Modern Data Analytics for Forestry and Natural Resources

Efficient Coding, Data Analysis, and AI-Assisted Debugging

Dr. Suborna Ahmed and OER Team

2026-07-29

Table of contents

Preface

Edition: v2 build, 2026

Repository: https://github.com/subornaa/modern-computing-forestry
Cite as: Ahmed, S. (2026). *Modern Data Analytics for Forestry and Natural Resources.* UBC Faculty of Forestry & Environmental Stewardship. CC BY 4.0.

This open textbook teaches data analysis for forestry and natural resources using free, modern, openly licensed tools: Microsoft Excel (Chapters 1–3), R with the tidyverse (Chapters 4–9), and Quarto for reproducible reporting, an integrated case study, and presentations. Spatial analysis with the `sf` and `leaflet` packages is available as an optional extension. Every dataset used is openly licensed Canadian, BC, or Ontario government data. The book is designed to bridge from Excel to R for learners who have never written code before, while preparing them for advanced study in statistical modelling, GIS, and reproducible research.

It is the primary (free) text for **FRST 232 — Computer Applications in Forestry** and a pre-learning and refresher resource for **FRST 531**. The front-matter pages that follow — land acknowledgment, Indigenous data sovereignty, the accessibility statement, the EDI commitment, and *How to use this book* — are part of the resource, not boilerplate; please read them before the chapters.

See the **Authors** and **Acknowledgements** pages for the full project team and funding acknowledgement, and the **Datasets Used in This Book** page for every dataset and its download link.

Errata and contributing

This book is openly licensed. We welcome:

- **Bug reports** — open an issue on the GitHub repository.
- **Accessibility concerns** — see the *Accessibility statement*; we treat these as high priority.
- **EDI feedback** — see the *EDI commitment*; we welcome correction.
- **Pull requests with corrections, improvements, or additions** — under the same CC BY 4.0 (content) / MIT (code) licences.

The canonical version lives at https://github.com/subornaa/modern-computing-forestry.

Licence

Part I

Front matter

Authors

Suborna Ahmed

Dr. Suborna Ahmed is an Associate Professor of Teaching in the Department of Forest Resources Management, Faculty of Forestry & Environmental Stewardship, at the University of British Columbia. She teaches data analytics, computing, forest biometrics, and quantitative methods for forestry and natural-resources students.

Her teaching focuses on helping students move from spreadsheet-based data analysis to reproducible workflows in R, Quarto, and modern open-source tools. She develops open educational resources that use real forestry and environmental datasets, scaffolded practice, active learning, and AI-assisted debugging to support students with different levels of coding experience.

Dr. Ahmed's scholarly and teaching interests include forestry data analytics, forest biometrics, reproducible research, open education, inclusive quantitative pedagogy, and the responsible use of generative AI in technical learning. She is the project lead for this OER and has developed it with support from student contributors and collaborators to make data-analysis training more accessible, practical, and relevant for forestry and natural-resources education.

OER Team

This resource was developed with the OER Team:

Table 1: The OER Team

Contributor	Affiliation
Ehsan Karim	Associate Professor, School of Population and Public Health, Faculty of Medicine, UBC
Jiaxin Wen	BSc, Faculty of Arts and Economics; Statistics, UBC
Peace Damasco Uychiaoco	BSc, Faculty of Forestry & Environmental Stewardship, NRES - Conservation, UBC
Bennett Wardman	Department of Forest and Conservation Sciences, Faculty of Forestry & Environmental Stewardship, UBC
Rachel Lim	Forest Resources Management Department, Faculty of Forestry & Environmental Stewardship, UBC

Acknowledgements

Project team

This open educational resource was created by the FRST 232 OER project team, supported by the **UBC OER Fund Affordability Grant**.

Table 2: FRST 232 OER project team

Role	Team member	Affiliation
Principal Applicant	Dr. Suborna Ahmed	Faculty of Forestry & Environmental Stewardship, UBC
Co-Applicant	Dr. Ehsan Karim	Associate Professor, School of Population and Public Health, Faculty of Medicine, UBC
Team Member	Sophie Huang	Undergraduate Student, Bachelor of Arts in Psychology, Faculty of Arts, UBC
Team Member	Rebecca McCracken	Undergraduate Student, Bachelor of Science, Faculty of Forestry & Environmental Stewardship, UBC
Team Member	Rachel Lim	Graduate Student, Forest Resources Management Department, Faculty of Forestry & Environmental Stewardship, UBC
Team Member	Libby Taylor-Manning	Graduate Student, Forest Resources Management Department, Faculty of Forestry & Environmental Stewardship, UBC
Team Member	Michelle Zeng	Associate Director, Teaching and Learning Support, Dean's Office, Faculty of Forestry & Environmental Stewardship, UBC
Team Member	Dr. Stella Acquah	Senior Research Scientist, CSIR-Forestry Research Institute of Ghana; Adjunct Professor, Faculty of Forestry & Environmental Stewardship, UBC

Funding and support

This OER project was supported by the **Dean's Office of the Faculty of Forestry & Environmental Stewardship**. I gratefully acknowledge the support of **Dr. Robert Kozak**, **Dr. Sarah Gergel**, and **Jill Yu**. I am especially grateful to Dr. Sarah Gergel for her enormous encouragement and valuable suggestions, which helped move forward the modernization of FRST 232 and the development of this open educational resource.

Open data and open source

We acknowledge **Environmental Reporting BC**, the **Government of British Columbia**, the **Government of Ontario**, **Environment and Climate Change Canada**, **Natural Resources Canada**, and the **Columbia Shuswap Regional District** for publishing the forestry and natural-resources datasets used throughout this book under open government licences — making genuine learning with real data possible. (See the Datasets Used in This Book page for full sources and download links.)

We also thank the open-source community — the authors and maintainers of R, RStudio, Quarto, the tidyverse, `sf`, `leaflet`, and the many supporting packages — whose work makes this resource possible.

Learners and reviewers

We thank the FRST 232 learners (and the coming FRST 531 cohort) whose feedback shaped this resource, and the reviewers whose comments improved the curriculum and the open datasets behind it.

Land acknowledgment

This open educational resource was developed at the University of British Columbia, whose Vancouver campus is located on the traditional, ancestral, and unceded territory of the x mə k əyəm (Musqueam) people. UBC's Okanagan campus is located on the traditional, ancestral, and unceded territory of the Syilx Okanagan Nation.

The forestry datasets used throughout this book describe lands that are the traditional territories of many First Nations across British Columbia and Ontario — including the Coast Salish peoples (Sḵwx̱wú7mesh / Squamish, Səlílwətaɬ / Tsleil-Waututh, x mə k əyəm / Musqueam), the Stó:lō Nation, the Syilx, the Tsilhqot'in, the Anishinaabe, the Haudenosaunee, and many others not named here.

Indigenous peoples have practiced sophisticated, place-based forest stewardship across these lands for thousands of years before, during, and after the introduction of Western quantitative methods. The tools taught in this book — Excel spreadsheets, statistical summaries, ggplot2 charts, sf geometries, leaflet maps — are *one* way of describing and reasoning about forest ecosystems. **They are not the only way.**

We acknowledge that data about Indigenous lands has historically been collected and used in ways that have not always respected Indigenous data sovereignty, and that the responsibility to work in good relationship with First Nations rests with every practitioner using these tools.

Indigenous data sovereignty

Throughout this book, learners will work with publicly released government data from the BC Data Catalogue, Environmental Reporting BC, and the Government of Ontario. **Government open data is not the same thing as Indigenous data.**

When working with data that pertains to First Nations lands, peoples, knowledge, or resources, learners and instructors are expected to be familiar with two frameworks that govern respectful and rightful data practice:

OCAP® — First Nations principles

The **First Nations Information Governance Centre (FNIGC)** maintains the OCAP® framework, which asserts First Nations' sovereignty over data about their peoples, communities, and territories. The four principles are:

- **Ownership** — communities collectively own information about themselves the way an individual owns personal information.
- **Control** — First Nations have the right to control how information about them is collected, used, and disclosed.
- **Access** — First Nations must have access to information and data about themselves regardless of where it is held.
- **Possession** — physical control of data is the mechanism by which ownership and control can be asserted.

Learn more: https://fnigc.ca/ocap-training/

CARE — global Indigenous data principles

The **Global Indigenous Data Alliance** maintains the CARE principles, which complement the well-known FAIR (Findable, Accessible, Interoperable, Reusable) principles by adding an explicit ethical and relational layer:

- **Collective benefit** — data ecosystems must support Indigenous peoples' self-determination and collective benefit.
- **Authority to control** — Indigenous peoples' rights and interests must be recognized and their authority over data respected.
- **Responsibility** — those working with Indigenous data have a responsibility to share how that data is used to support self-determination and collective benefit.
- **Ethics** — Indigenous peoples' rights and well-being must be the primary concern at all stages of the data lifecycle.

Learn more: https://www.gida-global.org/care

Practical implications for this course

- **The data used in this book is government-published open data.** It is appropriate for teaching analytical methods.
- **It is not a substitute for working with First Nations directly.** Any forestry analysis that informs decisions affecting First Nations lands must be conducted in partnership with the affected Nation(s), and must respect OCAP and CARE principles.
- **The BC First Nations Forestry Council** (https://www.bcfnfc.ca/) is a primary point of reference for forestry-specific Indigenous data and management practice in BC.
- **A complete forestry analysis is more than statistics.** Traditional Ecological Knowledge (TEK), oral history, place-based observation, and lived experience are valid and necessary forms of forest knowledge that operate in partnership with — not in opposition to — quantitative methods.

Accessibility statement

This book is designed to **align with** the **Web Content Accessibility Guidelines (WCAG) 2.1, Level AA**, and with the **BC Accessibility Act (2021)** and UBC's Disability Accommodation policies. The **rendered HTML** has been checked directly against the criteria below; the **PDF** has known limitations (see *Verification and known limitations*). This is an alignment goal, not a certified conformance claim. Where we fall short, we welcome correction — please open an issue on the GitHub repository or email the author.

What we have done

- **Semantic HTML** with proper heading hierarchy (h1 → h2 → h3) for screen reader navigation.
- **Text labels alongside colour.** Quiz answers and status badges include / symbols and text in addition to colour. Most charts carry meaning in position, height, and labelled axes rather than colour; the few multi-series line charts that distinguish series mainly by colour also provide a labelled legend and descriptive alt text, and we are adding non-colour cues (such as line type) so no information depends on colour alone.
- **Keyboard accessibility** — every interactive element (quiz options, expand/collapse callouts, navigation) is reachable and operable via keyboard alone.
- **Alt-text on charts** — every R-generated chart includes a `fig-alt` description that conveys the chart's content to screen reader users.
- **High-contrast colour palette** — foreground/background combinations are chosen to align with WCAG AA contrast targets (4.5:1 for normal text, 3:1 for large text).
- **Embedded-resources HTML** — every rendered chapter is a single self-contained file that works offline, on slow connections, and with browser extensions for screen magnification or text-to-speech.
- **Plain-language headings** — chapter and section titles describe what the section does, not abstract jargon.
- **Free tools throughout** — no paid software required.
- **Multiple output formats** — chapter sources render to HTML and PDF so learners can choose the format that works best for their assistive technology. The HTML is the most accessible output; see *Verification and known limitations*.
- **Lab session flexibility** — every chapter's lab activity is structured to allow individual or group completion at the learner's pace.

What we ask of instructors

- **Confirm accommodations with the UBC Centre for Accessibility** (or your institution's equivalent) ahead of any timed exam or in-class presentation. Approved accommodations may include extended time, alternative formats, scribes, rest breaks, captioning, or remote participation.
- **Where the timed final exam creates equity concerns**, consider an alternative assessment format (e.g., a take-home practical exam, or an oral exam) for the *entire cohort*, not just learners with formal accommodations. Universal design benefits everyone.
- **Use the book's accompanying lecture material with captions and transcripts.** Whenever you record a lecture, caption it.
- **If a learner reports an accessibility barrier in this book, please report it on the GitHub repository.** Your report will lead to a fix in the next revision.

Verification and known limitations

What has been checked. The rendered **HTML** was audited directly: all 16 displayed charts carry a meaningful `fig-alt` description (with units such as hectares, parts per billion, and fiscal year), the book cover now has descriptive alt text, and the self-assessment quizzes include text and symbol labels plus a PDF-friendly fallback note. Contrast and colour-only checks are ongoing rather than certified.

The list below is honest about what still needs work. We are working on each item.

- **Figure alt text in the PDF.** The `fig-alt` descriptions are screen-reader accessible in the **HTML** edition, but the **PDF** does not expose them to assistive technology (a standard LaTeX PDF is not tagged). In the PDF each figure shows a visible in-image title; we are adding visible captions so the description is available in print as well. There is currently **no Word output**; request one if needed and we will generate a version with image alt text.
- **A few multi-series line charts** distinguish series mainly by colour; we are adding non-colour cues (such as line type) so they do not rely on colour alone.
- **Math notation** for statistical formulas (e.g., $\mu g/m^3$) uses Unicode characters; screen readers handle these inconsistently. If a learner's screen reader does not announce $\mu g/m^3$ correctly, the spelled-out *"micrograms per cubic metre"* is provided in parentheses on first use in each chapter.
- **Interactive leaflet maps** (in the optional spatial-data resource) depend on JavaScript and may not be fully usable by screen-reader users; a static `geom_sf` version of every interactive map is provided there as an alternative.
- **Spatial buffer and intersection results** are visual by nature; we describe what the result *shows* in prose immediately after each map.
- **Live R code execution** during the lab session requires fine motor control; learners who use voice input or alternative keyboards should request accommodation for practical assessments.

