72)
after  <- c(74,71,79,77)
t.test(after, before, paired = TRUE)  # SAME people, before vs after

prop.test(c(18,25), c(40,45))       # two PROPORTIONS: 18/40 vs 25/45
```

What you'll see

```
        Welch Two Sample t-test
t = 2.49, df = 8.3, p-value = 0.036
mean of x mean of y
 78.16667  66.50000
```

▪ READ THE OUTPUT

The two group means are about 78 and 67, and the p-value `0.036` is below 0.05, so the gap is bigger than noise would easily explain. `paired = TRUE` is the whole difference between "two separate groups" and "the same people measured twice." Choosing the right one is the chapter's key decision; in R it is one argument.

▪ YOUR TURN

Run the independent test again but with `paired = TRUE` removed where it does not belong, and read the mean difference. Which design (independent or paired) matches a weight-loss study that weighs the same people in January and June?

Chapter 11

When Three or More Groups Disagree

▪ WHAT THIS CHAPTER DOES IN R

Run a one-way ANOVA and, if it is significant, find *which* groups differ.

When there are three or more groups, you do not run many t-tests. You run one ANOVA. Chapter 11's whole method is two lines in R.

R (type this in a script, then press Run)

```
score3 <- c(72,75,68,  80,83,78,  65,70,60)
grp    <- factor(rep(c("A","B","C"), each = 3))

fit <- aov(score3 ~ grp)    # "~" means "explained by"
summary(fit)                # the ANOVA table with the p-value
TukeyHSD(fit)               # WHICH pairs of groups differ
```

What you'll see

```
            Df Sum Sq Mean Sq F value  Pr(>F)
grp          2  392.7  196.33   15.44 0.00426 **
Residuals    6   76.3   12.72
```

▪ READ THE OUTPUT

Read `Pr(>F)`, the p-value: `0.004`, below 0.05, so at least one group's mean really differs. But ANOVA does not say which one. `TukeyHSD(fit)` then compares each pair (A vs B, A vs C, B vs C) and flags the real gaps. That two-step logic, "is there a difference, then where," is the heart of the chapter.

▪ YOUR TURN

Look at the `TukeyHSD` output. Which pair of groups has the largest difference? Does its interval include 0? (An interval that misses 0 means a real difference.)

Chapter 12

Counts, Categories, and Surprises

▪ WHAT THIS CHAPTER DOES IN R

Run both chi-square tests: goodness-of-fit (one variable) and independence (two variables).

Chapter 12 is about counts that surprise us. Are the categories as evenly split as expected? Are two categorical variables related? Two flavors of one test.

R (type this in a script, then press Run)

```
# GOODNESS OF FIT: a die rolled 60 times. Is it fair?
observed <- c(8, 12, 9, 11, 7, 13)      # counts for faces 1..6
chisq.test(observed)                     # H0: all faces equally likely

# INDEPENDENCE: are caffeine and sleep related?
chisq.test(tab)                          # tab is the 2x2 from Chapter 5
```

What you'll see

```
        Chi-squared test for given probabilities
X-squared = 2.8, df = 5, p-value = 0.7308
```

▪ READ THE OUTPUT

For the die, the p-value `0.73` is large (well above 0.05), so we do *not* have evidence the die is unfair; the ups and downs are ordinary luck. For the caffeine-and-sleep table, a small p-value would say the two variables are related, not independent. Same command, two questions, exactly as the chapter frames them.

▪ YOUR TURN

Change the die counts to something lopsided, like `c(2, 3, 4, 20, 6, 25)`, and re-run. Does the p-value drop below 0.05 now? What would you conclude about that die?

Chapter 13

Drawing the Line

▪ WHAT THIS CHAPTER DOES IN R

Measure correlation, fit a regression line, read its slope and R^2, and predict.

The last chapter draws the line through a scatterplot and asks what it means. R gives you the correlation, the line, and a prediction.

R (type this in a script, then press Run)

```
hours <- c(1,2,3,4,5,6)
grade <- c(52,58,63,70,74,80)

cor(hours, grade)          # correlation r: strength & direction
fit <- lm(grade ~ hours)   # fit the line: grade explained by hours
summary(fit)               # slope, intercept, R-squared, p-value

plot(hours, grade, pch = 19); abline(fit, col = "red")  # the picture
predict(fit, newdata = data.frame(hours = 3.5))          # predict at 3.5
```

What you'll see

