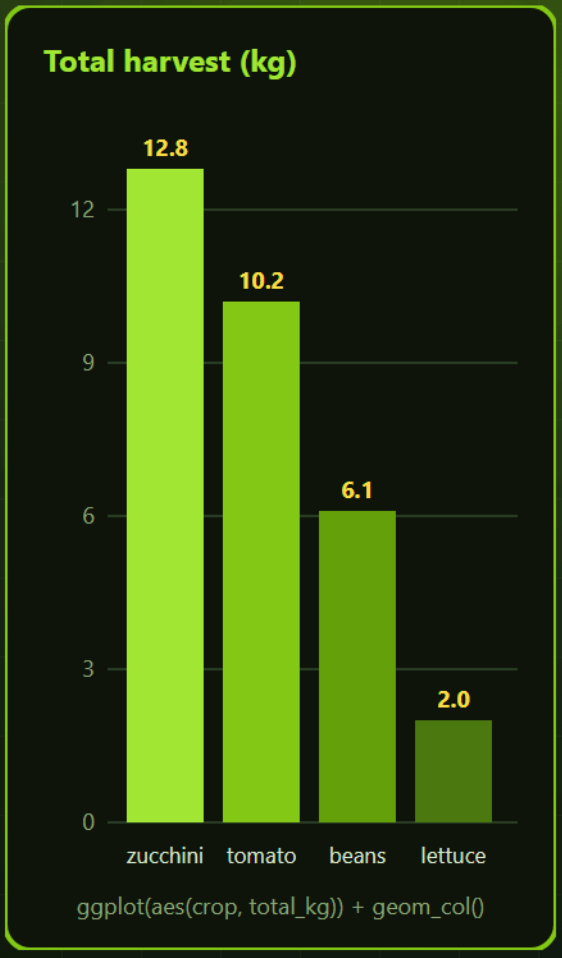
```r
# harvest by crop
harvests |>
  summarize(
    total_kg = sum(weight_kg,
      na.rm = TRUE),
    .by = crop) |>
  arrange(desc(total_kg))
```

```
# A tibble: 4 × 2
  crop     total_kg
  <chr>       <dbl>
1 zucchini     12.8
2 tomato       10.2
3 beans         6.1
4 lettuce       2
```



FROM FIRST STEPS TO INSIGHT

R Programming for Data Analysis

Import, transform, visualize, model, and report with the tidyverse

dplyr · tidyr · ggplot2 · statistics · Quarto · Shiny

BEGINNER FRIENDLY · STEP BY STEP · REPRODUCIBLE

THENEXBOOKS

R Programming for Data Analysis

From First Steps to Insightful Reports with the Tidyverse

Import · Transform · Visualize · Model · Communicate

First Edition

THENEXBOOKS

R Programming for Data Analysis
From First Steps to Insightful Reports with the Tidyverse

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. R, its packages, and its tools evolve continuously; printed output may differ slightly between versions. Always consult the documentation for the versions you use.

R is developed by the R Core Team as part of the R Project for Statistical Computing; this book is not affiliated with, sponsored by, or endorsed by the R Foundation or the R Core Team. RStudio, Positron, Quarto, Shiny, and the tidyverse are products and trademarks of Posit Software, PBC, or their respective owners; their use here is for identification only and does not imply endorsement.

Hollowbrook Community Gardens, the running example in this book, is fictional, as are all gardeners, plots, and data in its examples. Any resemblance to real organizations or people is coincidental.

Preface

Data is everywhere, but insight is rare. A spreadsheet of harvest weights, a survey export, a log of sales or sensor readings — each holds answers to useful questions, if you can clean it, combine it, summarize it, see it, test it, and explain what you found to someone else. R was built for exactly that work. It began as a language for statisticians, and today it is one of the most capable environments anywhere for data analysis, visualization, and reproducible reporting, with a friendly modern toolkit — the tidyverse — that makes everyday tasks read almost like sentences.