Privacy and analytics

This book is a free, open educational resource. We use a small amount of website analytics to understand which chapters are actually used, so we can improve the book and report on its reach as an OER. This page explains exactly what that means.

What we collect

The site uses **Google Analytics 4**. When you accept analytics cookies, it records:

- **which pages you visit** and how long you stay;
- **which files you download** (for example the datasets and the PDF edition);
- your **approximate location** (country or region — not your street address);
- the **site or search engine that referred you**, if any;
- your **browser and device type** (e.g., "Chrome on Windows").

What we do not collect

- **No names, emails, or student records.** Nothing in this book asks who you are, and analytics is not connected to Canvas or to any UBC system.
- **No IP addresses are stored.** Google Analytics 4 does not log or retain IP addresses; location is derived and then discarded.
- **No quiz answers.** The self-assessment quizzes run entirely in your browser. Your answers are never sent anywhere.
- **No advertising.** We do not run ads, and analytics data is not used for ad targeting.

Your choice

When you first visit, a banner asks whether to allow analytics cookies.

- **Decline, and no analytics cookies are set and nothing about your visit is recorded** — no page views, no downloads, nothing. The entire book — every chapter, dataset, and quiz — works exactly the same either way. No content is gated behind consent. (To be precise: your browser still fetches Google's analytics script file, as it would any script on a page, but if you decline it never runs and no measurement takes place.)
- You can also block analytics with a browser extension, or with Google's own opt-out add-on. We consider this entirely reasonable.

Where the data goes, and for how long

Analytics data is processed by **Google**, which stores it on servers **outside Canada** (primarily the United States). It is retained for **14 months** and then deleted automatically. Only the author has access to the reports, which show **aggregate totals** (for example, "Chapter 5 was viewed 240 times last month") — never individual readers.

Questions or concerns

If you would rather this book collected nothing at all, or you have a privacy question, please open an issue on the GitHub repository or email the author. We take the point seriously — analytics here exists to improve a teaching resource, nothing more.

Equity, diversity, and inclusion commitment

This book is open in two senses: **openly licensed** (anyone can reuse and adapt it), and **open to its audience** (no learner should feel that this book is not for them).

What "open" means in practice

Dimension	Our commitment
Cost	The book and every tool it teaches (R, RStudio, Quarto, Posit Cloud) are free of charge. No paywalls, no required textbook purchase.
Hardware	Posit Cloud (free tier) lets a learner complete every chapter from a smartphone, tablet, or library computer. No need for a personal high-end laptop.
Prior experience	The book assumes no prior coding background. Chapters 1–3 use Excel — a tool most learners already have access to — as the bridge into computational thinking.
Language background	A glossary is provided. Technical terms are defined on first use. Where English-language conventions diverge from common usage, we say so.
Learning differences	Each chapter offers multiple ways through: read, watch the screencast (where available), do the exercises, work the group lab, or take the quiz. There is no single required path.
Career direction	Chapters reference careers in research, government forestry agencies, First Nations forestry councils, consulting, NGOs, and field crews — not just academic paths.

Whose voices are in this book

Forestry analytics has historically been a field dominated by certain demographics. This book is committed to:

- **Citing and referencing the work of women, BIPOC, and Indigenous forestry researchers and practitioners.** Where classic R / tidyverse documentation is cited, we additionally cite ecological and forestry researchers from diverse backgrounds whose substantive work informs our examples.
- **Featuring diverse data sources.** The course uses Environmental Reporting BC, the Government of Ontario, and references the BC First Nations Forestry Council.
- **Avoiding default examples that exclude.** We do not centre Christian-calendar holidays, sports-team affiliation examples, or career arcs that assume a specific cultural background.

Inclusive language

This book follows the following conventions:

- **"Learner"** rather than "student" — inclusive of returning adults, mature learners, career-changers, and informal learners.
- **"Chapter"** rather than "week" — does not assume a fixed pacing.
- **"Course site"** rather than a specific LMS name — portable across institutional contexts.
- **Avoidance of ableist language** — we work to remove phrases like *"easy"*, *"obvious"*, *"just"*, *"see"*, or *"simply"* when they imply that a concept should be effortless. Learning takes work; that is normal.
- **Gender-neutral defaults** — we use "everyone", "the learner", or "they" instead of "guys" or default-masculine language.
- **Plain language alongside technical terms** — *"the tibble (a kind of data frame)"* rather than introducing jargon without translation.

Multiple ways of knowing

The quantitative methods in this book — descriptive statistics, group summaries, regression smoothers, spatial overlays — are *one* way of understanding forests. They are powerful, and they are necessary for many of the questions a working forester or researcher will ask. **They are not sufficient on their own.**

Traditional Ecological Knowledge (TEK), oral history, participatory observation, place-based experience, and First Nations forestry practices offer complementary forms of knowledge that are equally rigorous, equally evidence-based, and in many cases more deeply informed about the specific forest systems being analysed. A complete forestry analysis brings these forms of knowledge into partnership, not into hierarchy.

For deeper engagement with this idea, see:

- Atleo, C. (2011). *Tsawalk: A Nuu-chah-nulth Worldview*. UBC Press.
- Kimmerer, R. W. (2013). *Braiding Sweetgrass: Indigenous Wisdom, Scientific Knowledge, and the Teachings of Plants*. Milkweed Editions.
- BC First Nations Forestry Council resources at https://www.bcfnfc.ca/.
- Mills, A. & McGregor, D. (2022). *Indigenous Data Sovereignty in Canada*. (Various publications.)

How to use this book

The chapter pattern

Every chapter follows the same structure:

1. **Chapter goals** — what you will be able to do by the end.
2. **Expected output callout** — the concrete deliverable for the chapter assignment.
3. **Where we are** — how this chapter connects to the previous one.
4. **Excel-to-R bridge** (from Chapter 4 onward) — every Excel pattern paired with its R equivalent.
5. **Core content** — concepts and demonstrations.
6. **Active learning** — 5–6 short activities to try immediately.
7. **Group lab activity** — a single longer task designed for a lab session, often in groups of 3–4.
8. **Exercises** — 6–8 take-home exercises.
9. **Optional, advanced** — material for learners who want to go deeper.
10. **Glimpse of the next chapter** — a forward-looking preview.
11. **Self-assessment quiz** — 20 questions with instant feedback.
12. **AI as a debugging companion** — when to use AI, how to prompt it, and when not to.
13. **Reading** — references, including a mix of canonical and diverse sources.
14. **Instructor notes** (collapsed callout) — pacing, grading rubric, common mistakes.

Materials you will receive per chapter

File	Purpose
Chapter HTML (rendered from `.qmd`)	The chapter itself — read, follow examples
Learner workbook (`frst232_chNN_learner.qmd`)	A Quarto document with TODO chunks for the exercises
Solutions (`frst232_chNN_solutions.qmd`)	Completed answers — **for instructor use only** until released
Standalone quiz (`quiz_chNN.html`)	Self-paced practice quiz
Raw data (where applicable)	The forestry dataset for the chapter
Starter zip (Chapters 4 and 5)	Pre-built RStudio Project for first-time R users

How to get help

In order of who to ask first:

1. **Re-read the relevant chapter section.** The chapter has the answer in 70% of cases.
2. **Search the chapter's *Instructor notes* callout** for "Common stumbling points".
3. **Read the error message carefully** — R error messages are often more helpful than they look on first read.
4. **Ask a classmate** — a peer who solved the same problem 30 minutes ago is often the fastest help.
5. **Office hours, course site discussion forum, or email the TA.**
6. **AI assistants** — use them as debugging companions, not answer machines. The chapter has guidance on how to prompt effectively (and when NOT to use AI).

How to write a useful AI prompt

When you do reach for an AI assistant, a good prompt gets a useful answer. Use this formula:

> **Context + dataset + task + what I tried + error/output + what I need help with**

Worked example:

> *"I am using R in FRST 232. I have a tibble called* `bc` *from* `bc_disturbance_reforestation.xlsx`*. I want to calculate the total reforested area by decade. I tried this code:* `[paste your code]`*. I got this error/output:* `[paste the exact error/output]`*. Please explain what went wrong and suggest a fix. Do not skip steps, and tell me how to verify the answer."*

Every time you ask AI for help:

- Paste the **exact** error message — not a paraphrase.
- Include the **dataset and column names** you are working with.
- Ask the AI to **explain the code line by line**, not just hand you code.
- Ask it for **verification steps** you can run yourself.
- When you are unsure, **compare answers across two AI tools**.
- Check every suggestion against the **book, the package documentation, and your actual data**.
- **Never submit AI output you do not understand and have not verified.**

The goal is not to make AI do the work. The goal is to use AI to help you **understand, check, and improve your own work.**

Recommended pacing

The book is designed for a one-term course (twelve numbered chapters of content; the last is final-exam preparation). A reasonable pace is **one chapter per week**, with the lab session for each chapter being the in-class hands-on time.

- **Chapters 1–3** (Excel): 3 weeks
- **Chapters 4–5** (R foundations, Quarto): 2 weeks
- **Chapters 6–8** (dplyr, summarising, joining): 3 weeks

- **Chapter 9** (ggplot2): 1 week
- **Chapter 10** (integrated case study): 1 week
- **Optional:** spatial analysis (`sf`, `leaflet`) — extension material
- **Chapter 11** (presentations + portfolio): 1 week
- **Chapter 12** (review + final-exam preparation): 1 week

For FRST 531 (graduate students), the same material can be compressed into 10 weeks with Chapters 1–3 (Excel) treated as a one-week refresher.

A note on the lab sessions

Most chapters have a **Practice Demo Lab** — a guided practice activity in this book. The **official graded lab assignments are posted on Canvas**; the book demos help you learn the workflow first. A few weeks have special lab sessions in the current schedule:

- **Week 5** = guided integrated case-study / Excel-to-R transition lab.
- **Week 6** = Excel midterm (covers the Excel chapters), held in the lab session.
- **Week 13** = independent integrated case-study lab and project work session.
- **Week 14** = final-exam preparation; no lab.

Datasets Used in This Book

Every dataset in this book is **open government data** — free to download, no account required. This page collects them all in one place with download links so you can get the data before you start a chapter. Each data-using chapter also links back here.

Where the data goes

Put each downloaded file in your project's `data/` folder, **unchanged** from the source. Read it with `here("data", "<filename>")` in R, or open it directly in Excel. Keep the raw file; save any cleaned version separately in `outputs/`.

Download directly from this book

Click a **download link** (the file name) in the table below to get the **exact file used in the chapters**, served straight from this book — no GitHub or account needed. Beside each is a **source** link to the official publisher for provenance. Some files are *teaching extracts* prepared from the larger published dataset, so the source may hold more data than the file used here. **No links on this page are invented.**

All datasets at a glance

Table 3: Open datasets used across the book — click a file name to download it directly

Dataset	Source (licence)	Download · source	Type	Used in
BC silviculture — *Trends in Silviculture in B.C.*	Environmental Reporting BC / BC Ministry of Forests (OGL — BC)	bc_disturbance_reforestation.xlsx · source	.xlsx	Ch 1, 3, 4, 5, 8, 9, 12, 13
Ontario forest statistics — *Forest Resources of Ontario 2021*	Government of Ontario (OGL — Ontario)	on_forest_statistics_1.xlsx · source	.xlsx	Ch 2, 3, 4, 5, 8
BC air quality — *Verified Hourly Air Quality & Meteorological Data*	BC Ministry of Environment (OGL — BC)	airdata3.csv · source	.csv	Ch 5, 6, 7, 8, 9, 11

Dataset	Source (licence)	Download · source	Type	Used in
ECCC weather — daily climate, Vancouver Int'l A (Climate ID 1108447)	Environment and Climate Change Canada	`inter_air_van_weather.csv` · source	`.csv`	Ch 8
NRCan wildfire — National Fire Database, fire points	Natural Resources Canada / CWFIS	`nfdb_fires_2021_2024.csv` · source	`.csv`	Ch 4; Ch 9 (derived)
CSRD parks — Park Locations	Columbia Shuswap Regional District Open Data	`csrd_parks.geojson` · source	`.geojson`	*Optional spatial resource*

Dataset details

BC silviculture — `bc_disturbance_reforestation.xlsx`

- **Source:** Environmental Reporting BC, *Trends in Silviculture in B.C.* (BC Ministry of Forests). Open Government Licence — British Columbia.
- **How it's used:** the running example for the Excel chapters (1–3) and the first reproducible R import (Ch 4–5); annual harvested, natural-disturbance, and reforested area by fiscal year, 1987–2023.
- **File:** ~37 rows × 8 columns, three sheets (*About* / *ReadMe* / *Data*). Read the **Data** sheet. Small and clean — no preparation needed.

Ontario forest statistics — `on_forest_statistics_1.xlsx`

- **Source:** Government of Ontario, *Forest Resources of Ontario 2021.* Open Government Licence — Ontario.
- **How it's used:** the messier, categorical file for Excel cleaning, PivotTables, and grouped summaries; a cross-classification of area by ownership × seral stage × forest type × age class.
- **File:** the book uses a **Region 3E teaching extract** (~179 rows × 8 columns) of the larger published file.