```
[1] 0.9985
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  46.4667     0.6215   74.77 1.9e-07 ***
hours         5.6857     0.1596   35.63 3.7e-06 ***
Multiple R-squared:  0.9969
```

▪ READ THE OUTPUT

`cor = 0.9985` is a very strong positive relationship. The `hours` slope `5.69` means each extra study hour adds about 5.7 points. `R-squared = 0.997` says hours explain about 99.7% of the variation in grade (unusually clean, because this is teaching data). `predict()` then reads the line at 3.5 hours. Slope, strength, prediction: the whole chapter, on one screen.

▪ CHECK YOUR VOICE

You just fit a regression model, the thing that sounded impossible on day one. Notice that. The tools got smaller as the ideas got bigger, because the computer carries the arithmetic. You carried the understanding the whole way.

The Datasets, Chapter by Chapter

Every example in this book comes with a ready-made data file, so you can load the real numbers instead of typing them. All nine files live in a folder called `data`, sitting right next to this PDF. They are plain `.csv` files, which means they also open in **StatCrunch**, **jamovi**, Excel, Google Sheets, and (typed into lists) the **TI-84**. One set of data, every tool.

How to load a dataset (do this once per file)

In RStudio, the friendliest way is the button: Environment pane → Import Dataset → From Text (base), pick the file, click Import. RStudio writes and runs the line for you. Or type it yourself:

R (type this in a script, then press Run)

```
class <- read.csv("data/class.csv")    # load the file into a table called "
    class"
head(class)                             # peek at the first few rows
class$quiz_score                        # pull one column out with the $ sign
```

If R says it "cannot open the file," it is looking in the wrong place. Use Session → Set Working Directory → To Source File Location, then run the line again.

The nine datasets

File	Chapters	What's inside (columns)
`class.csv`	1, 3, 4, 8, 9	16 students: `student`, `study_spot` (Home/Library/Cafe), `quiz_score`
`population40.csv`	2	40 students to sample from: `student_id`, `gpa`
`caffeine_sleep.csv`	5, 12	100 people: `caffeine` (Yes/No), `slept_well` (Yes/No)
`survey40.csv`	8, 9	40 people: `person`, `response` (Yes/No; 18 said Yes)
`two_groups.csv`	10	Two independent groups: `id`, `group` (A/B), `score`
`before_after.csv`	10	Same people twice: `student`, `before`, `after`
`three_groups.csv`	11	Three groups for ANOVA: `id`, `group` (A/B/C), `score`
`dice60.csv`	12	A die rolled 60 times: `face` (1 to 6), `count`
`study_hours.csv`	13	`hours` studied and the `grade` earned (6 students)

Two chapters need no file. Chapter 6 (Predictable Randomness) uses R's built-in probability functions (`pnorm`, `dbinom`, and friends), and Chapter 7 (From Sample to Truth) *generates* its own data by simulation. Nothing to download for those two; the code makes the numbers.

Loading each one is the same move. For example:

R (type this in a script, then press Run)

```
caffeine <- read.csv("data/caffeine_sleep.csv")
table(caffeine$caffeine, caffeine$slept_well)   # rebuild the Chapter 5 two-way table

groups <- read.csv("data/two_groups.csv")
t.test(score ~ group, data = groups)            # the Chapter 10 two-group test
```

Notice `score ~ group`: once your data is a loaded table, you can use the tidy "`outcome ~ grouping`" form and pass `data =`. That is the same file, read the grown-up way.

Quick Reference: The Whole Book in R

One line per idea

Idea (chapter)	R
Make categories (1)	`factor(x)`
Random sample (2)	`set.seed(1); sample(x, n)`
Bar chart / histogram (3)	`barplot(table(x))` / `hist(x)`
Center & spread (4)	`mean, median, sd, IQR, summary`
Two-way table (5)	`prop.table(tab, margin=1)`
Normal / binomial (6)	`pnorm, qnorm, dbinom, pbinom`
Sampling distribution (7)	`replicate(1000, mean(sample(...)))`
Confidence interval (8)	`t.test(x)$conf.int`
One-sample test (9)	`t.test(x, mu=..)`, `prop.test`
Two groups (10)	`t.test(a, b)`, `paired=TRUE`
ANOVA (11)	`aov(y ~ g); TukeyHSD`
Chi-square (12)	`chisq.test(...)`
Regression (13)	`lm(y ~ x); summary; predict`

You did statistics, and you did it in code. That is not a small thing. Keep this page next to you; it is the whole course in thirteen lines.