This book teaches R from the very first line of code to a finished, reproducible report. You do not need any programming background. We start with values, vectors, and functions, then move to the tidyverse workflow that most R analysts use every day: importing data, transforming it with dplyr, reshaping it with tidyr, joining tables, and cleaning real-world strings, categories, dates, and missing values. Then we visualize with ggplot2, learn the statistics that matter for everyday analysis — descriptive statistics, confidence intervals, hypothesis tests, and regression — and finish by communicating results with Quarto reports and Shiny dashboards.

Throughout, we favor understanding over recipes. Every function is introduced with a small, concrete example you can check by eye, and every statistical result is explained honestly, including when the data is simply too small to support a confident conclusion.

Who This Book Is For

- **Beginners** who want to learn programming through data analysis rather than abstract exercises.
- **Spreadsheet users** — analysts, researchers, students, and small-organization staff — ready to make their work faster and reproducible.
- **Students** in statistics, science, social science, or business courses that use R.
- **Programmers from other languages** who need a clear path into R and the tidyverse.

What You Will Learn

- R fundamentals: values, types, vectors, factors, lists, data frames, control flow, and functions.
- A reproducible project workflow and importing data from CSV and Excel files.
- Transforming, grouping, summarizing, reshaping, and joining data with dplyr and tidyr.
- Cleaning text, categories, dates, and missing values with stringr, forcats, and lubridate.
- Iteration and functional programming with purrr.
- Visualization with ggplot2: the grammar of graphics, distributions, relationships, polish, and facets.
- A complete exploratory data analysis.
- Descriptive statistics, confidence intervals, hypothesis tests, linear and logistic regression, and an introduction to tidymodels.

- Reproducible reports with Quarto, better code with tests and packages, and interactive dashboards with Shiny.

How This Book Is Organized

Part I — Getting Started with R (Chapters 1–5) installs R and teaches the language basics. **Part II — The Tidyverse Workflow** (Chapters 6–10) builds the everyday import-transform-summarize-reshape-join workflow. **Part III — Working with Real-World Data** (Chapters 11–15) handles strings, categories, dates, missing values, and iteration. **Part IV — Data Visualization with ggplot2** (Chapters 16–20) turns data into clear graphics and ends with a full exploratory analysis. **Part V — Statistics and Modeling** (Chapters 21–24) covers the statistics every analyst needs. **Part VI — Communicating and Scaling Up** (Chapters 25–28) produces reports and dashboards and brings everything together in a case study. The appendices provide a cheat sheet, a function reference with templates, an analysis checklist, and a glossary.

Each chapter ends with a **Key Takeaways** box for quick review, and most include short practice exercises.

The Running Example: Hollowbrook Community Gardens

Throughout the book we help the volunteer data team of **Hollowbrook Community Gardens**, a fictional network of community gardens at three sites — North, Riverside, and Hilltop. Six gardeners tend six plots; they record every harvest, the gardens log monthly rainfall, and members answer a short satisfaction survey. The data is deliberately small, so you can check every result by eye and focus on understanding each step — and it has the same problems real data has: a missing value, categories to tidy, dates to parse, tables to join, and questions that tempt us to over-interpret a small sample. By the final chapter, you will have produced the gardens' annual report.

Software and Conventions

Examples use a recent version of R, the native pipe `|>`, and the tidyverse packages, written in an RStudio or Positron project. Code appears in `monospace`, and the results R prints are shown in blocks labeled **Output**. Because this is a printed book, plots are described in words; run the code to see each figure on your screen. Small details of printed output, such as spacing or the number of digits shown, can vary slightly between package versions.

> **TIP**
> A helpful technique or a shortcut that makes your work easier.

> **NOTE**
> Background information, version details, or a useful aside.

WARNING

A common mistake that leads to wrong results, errors, or misleading conclusions.