BC air quality — `airdata3.csv`

- **Source:** BC Ministry of Environment, *Air Quality and Climate Monitoring: Verified Hourly Air Quality and Meteorological Data.* Open Government Licence — British Columbia.
- **How it's used:** the real, messy file for cleaning (Ch 6), grouped summaries (Ch 7), the air + weather join (Ch 8), and `ggplot2` (Ch 9). The book focuses on **ground-level ozone (O_3)**, which is recorded at eight Lower Mainland stations.

- **File:** a **teaching extract** prepared from the source (selected Lower Mainland stations, hourly, January 2000), ~11,900 rows × 26 columns, with realistic missingness across pollutants.

ECCC weather — `inter_air_van_weather.csv`

- **Source:** Environment and Climate Change Canada, Historical Climate Data — Vancouver International Airport (Climate ID 1108447; web station ID 51442).
- **How it's used:** joined to the hourly air-quality data in Ch 8 to demonstrate a safe **many-to-one** join (hourly readings → daily weather).
- **File:** daily climate for the year 2000 (~366 rows × 31 columns). The *Direct CSV* link above is the ECCC bulk-download endpoint; the script `scripts/download_eccc_yvr_climate.R` regenerates it.

NRCan wildfire — `nfdb_fires_2021_2024.csv`

- **Source:** Natural Resources Canada, Canadian Wildland Fire Information System — Canadian National Fire Database (NFDB), fire point data.
- **How it's used:** a first-import practice file (Ch 4); in Ch 9 it is aggregated to an annual area-burned summary by `scripts/build_wildfire_summary.R`.
- **File:** the book uses a **2021–2024 subset** of the national point dataset (the source download is a shapefile covering 1950–present; convert/subset as needed).

CSRD parks — `csrd_parks.geojson`

- **Source:** Columbia Shuswap Regional District Open Data (ArcGIS Hub), *Park Locations* feature layer.
- **How it's used:** the real spatial layer for the **optional spatial extension** (`sf`: inspect CRS, reproject to BC Albers, static `geom_sf` map; interactive `leaflet` map). Spatial analysis is no longer part of the required book — see *Optional: Spatial Data Analysis in R*.
- **File:** 66 point features in WGS 84 (EPSG:4326). The *Direct GeoJSON* link is a live query; `scripts/download_csrd_parks.R` regenerates it.

Reproducible download scripts

Three small R scripts in `scripts/` let you regenerate the data files from source:

- `download_eccc_yvr_climate.R` → `inter_air_van_weather.csv`
- `download_csrd_parks.R` → `csrd_parks.geojson`
- `build_wildfire_summary.R` → annual wildfire summary (from the NFDB points)

Part II

Excel foundations

1 Excel as a Data-Analysis Workbench

Data for this chapter

This chapter uses the **BC silviculture** workbook (`bc_disturbance_reforestation.xlsx`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Open a real BC Government Excel workbook and orient yourself to its contents on first read.
- Read the **About**, **ReadMe**, and **Data** sheets in the right order, and explain why that order matters.
- Identify each column's **type** (text, number, date) and **unit** (hectares, year).
- Use **freeze panes** and a few keyboard shortcuts to move through a multi-column file without losing context.
- Apply **basic number formats** that distinguish display from value.
- Write a few **simple formulas** (`SUM`, `AVERAGE`, `MIN`, `MAX`) and **check whether the results make sense**.
- Set up a clean **project folder structure** that separates raw data from cleaned files.

Expected output for this chapter

At the end of this chapter, each learner (or group) will produce:

What you will hand in

1. The raw workbook **`bc_disturbance_reforestation.xlsx`** placed in your `data/` folder, **unchanged from the BC Data Catalogue download**.
2. A working copy you may modify, saved as **`bc_disturbance_reforestation_clean.xlsx`** in your `outputs/` folder, containing:
    - the original three sheets (**About**, **ReadMe**, **Data**) kept intact — you add to them, you do **not** create new sheets or delete the originals,
    - one short **data dictionary entry** in your own words for any column you choose, and

- the five **simple formulas** from this chapter computed in any empty cell.

3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

1.1 Why we start with Excel

Excel has flaws. It hides decisions in invisible mouse clicks. It is hard to reproduce or recreate. It is hard to version-control. We will spend most of this book moving away from it.

We still start with Excel deliberately. The interface is forgiving. Mistakes are visible. There is no console to fight. And the fundamental discipline we want learners to leave the course with — *inspect first, compute second, verify always* — is easier to teach when the data is sitting on the screen in plain sight.

When we move to R in Chapter 4, we will use **the same dataset** we analyse in Excel here. The continuity is the point: R is not a fresh start; it is a more honest version of what you already do. (See *A glimpse of Chapter 4* at the end of this chapter for a preview.)

1.1.1 What "data analysis" actually means

People say *"data analysis"* to mean different things. In this course it means a sequence of five repeatable steps:

1. **Inspect** the file. Know what is in it before computing anything.
2. **Document** the file. Write a data dictionary you can defend.
3. **Clean** the file. Remove what is wrong, document what was removed.
4. **Summarize** the file. Compute the numbers that answer your question.
5. **Communicate** the result. Tell the story with a chart, a table, or a sentence.

Chapter 1 covers steps 1 and 2 — *inspect* and *document*. The rest of the book builds the other three steps on top of that foundation.

1.2 The dataset for this chapter

This chapter uses one real, openly licensed Excel workbook from the **BC Government / BC Data Catalogue**:

Property	Value
File name	`bc_disturbance_reforestation.xlsx`
Source	Environmental Reporting BC — *Trends in Silviculture in B.C.*
Catalogue page	https://catalogue.data.gov.bc.ca/dataset/indicator-summary-data-trends-in-silviculture-in-b-c-
Licence	Open Government Licence — British Columbia

Property	Value
Rows of data	37
Columns	8
Time span	1987–2023 (fiscal years)
Spatial coverage	Province of British Columbia
Unit	Hectares (ha)

The file is **directly downloadable** from the BC Data Catalogue. You do not need an account. There is no shapefile bundle to manage. It is one Excel file with three sheets.

1.2.1 Why this dataset matters for forestry

This is not a textbook dataset. Every number was published by the BC Ministry of Forests under the **Trends in Silviculture in B.C.** program. It is the same indicator that the province uses to track whether reforestation is keeping pace with the area being harvested and lost to natural disturbance.

A working forester or natural-resources analyst in BC will encounter this dataset (or one with the same structure) within a few months of starting any monitoring or reporting role. Learning to read it now is real, applicable work.

The dataset captures the story of BC silviculture from 1987 onward:

- The shift in harvest practice from straight clearcutting toward **clearcutting-with-reserves** through the 1990s — a real policy change visible in the columns.
- BC's record-breaking **2017 wildfire season**, which dominates the natural-disturbance column.
- Steady growth in **reforestation** that, by the 2020s, exceeds total disturbance in most years.

! Where the data comes from — and what to do with it

This dataset is published by the BC Government and distributed through the **BC Data Catalogue**. Two rules:

1. **Keep the raw downloaded file unchanged.** Save the file as `bc_disturbance_reforestation.xlsx` exactly as you downloaded it. Do not sort it, edit it, or save over it. The raw file is your point of reference if anything later goes wrong.
2. **Save any cleaned or modified version separately.** Use a different file name (typically `bc_disturbance_reforestation_clean.xlsx`) and a different folder. That way you can always go back to the original.

This is the single most important habit in this chapter.

Why this dataset is right for this chapter

- **It is real.** Every number was published by the BC Ministry of Forests. You can verify any value against the official indicator page.
- **It is small.** 37 rows × 8 columns. You can scroll through every row and inspect every cell.
- **It has no missing values.** Cleaning comes later in the book.
- **The columns mean something.** Every column has a forestry meaning we can talk about.
- **It tells a story** — see *Why this dataset matters for forestry* above.

1.2.2 What the dataset measures

Each row is one **fiscal year** in BC. Each column is one **area-based forestry indicator** — harvested area, naturally disturbed area, reforested area. Units are always hectares (ha).

A fiscal year in BC runs from **April 1 to March 31** of the following calendar year. So "Fiscal Year 2023" means April 2023 through March 2024.

1.3 Project folder structure

Before you open the file, set up where it will live. The way you organize files now saves a hundred *"where is that file?"* moments later.

Create this folder on your computer (or on Posit Cloud):

```
frst232/
  chapter01/
    data/                                  Raw data files - never edit anything in here.
      bc_disturbance_reforestation.xlsx
    outputs/                               Cleaned or modified files you produce.
      bc_disturbance_reforestation_clean.xlsx   (you save this yourself)
    notes/                                 Lecture notes, screenshots.
    deliverables/                          What you submit to the course site.
  shared/
    data_dictionaries/                     ReadMe sheets you have written.
    references/                            Course PDFs, papers, links.
```

Three rules:

1. **Never edit raw data in place.** Always work on a copy that lives in `outputs/`.
2. **Use lowercase, snake_case names.** `bc_disturbance_reforestation.xlsx` — not `BC Disturbance Data (Final).xlsx`. Spaces and capitals cause problems when you move to R in Chapter 4.
3. **One folder per chapter.** Easier to find things in a dozen small folders than in one huge dump.

Where the file goes in this course

For every code chunk in this book to work, the data file should live at:

`your_project_root/data/bc_disturbance_reforestation.xlsx`

When you see `here("data", "bc_disturbance_reforestation.xlsx")` in a later chapter, that is the path it refers to.

1.4 The three-sheet pattern: About, ReadMe, Data

Download `bc_disturbance_reforestation.xlsx` from the BC Data Catalogue and place it in your `data/` folder. Open the file. You will see three tabs at the bottom:

- **About** — *what is this file?* Source, publisher, licence, citation, retrieval date.
- **ReadMe** — *what does each column mean?* One row per variable, with description, unit, and notes.
- **Data** — the actual 37 rows of data.

Read these tabs in order. Always. Reading the Data sheet before reading About means computing something before you know what it measures.

1.4.1 About sheet

The About sheet answers *where did this file come from?*. Typical fields:

Field	Example value
Dataset	*Indicator Summary Data: Trends in Silviculture in B.C.*
Publisher	Resource Practices Branch, Ministry of Forests, B.C.
Source URL	catalogue.data.gov.bc.ca/…
Licence	Open Government Licence — British Columbia
Cite as	Environmental Reporting BC, *Trends in Silviculture in B.C.*
Retrieved	2026-05-24

The point of the About sheet is **provenance**. A reader who finds this file two years from now needs to know where it came from and under what licence they can use it.

1.4.2 ReadMe sheet

The ReadMe sheet answers *what does each column mean?*:

Variable	Description	Unit	Notes	NA
Fiscal_Year	Fiscal year of the row. April 1 to March 31 of the next calendar year.	year (integer)	number	Integer between 1987 and 2023.
Clearcutting_ha	Area harvested using the clearcutting silvicultural system.	hectares	number	Most of BC's harvest historically. Declines after the mid-1990s as clearcutting-with-reserves grows.
Clearcutting_with_reserves_ha	Area harvested using clearcutting with reserves — some trees retained for wildlife, visual, or hydrological values.	hectares	number	Near-zero in the late 1980s. Increases sharply through the 1990s and 2000s due to BC silvicultural policy changes.
Partial_cutting_ha	Area harvested using partial cutting silvicultural systems.	hectares	number	Smaller share of total harvest. Used where ecological or visual constraints favour retention.
Harvested_ha	Total area harvested = sum of the three harvest types above.	hectares	number	Should equal Clearcutting_ha + Clearcutting_with_reserves_ha + Partial_cutting_ha for every row.
Natural_Disturbance_ha	Area disturbed by natural causes (wildfire, pest, wind) tracked for reforestation.	hectares	number	Note the 2017 spike (218,275 ha) — BC's record-breaking wildfire season.
Total_Disturbance_ha	Sum of harvested area + natural disturbance area.	hectares	number	Should equal Harvested_ha + Natural_Disturbance_ha for every row.
Reforestation_ha	Area planted or naturally regenerated in that fiscal year.	hectares	number	Includes both human planting and natural regeneration.

One row per column. Notice how the unit (hectares) appears in every numeric row. Never assume the unit; check it.

> **A pattern to keep forever**
>
> Every dataset you create for this course should ship with About and ReadMe sheets like these. They cost an hour to write and save many hours of *"what does this column mean?"* arguments later.

1.4.3 Data sheet

The Data sheet holds the 37 rows of forestry indicators. We will inspect it next.

1.5 Inspecting the Data sheet

> **AI prompt to check your work**
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"I read the About, ReadMe, and Data sheets of `bc_disturbance_reforestation.xlsx` and concluded it has one row per fiscal year with area columns in hectares. Check whether my reading of the sheets and column meanings is complete and consistent, and tell me what a careful analyst might have missed on a first inspection. Do not summarise the file for me from scratch."*

Open the Data sheet. Before computing anything, answer these five questions:

1. **How many rows of data are there?** (Ignore the header.)
2. **How many columns?**
3. **What is the earliest and latest year?**
4. **Pick one column.** What does it measure, and in what units?
5. **Look at the first row.** Does anything look unexpected?

For this file:

Question	Answer
Rows of data	37
Columns	8
Earliest fiscal year	1987
Latest fiscal year	2023
Maximum Natural_Disturbance_ha (and its year)	218,275.3 ha in 2017

The 2017 spike in `Natural_Disturbance_ha` is BC's record-breaking wildfire season. The dataset would look like a flat trend without it. The 2017 row alone is a lesson in why you *always look at the extremes* of every column.

1.5.1 The eight columns, one by one

Column	Type	What it measures
`Fiscal_Year`	Integer year	The fiscal year the row describes (April–March, named by starting year).
`Clearcutting_ha`	Number (hectares)	Area harvested using the clearcutting silvicultural system.

Column	Type	What it measures
`Clearcutting_with_reserves_ha`	Number (hectares)	Area harvested using clearcutting **with reserves** — some trees left standing.
`Partial_cutting_ha`	Number (hectares)	Area harvested using partial cutting (shelterwood, selection, etc.).
`Harvested_ha`	Number (hectares)	Sum of the three harvest types above.
`Natural_Disturbance_ha`	Number (hectares)	Area disturbed by natural causes (fire, pest, wind).
`Total_Disturbance_ha`	Number (hectares)	Total harvested + naturally disturbed area.
`Reforestation_ha`	Number (hectares)	Area planted or naturally regenerated that year.

A small spot check

`Total_Disturbance_ha` is the sum of `Harvested_ha` and `Natural_Disturbance_ha`. Pick any one row — say row 38 (year 2023) — and check that the formula `=E38 + F38 = G38` evaluates to `TRUE`. If it does, the dataset is internally consistent.

1.6 Data types — and why they matter

Every cell in Excel has a **type**. The three you care about for this chapter:

Type	Example	Default alignment
Number	`216304.919`	right
Date	`2024-09-15`	right
Text	`Vancouver`	left

Excel guesses the type when you open a file. Most of the time it guesses right. But sometimes a number arrives stored as text (maybe because of an extra apostrophe or a stray space) and is then silently skipped when you `SUM` the column.

A quick test: select a numeric column and look at the status bar at the bottom of the Excel window. It usually shows `Sum`, `Average`, and `Count`. If `Count` matches your row count but `Sum` is suspiciously small, some cells are stored as text.

1.7 Navigation: freeze panes and a few shortcuts

A 37-row file does not need navigation tricks. A 37 000-row file does. Learn these now while the file is small.

1.7.1 Freeze panes

Click row 2 (the row below the header). Then choose **View → Freeze Panes → Freeze Top Row**. The header row stays visible while you scroll.

For wider files, select cell B2 and choose **View → Freeze Panes → Freeze Panes**. Both the first row and the first column stay visible.

1.7.2 A few keyboard shortcuts

Shortcut	What it does
`Ctrl + Arrow`	Jump to the edge of the data
`Ctrl + Home`	Jump to A1
`Ctrl + End`	Jump to the last cell of the used range
`Ctrl + F`	Find
`Ctrl + ;`	Insert today's date as a value

Learning a handful of shortcuts saves real time on bigger files.

1.8 Number formats: display vs value

A single cell carries two pieces of information:

- The **value** (what the cell contains).
- The **display format** (how the value is shown).

These are different. Changing the format never changes the value.

1.8.1 A worked example

Type `216304.919` into a cell. The cell displays `216304.919`.

Now right-click the cell → **Format Cells → Number**. Set decimal places to 0. The cell now displays `216,305`. The underlying value is still `216304.919`, just rounded for display.

Try **Format Cells → Currency** with two decimals. The cell shows `$216,304.92`. Same underlying value.

Try **Format Cells → Custom** with the code `0 "ha"`. The cell shows `216305 ha`. The string `"ha"` is part of the *format*, not the value — formulas referencing this cell still treat it as a plain number.

1.8.2 Why this matters

Two cells can display the same thing but have different values. If cell A holds `100000` and cell B holds `99999.999999`, both can be formatted to display `100,000`. But `=A1=B1` returns `FALSE`.

This is the single most common silent bug in Excel-based analysis. **Always be aware of the underlying value, not just the display.**

Two formats worth knowing

- `#,##0` — thousands separator, no decimals. Use for hectares.
- `yyyy-mm-dd` — ISO 8601 date format. Sorts correctly as text *and* as a date. Universally readable across regions.

1.9 Writing simple formulas

The five most-used aggregating formulas. Type each in any empty cell on the Data sheet.

Formula	What it does
`=SUM(H2:H38)`	Adds every value in column H (Reforestation_ha)
`=AVERAGE(G2:G38)`	Mean Total_Disturbance_ha across all rows
`=MIN(F2:F38)`	Smallest Natural_Disturbance_ha
`=MAX(F2:F38)`	Largest Natural_Disturbance_ha
`=COUNT(A2:A38)`	Number of rows with a numeric year

On this file:

Formula	Result
=SUM(H2:H38)	8,264,064 ha
=AVERAGE(G2:G38)	228,089 ha
=MIN(F2:F38)	1,448 ha
=MAX(F2:F38)	218,275 ha
=COUNT(A2:A38)	37

1.10 Checking whether values make sense

Before you trust any number you compute, run a spot check.

1.10.1 Three quick checks

1. **Range check.** A mean should sit between the min and the max. If `MIN = 50,000`, `MAX = 250,000`, and your `AVERAGE` says `400,000`, something is wrong.
2. **Identity check.** If the dataset claims `Total_Disturbance_ha = Harvested_ha + Natural_Disturbance_ha`, pick one row and verify the identity.
3. **Order-of-magnitude check.** BC reforests roughly 200 000 ha per year. If your computed mean comes out at 20,000 or 2,000,000, you have likely selected the wrong column or misformatted the result.

These three checks take seconds and catch most bugs.

> Read the value, not the display
>
> After you write a formula, click the result cell and look at the formula bar. If the formula says `=SUM(B2:B38)` but you wanted column H, you have selected the wrong column. Always read the formula, not just the answer.

1.11 Building a data dictionary

> AI prompt to check your work
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"Here is my data-dictionary entry for one column: `[paste your column name, type, unit, and plain-language description]`. Check whether my description matches the ReadMe, whether I named the correct type and unit, and what a reviewer might question. Do not rewrite the dictionary for me."*

A data dictionary is a separate sheet that lists every column with:

- **Variable** name (exactly as it appears in the data).
- **Description** in plain English — what does it measure?
- **Unit** of measurement, always.
- **Type** (number, text, date).
- **Notes** — anything a future reader should know.

The ReadMe sheet of `bc_disturbance_reforestation.xlsx` already has one. Look at it.

1.11.1 How to build your own

In any new workbook:

1. Add a new sheet, name it `ReadMe`.
2. In row 1, type the column headers: `Variable`, `Description`, `Unit`, `Type`, `Notes`.
3. In row 2 onward, add one row per column from your Data sheet.

4. Be **specific**. *"Reforestation, in hectares"* is okay. *"Area planted or naturally regenerated in that fiscal year, in hectares"* is better.

> ⚠ Test your dictionary
>
> After you write a dictionary, hand the workbook to someone who has never seen the data and ask them to compute one statistic from it without asking you any questions. If they can, your dictionary is good. If they ask what a column means, the dictionary needs work.

1.12 The three habits: inspect first, compute second, verify always

If you take only one thing from this chapter, take this.

1.12.1 Inspect first

Before you compute *any* number, look at the data. Open the file. Read the About sheet. Read the ReadMe. Scroll through a few rows. Find the extremes — the largest, the smallest, the oldest, the newest.

This takes a few minutes and catches more bugs than any other practice in data analysis.

1.12.2 Compute second

Only after you have inspected do you start typing formulas. You should be able to **predict** what your formula will return before you press Enter. If the answer surprises you, that surprise is information — either the data is interesting or the formula is wrong.

1.12.3 Verify always

Every time you compute a number, verify it against something:

- **Against a published total.** If the BC Ministry says the total reforestation in 2020 was X, your formula should equal X.
- **Against a spot check.** A mean cannot be larger than the max or smaller than the min.
- **Against a second method.** If you computed it with `SUM`, also compute it with a PivotTable (Chapter 3). If both agree, trust the number.

1.13 Active learning

These short activities are designed to be done alongside the chapter. Pick whichever match the pace of your session.

1.13.1 Activity 1 — Read the About sheet aloud

In pairs, one learner reads the About sheet, the other listens and asks one clarifying question. Switch.

The point is **slowing down**. Force yourself to read the metadata before the data.

1.13.2 Activity 2 — Five-question inspection

On your own, complete this inspection worksheet, then compare with a partner. If your answers differ, go back to the About, ReadMe, and Data sheets to resolve them.

Check	Your answer
Number of rows	
Number of columns	
Earliest and latest fiscal year	
One column name, its unit, and meaning	
One value or pattern that seems important or surprising	

1.13.3 Activity 3 — Write five simple formulas

In your **working copy** (not the raw download), type each formula from the *Writing simple formulas* section into an empty cell. Confirm each result matches the expected value in the table.

1.13.4 Activity 4 — Verify one identity

On any one row of the Data sheet, verify that `Harvested_ha + Natural_Disturbance_ha = Total_Disturbance_ha`. Use a formula like `=E2 + F2 - G2`. The result should be 0 (or very close — Excel sometimes shows tiny non-zero values for floating-point reasons).

1.13.5 Activity 5 — Build a one-row data dictionary

Pick any column from the Data sheet. Write a one-sentence description for it without looking at the ReadMe. Then compare with the ReadMe. How close did you get?

1.14 Practice Demo Lab

! Practice demo only

This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.
2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?
4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

i Lab: Inspect a real BC dataset together

Work in groups of 3 or 4. Use one shared copy of `bc_disturbance_reforestation.xlsx` (one machine per group, or share the screen).
Task 1 — Inspect the workbook. Open the file. Take turns reading the About sheet, the ReadMe sheet, and the Data sheet. Each group member should be able to answer:

- Where did this file come from?
- What does each column measure?
- How many rows are there?

Task 2 — Small data dictionary task. As a group, pick **two columns** of the Data sheet and write your own one-sentence description of each in your own words. Compare your descriptions to the ReadMe sheet's descriptions.
Task 3 — Compute one or two summary values. As a group, choose one of these:

- Mean `Reforestation_ha` across all 37 fiscal years.
- Maximum `Natural_Disturbance_ha` and the year it happened.
- Total `Harvested_ha` across all 37 fiscal years.

Write the formula on the Data sheet. Then sanity-check the result **with the check that fits your statistic**:

- **Mean** — must fall *between* the column's `MIN` and `MAX`.
- **Maximum** — must *equal* the single largest value in the column (and be every other row).

- **Total** — must be *at least as large as the* `MAX` (usually much larger), and never negative for an area column.

Task 4 — Verify one formula. Pick any row of the Data sheet. Verify that `Harvested_ha + Natural_Disturbance_ha = Total_Disturbance_ha`. Use a formula like `=E10 + F10 - G10`. The result should be 0.
Task 5 — Submit. Submit either a short worksheet (a one-page document with your group's answers to Tasks 1–4) **or** a screenshot of the Data sheet showing all of your group's formulas. Include all group members' names on the submission.
This lab is designed to fit comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 1 reference solution (HTML answer key) — the completed, task-by-task answer key — or **download the completed Excel workbook** with the formulas worked into the sheets (it recalculates when you open it in Excel). Open it **after** your own attempt and use it to check your work.
*This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

1.15 Exercises

These are the take-home exercises that go with this chapter.

1. Download `bc_disturbance_reforestation.xlsx` from the BC Data Catalogue. Place it in the `data/` folder of your project. Open it and read the About sheet aloud.
2. In one sentence, write down what `Clearcutting_with_reserves_ha` measures **in your own words**, without looking at the ReadMe sheet. Then compare with the ReadMe.
3. Find the fiscal year with the smallest `Total_Disturbance_ha`. What year was it, and what was the value? *(Hint: `=MIN(...)` finds the value; sort the table or scan to find the year.)*
4. Add a new column `F_to_D` (reforestation-to-disturbance ratio) = `Reforestation_ha / Total_Disturbance_ha`. In what fraction of fiscal years was reforestation *greater than* total disturbance?
5. Set up the folder structure from the *Project folder structure* section above. Place a copy of the BC silviculture file in `data/`. Take a screenshot of your folder structure.
6. Save a cleaned copy of the workbook (rename it `bc_disturbance_reforestation_clean.xlsx`) into the `outputs/` folder. The raw file in `data/` should remain unchanged.
7. Apply ISO date format (`yyyy-mm-dd`) to any date in any file. Sort by that column. Confirm the sort works correctly.

1.16 Optional, advanced

Items in this section are not required for the main path through the chapter. Read them once you are comfortable with the basics above.

1.16.1 Named ranges

You can give a range of cells a name and refer to it by that name in formulas:

- Select cells A2:A38 (the `Fiscal_Year` column).
- Type `years` into the Name Box (left of the formula bar).
- Press Enter.

Now anywhere in the workbook you can write `=MAX(years)` instead of `=MAX(A2:A38)`. The formula is self-documenting. Manage named ranges with **Formulas → Name Manager** (or `Ctrl + F3`).

Skip named ranges in Chapter 1 if you find them overwhelming — the chapter does not depend on them. Chapter 2 will show **structured tables**, which solve the same problem more cleanly.

1.16.2 Custom number formats

Beyond the standard formats, Excel supports custom format codes. For example, the custom format `0.000 "ha"` displays `216304.919 ha`. The `"ha"` is part of the format, not the value.

This is useful for at-a-glance unit labels, but use sparingly — too much custom formatting can make a workbook harder for a collaborator to read.