Contents

Preface

PART I — GETTING STARTED WITH R

1 Why R? Data Analysis, Statistics, and the R Ecosystem

2 Installing R and an IDE, and Your First Session

3 R Basics: Values, Types, Vectors, and Arithmetic

4 Data Structures: Factors, Lists, Matrices, and Data Frames

5 Control Flow and Writing Your Own Functions

PART II — THE TIDYVERSE WORKFLOW

6 Projects, Packages, and Importing Data with readr and readxl

7 Transforming Data with dplyr: filter, arrange, select, and mutate

8 Grouping and Summarizing: summarize, .by, Counts, and Window Functions

9 Tidy Data and Reshaping with tidyr

10 Joining Tables with dplyr

PART III — WORKING WITH REAL-WORLD DATA

11 Strings and Regular Expressions with stringr

12 Factors and Categorical Data with forcats

13 Dates and Times with lubridate

14 Missing Values, Data Cleaning, and Validation

15 Iteration and Functional Programming with purrr

PART IV — DATA VISUALIZATION WITH GGPLOT2

16 ggplot2 Fundamentals: The Grammar of Graphics

17 Visualizing Distributions, Relationships, and Trends

18 Polishing Plots: Scales, Labels, Themes, and Annotations

19 Facets, Combined Plots, and Exporting Figures

20 Exploratory Data Analysis: A Complete Walkthrough

PART V — STATISTICS AND MODELING

21 Descriptive Statistics and Probability Basics

22 Statistical Inference: Confidence Intervals and Hypothesis Tests

23 Linear Regression and Model Diagnostics

24 Beyond Linear Regression: Logistic Regression and an Introduction to tidymodels

PART VI — COMMUNICATING AND SCALING UP

25 Reproducible Reports with Quarto

26 Writing Better R Code: Functions, Debugging, Testing, and Packages

27 Interactive Dashboards with Shiny

28 Case Study: The Hollowbrook Community Gardens Annual Report

APPENDICES

A R and Tidyverse Cheat Sheet

B Function Reference and Templates

C Data Analysis Checklist

D Glossary

PART I

Getting Started with R

Why R is a great home for data analysis, how to install it with a modern IDE, and the language itself: values, vectors, data structures, control flow, and your own functions.

1 · Why R?
2 · Installing R and Your First Session
3 · Values, Types, and Vectors
4 · Data Structures
5 · Control Flow and Functions

CHAPTER 1 · PART I — GETTING STARTED WITH R

Why R? Data Analysis, Statistics, and the R Ecosystem

Every spreadsheet, survey, and sensor log holds answers waiting to be found. This chapter explains what R is, why so many analysts rely on it, and how this book will take you from your very first line of code to a polished, reproducible report.

Part I at a Glance

Part I of this book is your foundation. Over five chapters you will learn what R is and why it exists (this chapter), how to install R and a comfortable editor and run your first commands (Chapter 2), how R represents numbers, text, and true/false values (Chapter 3), how R organizes those values into larger structures such as data frames (Chapter 4), and finally how to make decisions and write your own functions (Chapter 5). You do not need any programming experience to begin. If you can use a spreadsheet, send an email, and save a file into a folder, you have every skill this part assumes.

We take a deliberately gentle pace in these first chapters. Programming languages reward a firm foundation: a few minutes spent truly understanding what a vector is, or why a missing value behaves the way it does, will save you hours of confusion later when you are working with real data and real deadlines. When something in Part I seems almost too simple, enjoy it. The ideas are small, but they are the bricks from which everything else in the book is built.

What Is Data Analysis?

Data analysis is the process of collecting, cleaning, exploring, summarizing, and interpreting data so that people can make better decisions. That definition sounds grand, but the everyday reality is often modest. A shop owner wants to know which products sell best in winter. A teacher wants to see whether a new reading program changed test scores. A volunteer group wants to understand which of its garden plots produce the most vegetables. In each case there is a question, there is some data, and there is a need to turn that data into a clear, trustworthy answer.