1.17 A glimpse ahead: From Excel to R

In Chapter 4 we will install R and re-open *this same file* reproducibly. The Excel moves you learned in this chapter map directly to R commands you will meet later. Here is a preview:

What you did in Excel	What you will do in R (Ch 4–5)
Open the file by double-clicking	`read_excel(here("data", "bc_disturbance_reforestation.xlsx"), sheet = "Data")`
Look at the status bar `Count`	`dim(bc)` and `nrow(bc)`
Status-bar `Sum` of column H	`sum(bc$Reforestation_ha)`
Status-bar `Average` of column G	`mean(bc$Total_Disturbance_ha)`
Scan the Data sheet visually	`glimpse(bc)` and `summary(bc)`
Apply `MIN` and `MAX`	`range(bc$Natural_Disturbance_ha)`

The numbers will be exactly the same. Same dataset, same columns, same answers — but the R version is scriptable, shareable, and reproducible.

You do not need to remember any of this now. The point is just to see that R is not a fresh start. Everything you learn in Chapters 1–3 transfers directly. Chapter 4 will walk through each step in detail.

1.18 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

1.19 AI as a debugging companion

When to use AI in this chapter

- Ask AI to **define** a forestry term that appears in the ReadMe but is new to you. *"What does 'silvicultural system' mean?"*
- Ask AI to **explain** an Excel formula. Paste the formula verbatim and ask what it does.

1.19.1 When *not* to use AI in this chapter

- AI is not the source of truth for what a BC dataset measures. The BC indicator page is. Use the ReadMe and the source URL first.
- Do not paste an exercise question into AI and copy the answer back. The point of an exercise is the thinking, not the result.

1.20 Reading

- *About* and *ReadMe* sheets of `bc_disturbance_reforestation.xlsx`.
- BC Government silviculture indicator background: https://www.env.gov.bc.ca/soe/indicators/land/silviculture.html
- BC Data Catalogue dataset page: https://catalogue.data.gov.bc.ca/dataset/indicator-summary-data-trends-in-silviculture-in-b-c-
- Open Government Licence — British Columbia: https://www2.gov.bc.ca/gov/content/data/open-data/open-government-licence-bc

Instructor notes

Pacing and emphasis

- This chapter is deliberately beginner-friendly. The main path covers six skills: inspecting About / ReadMe / Data; identifying columns, units, and types; writing five simple formulas; checking whether values make sense; building a one-row data dictionary; setting

up a folder structure.

- **Named ranges** and **custom number formats** are moved to the *Optional, advanced* section. Touch on them only if your group is moving quickly.
- The group lab is the central hands-on deliverable. Plan to give learners time to complete all five tasks in one session.

1.20.1 Expected output checklist (matches the top-of-chapter callout)

1. Raw `bc_disturbance_reforestation.xlsx` in `data/`, unchanged.
2. Working `bc_disturbance_reforestation_clean.xlsx` in `outputs/` containing About / ReadMe / Data, a one-row data dictionary entry, and the five chapter formulas.
3. Group-lab worksheet or screenshot.

1.20.2 Materials provided alongside this chapter

File	Purpose
`bc_disturbance_reforestation.xlsx`	The raw open-data workbook (About + ReadMe + Data). Place in `data/`.
`bc_disturbance_reforestation_learner.xlsx`	Learner-facing exercises workbook with blanks and formula prompts. Distribute through the course site.
`bc_disturbance_reforestation_solutions.xlsx`	Instructor solutions with completed formulas and worked answers. **Do not distribute to learners.**
`quiz_ch01.html`	Standalone interactive quiz. Embedded in the rendered chapter; can also be hosted separately.

1.20.3 Suggested facilitation guidance for the Practice Demo Lab

*These are optional facilitation and feedback suggestions for instructors running this book demo — **not** a grading key. The percentages below are relative emphasis, not Canvas marks; the graded lab assignment is on Canvas.*

Task	Weight	What to look for
Inspect workbook (Task 1)	20 %	Group can articulate source, columns, row count.
Data dictionary (Task 2)	20 %	Two columns described in own words; compared with ReadMe.
Summary value (Task 3)	30 %	Correct formula AND a spot check.
Verify identity (Task 4)	20 %	`Harvested + Natural = Total` confirmed on at least one row.
Submission (Task 5)	10 %	Worksheet or screenshot is legible; group members named.

1.20.4 Common stumbling points

- Some learners try to compute the answer to Task 3 from the Summary sheet of the solutions file. Hide that file.
- Learners often write `=E2+F2=G2` and expect to see a number; they get `TRUE/FALSE` instead. That is correct — point out the difference between a comparison and an arithmetic expression.
- Some learners edit the raw `bc_disturbance_reforestation.xlsx` in place. Remind them that the raw file should stay unchanged.
- *Excel-to-R bridge:* If learners ask why we are starting in Excel "if we are going to move to R anyway", the answer is twofold: (a) the column meanings and forestry context are easier to learn with the data sitting visible on screen, and

  (b) Chapter 4 deliberately uses the *same file* to make the transition explicit. The "A glimpse ahead" table at the end of this chapter is the preview of that bridge.

2 Clean and Summarize Data in Excel

Data for this chapter

This chapter uses the **Ontario forest statistics** workbook (`on_forest_statistics_1.xlsx`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- **Sort** and **filter** large tables to surface unusual records.
- Detect **duplicate rows** on a single column and on a composite key.
- Use **structured tables** (`Ctrl+T`) to keep formulas connected to the data as it grows.
- Compute **single-cell summary statistics** — `SUM`, `AVERAGE`, `MEDIAN`, `MIN`, `MAX`, `COUNT`, `COUNTA`, `STDEV`.
- Use **relative, absolute, and mixed cell references** safely so a copied formula does what you expect.
- Apply **conditional aggregation** with `SUMIF`, `COUNTIF`, and `AVERAGEIF` (and their plural counterparts).
- Use **conditional logic** with `IF`, nested `IF`, and `IFS` to recode values into categories.
- **Look up** values from a reference table with `VLOOKUP` and `XLOOKUP`.
- Maintain a **cleaning log** that documents every change you made to the data.

Expected output for this chapter

What you will hand in

1. The raw workbook **`on_forest_statistics_1.xlsx`** placed in your `chapter02/data/` folder, **unchanged from the Ontario Open Data download**.
2. A working copy you may modify, saved as **`on_forest_statistics_1_clean.xlsx`** in your `chapter02/outputs/` folder, containing:
   - the dataset converted to a **structured table** with a meaningful name,
   - a one-page **cleaning log sheet**,
   - a `size_class` column built with `IFS`, and
   - a small **forest-type lookup** sheet wired to the Data sheet with `XLOOKUP`.

3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

2.1 Why a cleaning log matters

Real data arrives messy. Duplicate stems. Inconsistent species labels. Missing measurements. Unit ambiguity. The discipline introduced in this chapter — *document every change, never edit data silently* — is the foundation of every later chapter.

A **cleaning log** is a separate sheet (or document) that records every change you made between the raw file and your analysis-ready version:

Date	Step	What I changed	Why
2026-09-15	1	Converted Data sheet into a structured table named `ontario_data`	Self-documenting formulas; auto-extends.
2026-09-15	2	Added a `size_class` column using `IFS`	Group small / medium / large areas for later summaries.
2026-09-15	3	Built a `codes` sheet mapping forest-type codes to plain-English names	Future joins / lookups need a clean reference.
2026-09-15	4	Verified the composite key has no duplicates	Confirms the file is a true cross-classification.

Keep the cleaning log next to your data forever. When someone asks *"why is your number different from the official report?"* — the answer is in the cleaning log.

⚠ The two cardinal sins of data cleaning

1. **Editing raw data in place.** Once you change a cell, the original is gone. Always work on a *copy* of the raw file.
2. **Cleaning without a log.** Without documentation, your numbers become impossible to defend.

A cleaning log is your audit trail. It protects you when a manager, reviewer, or collaborator questions your numbers months later.

2.2 The dataset for this chapter

This chapter uses one openly licensed Excel workbook from the **Government of Ontario Open Data Catalogue**, alongside the BC silviculture file you met in Chapter 1.

Property	Value
File name	`on_forest_statistics_1.xlsx`
Source	Government of Ontario — *Forest Resources of Ontario 2021*
Catalogue page	https://data.ontario.ca/dataset/forest-resources-of-ontario-2021
Licence	Open Government Licence — Ontario
Rows in this teaching extract	179
Columns	8
Coverage	Region 3E only (teaching subset of the multi-region file)
Time period	Snapshot for 2021

! Where the data comes from — and what to do with it

This dataset is published by the **Government of Ontario** and distributed through the **Ontario Open Data Catalogue**. Two rules (same as Chapter 1):

1. **Keep the raw downloaded file unchanged.** Save the file as `on_forest_statistics_1.xlsx` exactly as you downloaded it. Do not sort it, edit it, or save over it.
2. **Save any cleaned or modified version separately** as `on_forest_statistics_1_clean.xlsx` in your `outputs/` folder.

The teaching extract in this book is **Region 3E only**. If you need the full multi-region file (all landscape guide regions), download it directly from the catalogue URL above.

2.2.1 Why this dataset matters for forestry

Where the BC silviculture file (Chapter 1) was a **time series** of provincial totals, this Ontario file is a **cross-classification** — every row is one combination of region × ownership × seral stage × forest type × age class, with the total area for **that combination**.

This **shape** — how the rows and columns are laid out (here, one row per category combination) — is what a working forester sees most often:

- The Government of Ontario uses cross-classified tables like this to track stand structure across the landbase.
- BC's *Vegetation Resources Inventory* (VRI) and the National Forest Inventory both produce summary tables with the same shape.
- Any FRPA, FMP, or sustainable-forest-management plan eventually rolls up to a table like this.

Learning to navigate, summarize, and recode a cross-classified file is one of the most transferable skills in forestry data work.

2.2.2 The columns

Column	Type	What it represents
`landscape_guide_region`	text	Ontario landscape guide region (here: 3E only)
`own_group`	text	Ownership group (here: CRN = Crown only)
`LGDS`	text	Seral stage code (P / S / I / M / L)
`SERAL_NAME`	text	Human-readable seral stage name
`PFT`	text	Provincial forest type code (3-letter)
`PFTName`	text	Human-readable forest type name
`AC_10`	number (year)	Age-class midpoint, in 10-year bins
`SumOfTotalHa`	number	Total area in hectares for that combination

> **i** A note for Mac users
>
> The Excel chapters are written for **Windows**, but almost everything is identical on **Mac** with two routine substitutions: press **Cmd** wherever the text says **Ctrl** (e.g. Cmd+T, Cmd+Click), and a few ribbon tabs are named slightly differently (for example **Table Design** on Windows is the **Table** tab on Mac). Where a Mac step differs in a way that is easy to miss, a **Mac:** note is called out inline.

2.3 Sorting

Click any cell inside the data, then choose **Data → Sort**.

A dialog opens. Choose:

- **Column**: `SumOfTotalHa`
- **Sort On**: Values
- **Order**: Largest to Smallest

Click OK. The largest cell rises to the top — a mature Conifer Lowland area of over 300,000 ha. Look at the context: `MCL`, `Mature`, an age class around 135. Does it make sense for a Region 3E mature lowland that has been growing for over a century? Yes — old conifer lowland covers much of northern Ontario.

2.3.1 Multi-key sort

Sort by `PFTName` (A→Z), then by `AC_10` (smallest to largest), then by `SumOfTotalHa` (largest to smallest). Use **Data → Sort → Add Level** to add each additional sort column. **Mac:** the Sort dialog has no *Add Level* button — click the + button below the list of sort fields to add a level (and − to remove one).

Now the table reads like a story: for each forest type, you see every age class in order, and within each age class the largest area is on top. This is the human equivalent of `group_by() |> arrange()` in R, which you will meet in Chapter 7.

Sorting changes the file

Unlike filtering, sorting **rearranges** the underlying rows. To recover the original order:

- Add an `original_row_number` column before sorting (formula: `=ROW()` in the first row, dragged down), **or**
- Convert your data into a **structured table** first (next section) — sorts on a table do not destroy original order if you have an `original_row_number` column.

2.4 Filtering

AI prompt to check your work

Use this **after** your own attempt. It should check your reasoning, not hand you the answer:

> *"In Excel I filtered the data to show only `[describe your criteria]` and got `[N]` visible rows. Check whether my criteria actually capture what I intended, whether I might be hiding rows I meant to keep, and how to confirm the visible count is right. Do not give me the final filter."*

Two ways: **AutoFilter** and **Slicers**.

2.4.1 AutoFilter

Click any cell in your data, then **Data → Filter**. A small dropdown **arrow appears on each column header**.

Click the arrow on the `PFTName` header, uncheck *Select All*, and check only "Jack Pine". The view collapses to roughly two dozen rows. **The data is unchanged** — you have only *hidden* rows, not deleted them.

To re-show everything, click the arrow again and choose *Clear Filter From*.

2.4.2 Multiple filter criteria

Click the arrow on `SumOfTotalHa`. Choose **Number Filters → Greater Than**. Type `50000`. The view filters to rows where the area exceeds 50,000 hectares. **Mac:** there is no *Number Filters* submenu — open the column arrow, use the **"Choose One"** dropdown, pick **Greater Than**, and enter `50000`.

You can stack column filters: first filter `PFTName = "Jack Pine"`, *then* filter `SumOfTotalHa > 5000`. Excel shows only the rows matching **both** conditions — equivalent to `filter(PFTName == "Jack Pine" & SumOfTotalHa > 5000)` in R.

2.4.3 Slicers (with structured tables)

Slicers are visual filter panels. They only work on **structured tables** (next section). Click any cell in a structured table, then **Table Design → Insert Slicer**. Check which fields you want.

Slicers are clickable buttons sitting beside the table. They are visual, fun, and dangerous: a slicer can be on without the viewer noticing. The number changes but the title does not. Always check the slicer state before trusting a screenshot.

2.5 Structured tables (`Ctrl+T`)

Excel's most undersold feature.

Click any cell in your data. Press **Ctrl+T** (Cmd+T on Mac). Confirm *"My table has headers"*. Click OK.

What changed:

- The data became a *structured table* with a name (default `Table1`, which you should rename in the **Table Design** ribbon — **Mac:** the **Table** tab — try `ontario_data`).
- Headers got a coloured band.
- AutoFilter is automatically on.
- Formulas that reference table columns use *column names* instead of cell ranges:

```
=AVERAGE(ontario_data[SumOfTotalHa])
```

Compare to the un-tabled version:

```
=AVERAGE(H2:H180)
```

The structured version is **self-documenting** and **self-resizing**. If you add a row at the bottom, the formula automatically includes it.

Always start with Ctrl+T

When you open any new Excel file, the first thing to do is convert each sheet's data into a structured table. The benefits compound: slicers work, formulas are readable, growth is automatic, and references are stable.

2.5.1 Renaming the table

Click anywhere in the table. Look at the **Table Design** ribbon (**Mac:** the **Table** tab). In the leftmost field labelled *Table Name* — on Mac, the **Table Name** box at the left of that tab — type a meaningful name:

- `ontario_data`
- `bc_silviculture`
- `Table1`, `Sheet1!_FilterDatabase`

Use snake_case (lowercase with underscores). No spaces, no special characters. This matches the R convention you will meet in Chapter 4.

2.6 Detecting duplicates

A real summary table should have **one row per combination** of the grouping columns. If two rows have the same combination, something went wrong upstream.

2.6.1 On a single column

Select the column you want to check (e.g., `PFTName`). Choose **Home → Conditional Formatting → Highlight Cells Rules → Duplicate Values**. Duplicate values get highlighted.

For the Ontario file, every `PFTName` appears many times (because the file has many seral × age combinations per forest type). That is expected — not a bug. Duplicates on a *single* column only signal a problem when the column should be a *unique key*.

2.6.2 On a composite key

In a cross-classification like this file, the natural unique key is `(landscape_guide_region, own_group, LGDS, PFTName, AC_10)`. A composite key. To detect duplicates on it:

Add a helper column `composite_key` next to the data:

```
=A2 & "|" & B2 & "|" & C2 & "|" & F2 & "|" & G2
```

The `|` separator prevents false collisions between, say, `"AB" + "C"` and `"A" + "BC"`. If any concatenated key appears twice, the cell highlights when you apply **Conditional Formatting → Highlight Cells Rules → Duplicate Values** to the helper column.

```
Verification in R: 0 duplicate key combinations in the Ontario file.
```

For this file, the answer is zero. Good — the upstream pipeline is clean.

2.6.3 Removing duplicates

If you do find duplicates and want to remove them, **Data → Remove Duplicates**. A dialog asks which columns to use as the key. Always work on a copy. Document the removal in your cleaning log with row counts before and after.

2.7 Summary formulas — the essentials

The seven most-used aggregating functions:

Formula	What it does	Example
`=SUM(range)`	Adds every numeric value	`=SUM(H2:H180)`
`=AVERAGE(range)`	Arithmetic mean (sum ÷ count)	`=AVERAGE(H2:H180)`
`=MEDIAN(range)`	Middle value when sorted; robust to outliers	`=MEDIAN(H2:H180)`
`=MIN(range)`	Smallest numeric value	`=MIN(H2:H180)`
`=MAX(range)`	Largest numeric value	`=MAX(H2:H180)`
`=COUNT(range)`	Number of *numeric* values	`=COUNT(H2:H180)`
`=COUNTA(range)`	Number of *non-empty* values (text + numeric)	`=COUNTA(A2:A180)`
`=STDEV(range)`	Standard deviation (sample)	`=STDEV(H2:H180)`

The distinction between `COUNT` and `COUNTA` matters: `COUNT` skips text cells. If your column should have 179 numeric entries and `COUNT` says 177, two cells contain text or are blank.

2.7.1 Worked examples on the Ontario file

Statistic	Value
Total area covered (sum of SumOfTotalHa)	9.00 million ha
Mean area per combination	50,257 ha
Median area per combination	12,676 ha
Maximum combination	320,678 ha
Number of rows	179

Look at the gap between mean and median. The mean is much larger — a few very large rows pull the average up. The median is closer to the *typical* combination. Both numbers are correct; they answer different questions.

2.8 Conditional aggregation: SUMIF, COUNTIF, AVERAGEIF

What if you want the sum, count, or average **only for rows that match a condition**?

2.8.1 `COUNTIF(range, criterion)`

How many rows describe Jack Pine?

```
=COUNTIF(F2:F180, "Jack Pine")
```

The criterion can be a number, text, a comparison, or a wildcard:

Criterion	Matches
`50000`	Exact 50000
`">50000"`	Greater than 50000
`"<>0"`	Anything not zero
`"Jack*"`	Text starting with "Jack"
`"*pine*"`	Text containing "pine" (case-insensitive)

2.8.2 `SUMIF(criterion_range, criterion, sum_range)`

What is the total area of "Mature" forest in Region 3E?

```
=SUMIF(D2:D180, "Mature", H2:H180)
```

Read it: *"sum the `SumOfTotalHa` values for rows where `SERAL_NAME` equals 'Mature'".*

2.8.3 `AVERAGEIF(criterion_range, criterion, average_range)`

What is the average area for "Jack Pine" combinations?

```
=AVERAGEIF(F2:F180, "Jack Pine", H2:H180)
```

2.8.4 `SUMIFS`, `COUNTIFS`, `AVERAGEIFS` — multiple criteria

The plural versions allow multiple criteria. The argument order is different — the sum range comes **first**, then criterion pairs:

```
=SUMIFS(H2:H180, F2:F180, "Jack Pine", D2:D180, "Mature")
```

Read it: *"sum the `SumOfTotalHa` values for rows where `PFTName` is 'Jack Pine' AND `SERAL_NAME` is 'Mature'".*

When to use these vs a PivotTable

For a one-off number, `SUMIF` is fastest. For a whole table of numbers (cross-classification across multiple categories), a PivotTable is the right tool. We will meet PivotTables in Chapter 3.

2.9 Safe cell references: the dollar sign

Type `=B2 + C2` in cell `D2`. Copy down. `D3` becomes `=B3 + C3`. `D4` becomes `=B4 + C4`. The references **moved** with the formula. That is a **relative** reference.

Type `=B2 + $C$2` in cell `D2`. Copy down. `D3` becomes `=B3 + $C$2`. `D4` becomes `=B4 + $C$2`. The first part moves; the second part **stays put**. The dollar signs locked it. That is an **absolute** reference.

You can mix:

- `$A1` — column locked, row moves.
- `A$1` — row locked, column moves.
- `$A$1` — both locked.

2.9.1 A worked bug

A common bug: you want to divide every value by the total. You write `=H2 / H180` in cell `I2`, where `H180` is the grand total. Copy down. `I3` becomes `=H3 / H181`. But `H181` is empty. Every percentage below the first is wrong.

The fix is `=H2 / $H$180`. The denominator stays put.

A keyboard shortcut worth memorizing

While editing a cell reference, press **F4** to cycle through:
`$A$1` → `A$1` → `$A1` → `A1` → `$A$1` → …
This is the fastest way to add or remove dollar signs.

2.10 Conditional logic: IF, nested IF, IFS

2.10.1 `IF(condition, value_if_true, value_if_false)`

Basic form. Returns one of two values based on a logical test.

```
=IF(H2 > 1000, "large", "small")
```

2.10.2 Nested `IF` — multiple categories

Add a `size_class` column on the Ontario file based on `SumOfTotalHa`:

```
=IF(H2 < 1000, "small",
    IF(H2 < 50000, "medium", "large"))
```

This **nested IF** evaluates left to right. If the first condition is true, return "small". Otherwise check the second. Otherwise return "large".

2.10.3 `IFS` — cleaner for many conditions

Available in Excel 2019+:

```
=IFS(H2 < 1000, "small",
     H2 < 50000, "medium",
     TRUE, "large")
```

`IFS` evaluates each pair left to right. The first matching condition returns its value. The trailing `TRUE` is a catch-all for "everything else". Without it, unmatched rows return `#N/A`.

2.10.4 `SWITCH` — exact-match recoding

When you are recoding based on **one cell's exact value** (not a threshold test), `SWITCH` is often clearer than a nested `IF`:

```
=SWITCH(D2, "Mature", "old", "Young", "new", "other")
```

`SWITCH` compares `D2` to each value in turn and returns the label after the first match; the final lone argument is the catch-all default. Rule of thumb: use `IFS` for **range/threshold** tests and `SWITCH` for **exact-match** recoding.

size_class	n
large	60
medium	69
small	50

2.10.5 `AND`, `OR`, `NOT` — combining conditions

```
=IF(AND(H2 > 1000, F2 = "Jack Pine"), "flag", "")
```

Use `AND` when *all* conditions must be true. Use `OR` when *any* condition can be true. Use `NOT` to reverse a condition.

2.11 Lookups: VLOOKUP and XLOOKUP

Suppose you have a small reference sheet mapping forest-type codes to plain-English names:

Code	Name
PJK	Jack Pine
MCU	Conifer Upland
MCL	Conifer Lowland
MIX	Mixedwood
POP	Poplar
BWT	White Birch
PWR	Red and White Pine
TOL	Tolerant Hardwoods

Place this on a new sheet called `codes`. Your data uses the code in column E. In column I you want the plain-English name.

2.11.1 `XLOOKUP` (Excel 2019+, recommended)

```
=XLOOKUP(E2, codes!A:A, codes!B:B, "unknown")
```

Arguments: *value to find, where to look, what to return, default if not found.* Clear and modern.

> **i** The sheet name in the formula must match
>
> `codes!A:A` refers to a sheet named **`codes`**. If your lookup sheet has a different name — or you paste the formula before renaming the sheet — Excel returns `#REF!` or `#N/A`. Rename the reference sheet first, or fix the sheet name in the formula.

2.11.2 `VLOOKUP` (everywhere)

```
=VLOOKUP(E2, codes!A:B, 2, FALSE)
```

Arguments: *value to find, table range starting with the lookup column, column number to return, exact match.* The `FALSE` at the end forces exact match — **always include it**. The default of `TRUE` (approximate match) silently returns the wrong value if the lookup table is unsorted.

> **!** When lookups go wrong
>
> - **Missing key** — `#N/A`. Either the value is missing, or the key is misspelled. Wrap with `IFERROR` to flag rather than crash: `=IFERROR(VLOOKUP(...), "missing")`.
> - **Duplicate key** — silently returns the *first* match. Always check that your lookup table has each key exactly once with `=COUNTIF(codes!A:A, "PJK")` (should return 1).
> - **Wrong type** — looking up `"123"` (text) against `123` (number) fails. Inspect both sides.

- **Trailing spaces** — `"PJK "` does not match `"PJK"`. Use `=TRIM(...)` to clean strings first.

2.12 Putting it together — a worked cleaning sequence

You receive a colleague's spreadsheet. Here is a complete cleaning sequence:

Step	Action	Excel feature
1	Make a working copy of the raw file	File → Save As (with `_clean` suffix)
2	Open the working copy	—
3	Convert data to structured table	Ctrl+T, rename in Table Design
4	Apply `TRIM` to text columns to remove stray spaces	Helper column with `=TRIM(A2)`
5	Check for duplicates on the natural key	Conditional Formatting on composite key
6	Investigate missing values column by column	`=COUNTBLANK(...)` per column
7	Apply `IFS` to recode messy categories	New column with recoded values
8	Look up reference values where needed	`XLOOKUP` against reference sheet
9	Compute summary statistics	`SUMIF`, `COUNTIF`, `AVERAGEIF`
10	Write the cleaning log	New sheet, document every change

Each step is reversible because we are working on a copy and never modifying the raw values directly. Every step is documented in the log. **This is what reproducible Excel looks like.**

2.13 Active learning

These short activities are designed to be done alongside the chapter.

2.13.1 Activity 1 — Find duplicates in the Ontario file

Apply the duplicate-detection technique to the Ontario file. Use the natural composite key. Report how many duplicates you found.

```
Verification: 0 duplicates on the natural key (should be zero).
```

2.13.2 Activity 2 — Mature area by forest type

Using `SUMIF` (or `SUMIFS`), compute the total area of *mature* forest in Region 3E for each forest type.

PFTName	mature_ha
Conifer Lowland	903,009
Conifer Upland	745,944
Mixedwood	471,449
White Birch	242,519
Poplar	219,451
Jack Pine	92,894
Red and White Pine	4,026
Tolerant Hardwoods	1,142

2.13.3 Activity 3 — Classify and count

On the Ontario file, add the `size_class` column from the conditional-logic section. Then count rows in each class. Compare with your neighbour.