Most analyses, large or small, follow a similar rhythm. It is worth naming the steps now, because the structure of this book mirrors them:

1. **Import.** Bring the data into your tool from a file, a database, or a website.
2. **Tidy.** Arrange the data into a consistent shape, usually one row per observation and one column per variable.
3. **Transform.** Filter rows, create new columns, and compute summaries such as totals and averages.

4. **Visualize.** Draw charts that reveal patterns, outliers, and mistakes that tables hide.
5. **Model.** Use statistics to describe relationships and to measure how confident you can be in them.
6. **Communicate.** Share the results in a report, a slide, or an interactive dashboard that others can read and trust.

In practice these steps form a loop rather than a straight line. A chart reveals a strange value, so you go back and clean the data; a model raises a new question, so you transform the data in a new way. A good tool makes it cheap to go around that loop many times. That, in one sentence, is the strongest argument for R.

What Is R?

R is a free, open-source programming language and software environment designed for statistical computing and graphics. It was created in the early 1990s by Ross Ihaka and Robert Gentleman at the University of Auckland in New Zealand, as an open implementation inspired by the earlier S language from Bell Laboratories. Today R is maintained by a group of volunteers known as the R Core Team and is released under the GNU General Public License, which means you can download it, use it for any purpose, and share it without paying anything.

Two words in that description deserve a closer look. R is a **language**: you give it instructions by typing text, called **code**, rather than by clicking menus. R is also an **environment**: when you start it, you get an interactive session in which you can type a command, press Enter, and see the result immediately. This combination of a real programming language with an interactive, conversational workflow is what makes R so pleasant for exploring data. You can ask a quick question, look at the answer, and ask a better question a few seconds later.

Here is what a tiny conversation with R looks like. We create a short list of numbers and ask for their average. Do not worry about the details yet; Chapter 2 walks you through running code yourself, and Chapter 3 explains every symbol you see here.

Console

```
weights <- c(3.2, 4.1, 1.8, 2.6)
mean(weights)
```

Output

```
[1] 2.925
```

The first line stores four harvest weights, in kilograms, under the name `weights`. The second line asks R for their mean, and R replies with `2.925`. The `[1]` at the start of the output is R telling you that this is the first (and here, the only) value in the answer. You will see that little bracket in front of almost every result in this book.

Why Choose R?

There are many tools for working with data, from spreadsheets to general-purpose programming languages. R occupies a special place among them. Here are the main reasons people choose it, and the reasons we chose it for this book.

It was built for data from the start

Many programming languages were designed for building websites, games, or operating systems, and data analysis was added later through extra libraries. R is different: it was designed by statisticians, for people analyzing data. Concepts that matter to analysts, such as missing values, categorical variables, and tables with columns of different types, are built into the core of the language. For example, R has a special value, `NA`, that means "this value is missing." You will meet it many times, because real data is almost never complete.

Statistics are first-class citizens

R ships with an enormous collection of statistical functions: means and medians, standard deviations, correlations, t-tests, linear regression, and much more. Methods that would require a specialist add-on in other tools are often a single function call in R. Because statisticians themselves use R for research, new statistical methods frequently appear as R code early, sometimes alongside the academic papers that introduce them.

Graphics that explain

R has always been strong at producing charts, and the **ggplot2** package, which you will study in Part IV, is widely admired for the quality and flexibility of its graphics. Instead of choosing a chart type from a menu, you describe how your data should map to visual properties such as position, color, and size, and ggplot2 builds the chart from that description. Once you learn the idea, you can create almost any statistical chart you can imagine.

Reproducibility

When you analyze data in a spreadsheet, many of your steps live only in your memory: which rows you deleted, which formula you dragged down, which cells you fixed by hand. If someone asks "How did you get this number?" six months later, the honest answer may be "I am not sure." **Reproducibility** means that someone else (including future you) can repeat your analysis exactly and get the same result. Because an R analysis is written as code, the code itself is a complete, readable record of every step. Run it again on the same data and you get the same answer; run it on next year's data and you get next year's answer with almost no extra work.

A huge, generous community

R is supported by a large, friendly community of users around the world. Thousands of people have written **packages**, which are bundles of code, data, and documentation that add new abilities to R. The official home for these packages is **CRAN**, the Comprehensive R Archive Network, which hosts

many thousands of them, all free and all checked by automated tests before they are published. There are packages for reading Excel files, drawing maps, analyzing text, building web dashboards, and nearly any other data task you can think of. When you get stuck, the chances are excellent that someone has already asked your question in a forum and received a helpful answer.

It is free

R costs nothing, runs on Windows, macOS, and Linux, and imposes no license limits on how many computers you install it on or what you use it for. For students, volunteers, small organizations, and anyone learning on their own, this matters a great deal. Your skills are never locked behind a subscription.

> **NOTE**
> R is not the only good choice for data work. Python is another popular, free language with excellent data libraries, and many professionals use both. The concepts you learn in this book, such as tidy data, grouping and summarizing, joining tables, and reproducible reports, transfer directly to other tools. Learning R well makes you a better analyst in any language.

R Compared with Spreadsheets

Most readers arrive at R with some spreadsheet experience, so it is worth being clear about how the two differ. Spreadsheets are wonderful for small, quick tasks: typing in a budget, making a simple chart, sharing a table with a colleague. They let you see and touch every cell, which is reassuring. R does not replace that experience; it complements it.

Question	Spreadsheet	R
How are steps recorded?	Mostly in your memory, spread across cells	In a script you can read, share, and rerun
Repeating for new data?	Often manual copying and pasting	Rerun the same code on the new file
Size of data?	Comfortable up to moderate sizes	Comfortable with much larger tables
Statistics?	Basic functions, some add-ons	Very broad, built in and via packages
Charts?	Quick, menu-driven	Highly customizable, code-driven
Learning curve?	Gentle at first	Steeper at first, then much faster

The last row is important and honest. R asks more of you at the beginning. You have to learn some vocabulary and accept that a missing comma will produce an error message. In exchange, tasks that would take an afternoon of careful clicking become a few lines of code you can run in a second, as often as you like. This book is designed to carry you up that early slope as smoothly as possible.

Meet Hollowbrook Community Gardens

Learning a language is easier with a story to follow. Throughout this book we work with one running example: **Hollowbrook Community Gardens**, a fictional, volunteer-run network of community gardens with three sites called North, Riverside, and Hilltop. Local residents rent small plots, grow vegetables, and record what they harvest. A small volunteer data team wants to use R to understand the harvests, compare the sites, look at rainfall, check how satisfied gardeners are, and publish a friendly annual report for members and funders.

The data is intentionally small. Small data lets you check every result by eye, which is exactly what you want while learning. When you see that the total harvest is 31.1 kilograms, you can add up the numbers yourself and confirm that R is right. Once the ideas are solid, the very same code works on tables with thousands or millions of rows.

The Hollowbrook team keeps five tables. You will create them as files in Chapter 6, but it helps to meet them now:

Table	Rows	What each row describes
`gardeners`	6	A gardener: ID, name, date joined, age group
`plots`	6	A plot: ID, site, size in square meters, soil, gardener
`harvests`	12	One harvest: ID, plot, date, crop, weight in kg
`rainfall`	3	A site's rainfall in July and August, in mm
`survey`	6	A gardener's satisfaction rating from 1 to 5

The table you will see most often is `harvests`. Here it is in full. Notice harvest H09: the gardener forgot to weigh it, so its weight is recorded as `NA`, meaning "not available." That single missing value will teach us a surprising amount about how R handles incomplete data.