2.13.4 Activity 4 — Multi-criteria SUMIFS

Compute the total area where `PFTName = "Jack Pine"` **AND** `SERAL_NAME = "Mature"` **AND** `AC_10 >= 95` (older mature jack pine).

```
Verification: 33,991 ha of older mature jack pine.
```

2.13.5 Activity 5 — Build a lookup

Build a reference sheet `codes` with the 8 forest-type codes (`PJK`, `MCU`, `MCL`, etc.) and their plain-English names. Then use `XLOOKUP` (or `VLOOKUP`) to retrieve the name from the `PFT` column of the Data sheet. Compare your retrieved column to the existing `PFTName` column.

2.14 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.
2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?
4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

i Lab: Clean and summarize the Ontario forest file together

Work in groups of 3 or 4. Use one shared copy of `on_forest_statistics_1.xlsx`.
Task 1 — Save a working copy. Save the workbook as `on_forest_statistics_1_clean.xlsx` in your `outputs/` folder. The raw file in `data/` should remain unchanged.
Task 2 — Make it a structured table. Convert the Data sheet into a structured table. Rename it to `ontario_data`. Confirm that autofilter dropdowns appear in the header row.
Task 3 — Add a `size_class` column. Use `IFS` (or nested `IF`) to classify each row's `SumOfTotalHa` into `small` / `medium` / `large` using the thresholds 1,000 and 50,000. Then use `COUNTIF` to count how many rows fall in each class. Compare across groups.
Task 4 — One conditional aggregation. As a group, choose **one** of these:

- Total area of *Jack Pine* in Region 3E (use `SUMIF`)
- Average area for *Mature* combinations (use `AVERAGEIF`)
- Count of rows where `SumOfTotalHa > 100,000` (use `COUNTIF`)

Write the formula. Confirm the result with a spot check.
Task 5 — Submit. Submit either a short worksheet (a one-page document with your group's answers to Tasks 1–4) **or** a screenshot of the cleaned workbook showing the structured table and the new `size_class` column. Include all group members' names on the submission.
This lab is designed to fit comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 2 reference solution (HTML answer key) — the completed, task-by-task answer key — or **download the completed Excel workbook** with the formulas worked into the sheets (it recalculates when you open it in Excel). Open it **after** your own attempt and use it to check your work.
*This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

2.15 Exercises

These are the take-home exercises that go with this chapter.

1. Open `on_forest_statistics_1.xlsx`. Filter to `LGDS = "M"` (Mature). What is the total area, in hectares, of mature forest in Region 3E? Use `SUMIF`.
2. Open `bc_disturbance_reforestation.xlsx` (from Chapter 1). Add a `Decade` column using `=FLOOR(A2, 10)`. Then use `AVERAGEIF` to compute the mean `Harvested_ha` per decade.
3. Build a `codes` lookup sheet on the Ontario file with the eight forest-type codes and their plain-English names. Add a `PFTName_check` column that uses `XLOOKUP` to retrieve the name from the code. Use `IF` to flag any mismatches between the looked-up name and the existing `PFTName` column.
4. Compute the *coefficient of variation* (standard deviation divided by mean) of `SumOfTotalHa` on the Ontario file. Why might this be useful alongside the mean?
5. Apply conditional formatting to the `SumOfTotalHa` column with a three-colour scale: green for small, yellow for medium, red for large. Take a screenshot and submit.
6. Write a one-page cleaning log for the changes you made to the working copy of the Ontario file. Use the format shown at the top of this chapter. Submit the log next to the workbook.
7. **Optional, harder.** Build a small summary table on a new sheet of the Ontario file using only `SUMIFS` and `COUNTIFS` (no PivotTable yet — that's Chapter 3). Rows = forest type, columns = seral stage, values = sum of `SumOfTotalHa` and count of combinations.

2.16 Optional, advanced

Items in this section are not required for the main path.

2.16.1 Custom number formats

Beyond standard formats, Excel supports custom format codes. For example, the custom format `#,##0 "ha"` displays `216,305 ha`. The `"ha"` is part of the format, not the value — formulas still treat the cell as a plain number.

2.16.2 INDEX / MATCH — the classic lookup combo

VLOOKUP and XLOOKUP cover most cases. For old-Excel compatibility or for *left-of-lookup-column* returns (which VLOOKUP cannot do), use INDEX / MATCH:

```
=INDEX(codes!B:B, MATCH(E2, codes!A:A, 0))
```

MATCH(E2, codes!A:A, 0) returns the row number of the match. INDEX(codes!B:B, n) returns the value in column B at that row. Combining them gives you a fully flexible lookup.

2.17 A glimpse ahead: From Excel to R

In Chapter 7 (Summarize forestry data overall and by group) you will do exactly what this chapter does — but in R. The Excel moves you learned here map directly to R commands:

What you did in Excel (Ch 2)	What you will do in R (Ch 6–7)
`=SUMIF(D2:D180, "Mature", H2:H180)`	`ontario |> filter(SERAL_NAME == "Mature") |> summarise(sum(SumOfTotalHa))`
`=COUNTIF(F2:F180, "Jack Pine")`	`ontario |> filter(PFTName == "Jack Pine") |> nrow()`
`=SUMIFS(H, F, "Jack Pine", D, "Mature")`	`ontario |> filter(PFTName == "Jack Pine", SERAL_NAME == "Mature") |> summarise(sum(SumOfTotalHa))`
`=IFS(H2<1000, "small", H2<50000, "medium", TRUE, "large")`	`mutate(size_class = case_when(SumOfTotalHa < 1000 ~ "small", SumOfTotalHa < 50000 ~ "medium", TRUE ~ "large"))`
`=VLOOKUP(E2, codes!A:B, 2, FALSE)`	`ontario |> left_join(codes, by = "PFT")`
Structured table named `ontario_data`	`ontario` (the tibble)
Cleaning log on a separate sheet	A Markdown chunk in the same `.qmd` file

The intent is identical. The R version is scriptable and shareable; the Excel version is visible and forgiving. By the time you reach Chapter 7, this whole chapter's logic will be familiar — you will just be writing it in a different language.

2.18 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

2.19 AI as a debugging companion

Useful prompts for this chapter

- *"In Excel, my formula `=VLOOKUP(A2, codes!A:B, 2, FALSE)` returns `#N/A` for some rows. The values in column A look the same as in the codes sheet. What are the three most common reasons VLOOKUP fails when the values look right?"*
- *"Explain step by step what this formula does: `=IFS(C2 < 1000, "small", C2 < 50000, "medium", TRUE, "large")`"*
- *"I want to write a `SUMIFS` that adds up the `SumOfTotalHa` column whenever `PFTName` is one of `{Jack Pine, Mixedwood, Poplar}`. Can I do this in a single formula?"*

2.19.1 Verifying AI's answers

After AI tells you why your `VLOOKUP` fails, **check**:

1. Are there trailing spaces? `=LEN(A2)` and `=LEN(codes!A2)` should match.
2. Are both values the same type? `=ISTEXT(A2)` and `=ISTEXT(codes!A2)`.
3. Is the case the same? VLOOKUP is case-insensitive in Excel, but worth eliminating.

AI's answer is a *starting point*, not the final answer. Always verify against the data.

2.19.2 When not to use AI

- Do not use AI to *write the cleaning log for you*. The log is a record of *your* decisions.
- Do not use AI to *invent* lookup values that are not in the reference table. If a value is missing, document the gap.

2.20 Reading

- Microsoft Excel official documentation on VLOOKUP.
- Microsoft Excel official documentation on XLOOKUP.
- Ontario open-data dataset page: https://data.ontario.ca/dataset/forest-resources-of-ontario-2021
- Open Government Licence — Ontario: https://www.ontario.ca/page/open-government-licence-ontario

Instructor notes

Pacing and emphasis

- This chapter is heavier than Chapter 1. Plan to spread it across multiple sessions.
- The main path covers seven skills: sort, filter, structured tables, summary formulas, conditional aggregation, conditional logic, lookups. The cleaning log ties them together.
- **Custom number formats** and **INDEX/MATCH** are moved to the *Optional, advanced* section. Touch on them only if your group is moving quickly or already comfortable.
- The group lab is the central hands-on deliverable. Plan for one full session.

2.20.1 Expected output checklist (matches the top-of-chapter callout)

1. Raw `on_forest_statistics_1.xlsx` in `chapter02/data/`, unchanged.
2. Working `on_forest_statistics_1_clean.xlsx` in `chapter02/outputs/` containing a structured table, a cleaning log sheet, a `size_class` column built with `IFS`, and a small `codes` lookup sheet wired with `XLOOKUP`.
3. Group-lab worksheet or screenshot.

2.20.2 Materials provided alongside this chapter

File	Purpose
`on_forest_statistics_1.xlsx`	The raw open-data workbook (About + ReadMe + Data). Place in `data/`.
`on_forest_statistics_1_learner.xlsx`	Learner-facing exercises workbook with blanks and formula prompts. Distribute through the course site.
`on_forest_statistics_1_solutions.xlsx`	Instructor solutions with completed formulas and worked answers. **Do not distribute to learners.**
`quiz_ch02.html`	Standalone interactive quiz. Embedded in the rendered chapter; can also be hosted separately.

2.20.3 Suggested facilitation guidance for the Practice Demo Lab

*These are optional facilitation and feedback suggestions for instructors running this book demo — **not** a grading key. The percentages below are relative emphasis, not Canvas marks; the graded lab assignment is on Canvas.*

Task	Weight	What to look for
Save working copy (Task 1)	10 %	Cleaned file in `outputs/`; raw file untouched.
Structured table (Task 2)	20 %	Table named meaningfully; autofilter present.
size_class column (Task 3)	30 %	`IFS` formula correct; counts in each class confirmed.
Conditional aggregation (Task 4)	30 %	Correct formula AND a spot check.
Submission (Task 5)	10 %	Worksheet or screenshot is legible; group members named.

2.20.4 Common stumbling points

- Some learners forget the trailing `TRUE` in `IFS` and get `#N/A` on rows that don't match any explicit condition.
- Some try to write a structured-table formula by typing the column name without the table prefix. The right form is `ontario_data[SumOfTotalHa]`, not `SumOfTotalHa` alone.
- A few will edit the raw `on_forest_statistics_1.xlsx` in place. Remind them often that the raw file stays unchanged.
- *Excel-to-R bridge:* The "A glimpse ahead" section at the end of this chapter previews how the same logic appears in R. Refer to it when learners ask why we are still in Excel — the bridge is concrete and forward-pointing.

3 Excel Visualization and PivotTables

Data for this chapter

This chapter uses the **BC silviculture** (`bc_disturbance_reforestation.xlsx`) and **Ontario forest statistics** (`on_forest_statistics_1.xlsx`) workbooks. Click each file name to download it directly, then save it in your `data/` folder. For sources and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Choose the **right chart type** for the question being asked (bar, column, line, scatter).
- Build clean Excel charts with **titles, axis labels, and captions** that another reader could understand on their own.
- Construct a **PivotTable** from a tidy dataset, choosing what goes in Rows, Columns, Values, and Filters.
- Switch the **aggregation function** (sum, count, average) inside a PivotTable and predict what each one returns.
- Connect a **PivotChart** to a PivotTable and keep them in sync.
- Add **Slicers** to a PivotTable to give a non-technical reader controls to explore the data.
- **Refresh** PivotTables when the underlying data changes, and recognize the silent-bug risk if you forget.
- Recreate a PivotTable manually with **`SUMIFS` and `COUNTIFS`** — the bridge to how the same logic works in R.
- Export a finished chart to a **PNG** with a caption ready for a report or a course-site submission.

Expected output for this chapter

What you will hand in

1. **One PivotTable + PivotChart** built on `on_forest_statistics_1.xlsx`, answering one forestry question of your choice (e.g., "Total mature area by forest type" or "Count of combinations by seral stage and age class"). Exported as a **PNG with caption**.
2. **One time-series chart** built on `bc_disturbance_reforestation.xlsx`, showing reforestation, harvest, and natural disturbance from 1987 to 2023 on the same axes. Exported as a **PNG with caption**.

3. **One short worksheet** matching common forestry questions to the correct chart type (provided in the learner workbook).
4. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

3.1 Why visualization matters

A number in a cell is a fact. A chart is an argument. The same data, in the same workbook, can become a *story about reforestation finally catching up to harvest* (a single time-series chart) or a *story about a record-breaking wildfire season* (a bar chart with 2017 visibly above the rest). Choosing how to display data is part of the analysis, not an after-the-fact decoration.

The transition matrix for this course explicitly treats Chapter 3 as the **bridge** from raw Excel work into tidy-data thinking. The PivotTable you build in this chapter is conceptually the same as a `group_by() |> summarise()` pipeline in R — which you will write in Chapter 7. The chart you build is the conceptual ancestor of `ggplot2`, which you will meet in Chapter 9.

For now, the goal is to be **fluent** in the Excel versions, so that when we move to R, you already know what each command is doing.

3.2 The two datasets for this chapter

This chapter uses two openly licensed files you have already met:

Property	BC silviculture	Ontario forest stats
File	`bc_disturbance_reforestation_clean.xlsx`	`forest_statistics_1.xlsx`
Source	BC Government / BC Data Catalogue	Government of Ontario / Ontario Open Data Catalogue
Used for	Time-series charts (line, area)	PivotTables, cross-classification charts
Shape	Time series (one row per fiscal year)	Cross-classification (region × ownership × seral × type × age)
Rows × Cols	37 × 8	179 × 8

! File-handling rules carry over

Same two rules as Chapters 1 and 2:

1. **Keep the raw downloaded files unchanged** in your `data/` folder.
2. **Save any cleaned or chart-ready version** under `_clean.xlsx` in your `outputs/` folder.