harvest_id	plot_id	date	crop	weight_kg
H01	P01	2025-07-05	tomato	3.2
H02	P01	2025-07-19	zucchini	4.1
H03	P02	2025-07-06	beans	1.8
H04	P02	2025-08-02	tomato	2.6
H05	P03	2025-07-12	lettuce	1.1
H06	P03	2025-08-09	tomato	4.4
H07	P04	2025-07-20	zucchini	3.7
H08	P04	2025-08-16	beans	2.3
H09	P05	2025-07-27	tomato	NA

harvest_id	plot_id	date	crop	weight_kg
H10	P05	2025-08-23	lettuce	0.9
H11	P06	2025-08-03	zucchini	5.0
H12	P06	2025-08-30	beans	2.0

Take a moment to read the table the way an analyst would. Each row is one harvest event. The `plot_id` column links each harvest to a plot in the `plots` table, which in turn tells us the site and the soil type. The `date` column spans July and August 2025. Four crops appear: tomato, zucchini, beans, and lettuce. Already you can imagine questions: Which crop produced the most? Did August beat July? Do bigger plots produce more? By the end of this book you will answer all of them with R, and you will also learn how much (and how little) twelve rows of data can really tell you.

A First Taste of R with Hollowbrook Data

Before we install anything, let us look at a short piece of R code that answers a real Hollowbrook question: what was the total harvest, and what was the average weight of a harvest? You will not run this code until Chapter 2, and you do not need to understand every symbol. The goal is simply to see how R reads, and how close the code is to the question in plain English.

Console

```
weight_kg <- c(3.2, 4.1, 1.8, 2.6, 1.1, 4.4, 3.7, 2.3, NA, 0.9, 5.0, 2.0)
sum(weight_kg, na.rm = TRUE)
```

Output

```
[1] 31.1
```

The first line collects all twelve harvest weights, including the missing one, and stores them under the name `weight_kg`. The second line adds them up. The piece `na.rm = TRUE` means "remove missing values before calculating." The answer, 31.1 kilograms, is the total of the eleven harvests that were actually weighed. Now look at what happens when we ask for the mean and forget that instruction:

Console

```
mean(weight_kg)
mean(weight_kg, na.rm = TRUE)
```

Output

```
[1] NA
[1] 2.827273
```

The first answer is `NA`. At first this can feel unhelpful, but it is R being careful and honest. If one weight is unknown, then the true average of all twelve harvests is also unknown; it could be higher or lower depending on what H09 weighed. R refuses to guess. When we explicitly say that we want the average of the known values, R gives us 2.827273 kilograms. This small example captures something important about R's personality: it does not hide problems in your data. It makes you decide, in writing, how to handle them, and that decision stays in your code for anyone to see.

> **TIP**
> Whenever a summary in R returns `NA` unexpectedly, your first suspect should be a missing value somewhere in the data. Adding `na.rm = TRUE` is often the fix, but first ask yourself whether ignoring the missing values is the right decision for your question. Chapter 14 is devoted to this topic.

Here is a slightly bigger taste, using the tidyverse tools you will learn in Part II. Suppose the `harvests` table has already been loaded. This code keeps only the weighed harvests, counts them and totals their weight for each crop, and sorts the result alphabetically by crop:

analysis/crop-totals.R

```
harvests |>
  filter(!is.na(weight_kg)) |>
  summarize(n = n(), total_kg = sum(weight_kg), .by = crop) |>
  arrange(crop)
```

Output

```
# A tibble: 4 × 3
  crop         n total_kg
  <chr>    <int>    <dbl>
1 beans        3      6.1
2 lettuce      2      2
3 tomato       3     10.2
4 zucchini     3     12.8
```

Even without any training, you can probably follow the logic. The symbol `|>` is called the **pipe**; read it aloud as "and then." Take the harvests, *and then* filter out the missing weights, *and then* summarize by crop, *and then* arrange by crop. The result is a small table showing that zucchini produced the most weight, 12.8 kilograms from three harvests, while lettuce produced 2 kilograms from two. Tomato shows three harvests rather than four because the unweighed H09 was filtered out. You will write code like this yourself in Chapters 7 and 8.

The R Ecosystem

When people say "R," they often mean a whole family of tools that work together. Knowing the names of the main pieces will help you read documentation, follow tutorials, and search for help.

Base R

Base R is the language and the set of functions that come with every installation of R. It includes arithmetic, vectors, data frames, statistical functions such as `mean()` and `lm()`, a graphics system, and a help system. You can do a remarkable amount with base R alone, and Part I of this book focuses mostly on it, because base R concepts underpin everything else.

Packages and CRAN

A **package** extends R with new functions, data, and documentation. You install a package once, using `install.packages()`, and then load it in each session where you need it, using `library()`. Most packages come from CRAN, which also distributes R itself. Chapter 2 shows you how to install your first packages, and Chapter 6 explains packages in more depth.

The tidyverse

The **tidyverse** is a collection of packages that share a common design philosophy, a consistent style, and a shared idea of what well-organized data looks like. Installing the `tidyverse` package installs all of its core members at once. This book uses the tidyverse heavily from Part II onward, because it makes everyday data analysis readable and consistent. The core packages are:

Package	Purpose	Where in this book
`readr`	Reading text files such as CSV	Chapter 6
`dplyr`	Filtering, transforming, summarizing, joining	Chapters 7, 8, 10
`tidyr`	Reshaping data into tidy form	Chapter 9
`stringr`	Working with text	Chapter 11
`forcats`	Working with categories (factors)	Chapter 12
`lubridate`	Working with dates and times	Chapter 13
`purrr`	Repeating tasks over lists	Chapter 15
`ggplot2`	Building charts	Chapters 16 to 20
`tibble`	A modern version of the data frame	Chapters 4 and 6

Beyond the core, we will also use a few related packages: `readxl` for Excel files, Quarto for reports, Shiny for interactive dashboards, and tools such as `testthat` and `renv` for writing robust, long-lived code.

Editors: RStudio and Positron

You could type R code into a plain text editor and run it from a terminal, but almost nobody does. Instead, people use an **integrated development environment**, or **IDE**: an application that combines a code editor, the R console, a file browser, a plot viewer, and a help viewer in one window.

The two most popular IDEs for R are **RStudio** and **Positron**, both made by the company Posit. Both are free for individual use, and either works perfectly with this book. Chapter 2 helps you install one and find your way around it.

Reports and dashboards

An analysis is only useful if people can read it. **Quarto** lets you combine prose, R code, tables, and charts in a single document that renders to a web page, a PDF, or a Word file. Its predecessor, R Markdown, is still widely used and works in a very similar way. **Shiny** lets you turn R code into an interactive web application, such as a dashboard where members choose a site from a menu and see its harvest chart update instantly. Part VI covers both, and Chapter 28 brings everything together in the Hollowbrook annual report.

Who Uses R?

R is used wherever people need to learn from data. University researchers in biology, ecology, psychology, economics, and medicine use it to analyze experiments. Public health teams use it to track trends. Journalists use it to check claims and build charts for news stories. Businesses use it for sales forecasting, marketing analysis, and quality control. Nonprofits and volunteer groups, like our Hollowbrook data team, use it to understand their programs and report to their communities. What these users share is not a job title but a need: to answer questions with data in a way that is careful, repeatable, and clear.

You do not need to be a statistician or a professional programmer to benefit. Many of the most productive R users came to it from other fields and learned exactly as much programming as they needed to answer their questions. That is the path this book offers you.

How This Book Is Organized

The book is divided into six parts that follow the data analysis rhythm described earlier:

- **Part I, Getting Started with R** (Chapters 1 to 5), teaches the language itself: values, vectors, data structures, decisions, and functions.
- **Part II, The Tidyverse Workflow** (Chapters 6 to 10), shows how to organize a project, import data, transform and summarize it, reshape it, and join tables together.
- **Part III, Working with Real-World Data** (Chapters 11 to 15), handles the messy parts: text, categories, dates, missing values, and repetitive tasks.
- **Part IV, Data Visualization with ggplot2** (Chapters 16 to 20), teaches you to build, polish, and combine charts, ending with a complete exploratory analysis.
- **Part V, Statistics and Modeling** (Chapters 21 to 24), introduces descriptive statistics, confidence intervals, hypothesis tests, and regression, with an honest eye on what small samples can and cannot show.

- **Part VI, Communicating and Scaling Up** (Chapters 25 to 28), covers reproducible reports, better code, interactive dashboards, and the final Hollowbrook case study.

The chapters are designed to be read in order, since each builds on the ones before it. If you already know some R, you can skim Part I, but we suggest at least reading the Key Takeaways at the end of each chapter to make sure there are no gaps.

How to Read This Book

A few conventions will make the pages easier to follow.

- **Code blocks** appear in a monospaced font. A label above each block tells you where the code goes: *Console* means you type it directly into the R console, while a file name such as `analysis/crop-totals.R` means the code belongs in a script file with that name.
- **Output blocks**, labeled *Output*, show exactly what R prints when the code runs. Your output should match, apart from small differences such as the width of your window.
- **Function names** are written with parentheses, such as `mean()`, to make it clear that they are functions.
- **New terms** appear in bold the first time they are defined.
- **Callout boxes** labeled Tip, Note, and Warning highlight shortcuts, background information, and common mistakes.

In R code, anything after a `#` symbol on a line is a **comment**: a note for human readers that R ignores. We use comments to explain code as we go, and we encourage you to do the same in your own scripts.

> **WARNING**
>
> Resist the temptation to copy and paste code from this book. Typing it yourself is slower, but it trains your fingers and your eyes to notice details such as commas, quotes, and brackets. Most early errors in R come from exactly those small details, and you learn to spot them fastest by making and fixing them yourself.

Learning R Effectively

Learning a programming language is more like learning a musical instrument than memorizing facts. A few habits make a big difference:

- **Practice little and often.** Twenty minutes a day of typing and experimenting beats a single long session each week.
- **Experiment freely.** Change a number, remove an argument, swap two lines, and see what happens. You cannot break your computer by running R code on a small vector.

- **Read error messages slowly.** They often look intimidating, but they usually tell you exactly what went wrong, such as "object not found" when you misspell a name.
- **Check results by hand.** With small data like Hollowbrook's, you can confirm a total or an average with a calculator. Doing so builds trust in both R and yourself.
- **Use the help system.** Every function has a help page, which you open by typing a question mark before its name, such as `?mean`. Chapter 2 shows you how to read one.

Finally, be kind to yourself. Every experienced R user has stared at an error message that made no sense, only to discover a missing comma. Confusion is not a sign that you are failing; it is a normal and temporary part of learning.

Practice

1. Using the `harvests` table shown in this chapter, add up the weights of the four tomato harvests by hand. What happens because of H09? Write one sentence explaining what you think the honest total should be. *Hint:* compare your answer with the tomato row in the crop table above.
2. List three questions about Hollowbrook that the five tables could help answer. For each, name the tables you would need. *Hint:* questions about soil need `plots` as well as `harvests`.
3. Think of a data task you currently do in a spreadsheet. Write down each step you perform. Which steps would be easiest to forget if you had to repeat the task next year?
4. Read the pipeline in the "First Taste" section aloud, saying "and then" for each `|>`. Describe in your own words what each line does. *Hint:* there are four steps.

Key Takeaways

- Data analysis is a loop of importing, tidying, transforming, visualizing, modeling, and communicating, and this book follows that loop.
- R is a free, open-source language and interactive environment built specifically for statistics and graphics.
- Writing an analysis as code makes it reproducible: anyone can rerun it and get the same answer.
- R treats missing values with care: `mean()` returns `NA` when data is missing unless you say `na.rm = TRUE`.
- Packages, mostly from CRAN, extend R; the tidyverse is a family of packages with a consistent style for everyday data work.
- RStudio and Positron are free IDEs that make writing and running R code comfortable.
- The Hollowbrook Community Gardens data is small on purpose, so you can check every result by hand while you learn.