You will work mostly on the *clean* copies for this chapter.

3.3 Choosing the right chart type

Before opening the chart dialog, answer one question: **what is the chart for?**

If your question is…	The natural chart is…
How has this number changed over time?	**Line chart** (or area chart)
How do these categories compare?	**Bar chart** (horizontal) or **column chart** (vertical)
How does Y relate to X across observations?	**Scatter plot**
What share does each part take of a whole?	**Stacked bar** (or pie, sparingly)
Where are the extremes / outliers?	**Bar chart** sorted by value
How does one group differ from another over time?	**Multi-series line chart**

> ⚠ When **not** to use a pie chart
>
> Pie charts are easy to draw and hard to read. The human eye is bad at comparing angles. Use a sorted bar chart instead, except for:
>
> - 2–3 categories where you specifically want to communicate *share of whole*,
> - A clear majority slice (e.g., "97 % vs 3 %").
>
> In every other case, a **sorted bar chart** is the better tool.

3.3.1 Worked example — choosing a chart for BC silviculture

Suppose you want to answer: *"How has BC reforestation kept pace with disturbance since 1987?"*

The question is **about change over time**. So: line chart, with two series:

- `Total_Disturbance_ha` over `Fiscal_Year`
- `Reforestation_ha` over `Fiscal_Year`

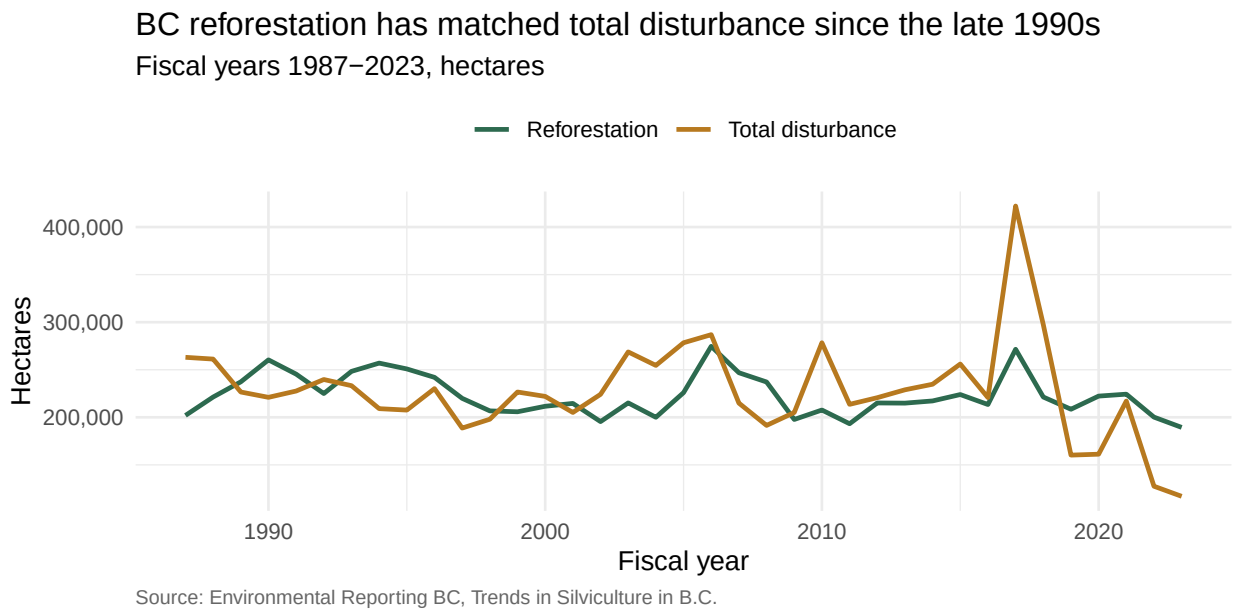

Figure 3.1: Preview of an Excel time-series chart: BC total disturbance and reforestation area (ha) by fiscal year, 1987–2023.

In Excel:

1. Select the three columns `Fiscal_Year`, `Total_Disturbance_ha`, and `Reforestation_ha`. If they are not next to each other, click the first column, then hold **Ctrl** (**Cmd** on Mac) and click the others to select non-adjacent columns.
2. **Insert → Charts → Line chart**.
3. Add chart elements: title, axis titles, data labels (sparingly), caption / source line.

The 2017 spike — BC's record wildfire season — is the most visible feature. The chart itself is the argument.

3.4 Chart anatomy

A finished chart has six elements. Every one is editable; many are hidden by default.

Element	Why it matters	How to edit
Title	Tells the reader what the chart is *for*. Should be a one-line sentence.	Click chart → Chart Elements (+) → Chart Title
Axis titles	Tell the reader what each axis is measuring (and in what units).	Chart Elements → Axis Titles
Legend	Identifies each series. Skip if there is only one series.	Chart Elements → Legend

Element	Why it matters	How to edit
Data labels	Numbers on the data points. Use sparingly.	Chart Elements → Data Labels
Source / caption	One line at the bottom naming the data source and date.	Insert → Text Box, anchor below chart
Theme	Colour and font choices. Should match the rest of the deliverable.	Chart Design ribbon → Chart Styles

3.4.1 Title rules

A bad title is *"Chart 1"* or *"Reforestation_ha vs Fiscal_Year"*.

A good title is *"BC reforestation has matched total disturbance since the late 1990s"* — a one-sentence claim the chart supports.

If you cannot write a one-sentence claim for the chart, the chart is the wrong chart for your question.

3.4.2 Axis labels and units

Always include units in axis labels: *"Hectares"*, not just *"Area"*. *"Fiscal year"*, not just *"Year"*.

For very large numbers, use a thousands separator and consider abbreviating the axis (200K instead of 200,000). Excel does this through **Format Axis → Display units → Thousands**.

3.5 PivotTables

> **i** AI prompt to check your work
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"I built a PivotTable with `[fields]` in Rows/Columns and `[field]` summarised by `[Sum/Average/Count]`. Check whether that layout answers my question, whether Sum vs Average vs Count is the right summary, and what to verify (grand totals, blank cells, a manual spot-check, refreshing after data changes). Do not build a different pivot for me."*

A PivotTable is Excel's most underused feature for forestry data. It takes a **tidy** table — one row per observation, one column per variable — and rearranges it into a summary.

The Ontario file is tidy and cross-classified, which makes it perfect for a PivotTable.

3.5.1 What a PivotTable does

Given the 179-row Ontario file, a PivotTable can produce any of:

- A table of *total area* by forest type (one number per row).
- A table of *total area* by forest type × seral stage (rows × columns).
- A table of *average area* by forest type × seral stage.
- A *count* of how many age-class combinations exist for each forest type.

All without writing a single formula.

Table 3.1: Total area (ha) by forest type × seral stage — what your PivotTable should look like

PFTName	Immature	Late Successional	Mature	Pre-Sapling	Sapling
Conifer Lowland	426,314	1,193,510	903,009	240,759	229,741
Conifer Upland	655,387	474,772	745,944	289,253	700,537
Jack Pine	132,570	6,833	92,894	39,072	156,346
Mixedwood	493,741	153,687	471,449	79,606	145,449
Poplar	189,289	169,293	219,451	64,766	195,822
Red and White Pine	2,222	2,436	4,026	2,786	2,180
Tolerant Hardwoods	475	113	1,142	82	20
White Birch	105,663	93,154	242,519	31,927	37,802

3.5.2 Building one — step by step

1. Click any cell inside the Ontario `Data` sheet.
2. **Insert → PivotTable**. A dialog opens. (**Mac:** if you do not see it directly, use **Insert → Tables → PivotTable**.)
3. Choose **New Worksheet** as the destination. Click OK.
4. A blank PivotTable appears on a new sheet, with a **PivotTable Fields** panel on the right.
5. Drag fields into the four zones:
    - **Rows**: `PFTName`
    - **Columns**: `SERAL_NAME`
    - **Values**: `SumOfTotalHa`
    - **Filters**: (leave empty for now)
6. Excel automatically picks `Sum` as the aggregation. Confirm.
7. Format the numbers: right-click any value → **Number Format → Number → 1000 separator → 0 decimals**.

You should now see the cross-tab shown above.

3.5.3 Changing the aggregation

Right-click any value in the PivotTable → **Value Field Settings**. A list of aggregations appears. **Mac:** if right-click does not show it, click the small **information icon (i)** next to the field name in the **PivotTable Fields** panel to open the same settings.

Aggregation	What it returns	When to use
Sum	Total of the values	Default for most quantities.
Count	How many rows match	"How many combinations are there?"
Average	Mean of the values	"What is the typical row in this group?"
Max	Largest value	"Which group has the biggest single cell?"
StdDev	Variability around the mean	"How spread out are the values?"

The same PivotTable can answer multiple questions by switching the aggregation. Try it.

3.5.4 Drag-and-drop pivot

The most useful PivotTable habit is **moving fields around**:

- Drag `PFTName` from Rows to Columns. The whole table flips.
- Drag `AC_10` into Rows below `PFTName`. You get nested groups.
- Drag `SERAL_NAME` into Filters. A dropdown appears above the table — choose one seral stage to filter the whole pivot.

This is the *fluency* PivotTables train. Drag, observe, drag back.

3.5.5 Refreshing

A PivotTable is a **snapshot** of the data at the moment you built it. If the underlying Data sheet changes, the PivotTable does *not* automatically update.

To refresh: click anywhere in the PivotTable, then **PivotTable Analyze → Refresh** (or right-click → **Refresh**).

> ⚠ A silent-bug warning
>
> If you change the Data sheet and forget to refresh the PivotTable, the chart you export will be based on **stale data**. There is no warning. Always refresh before exporting.

3.6 PivotCharts

A PivotChart is a chart **wired** to a PivotTable. When you change the PivotTable, the chart changes with it.

To create one:

1. Click anywhere inside an existing PivotTable.
2. **PivotTable Analyze → PivotChart**.
3. Choose a chart type. For a cross-classification, a **clustered column** or **stacked column** chart usually works best.

The chart appears with field buttons on it. Those buttons are interactive — clicking them filters the chart (and the PivotTable behind it).

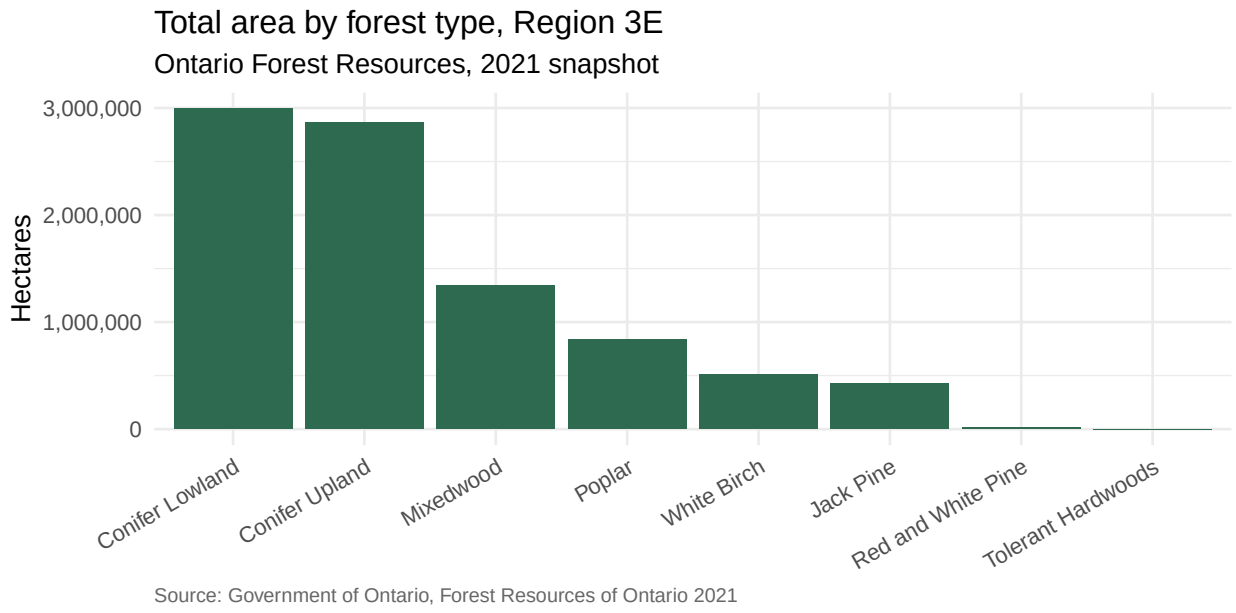

Figure 3.2: Preview of an Excel PivotChart: total area (ha) by Ontario forest type.

3.6.1 Slicers

A **Slicer** is a visual filter panel. Click any PivotTable → **PivotTable Analyze → Insert Slicer**. Check `PFTName`. Click OK.

A panel of buttons appears, one per forest type. Click "Jack Pine" — the PivotTable (and the PivotChart) filter to Jack Pine only. **Ctrl+Click** (**Cmd+Click** on Mac) to add more forest types, or **Shift+Click** to select a range.

Slicers are excellent for **presentations** because they let a non-technical reader explore the data without learning Excel.

3.7 Recreating a PivotTable with formulas

PivotTables are fast and visual, but they are not the only way to build a cross-tab. You can recreate one with `SUMIFS` and `COUNTIFS` — the formula approach.

This matters because **the formula version is the closest thing to what you will write in R**.

3.7.1 Example: SUMIFS cross-tab on the Ontario file

Build a small table on a new sheet:

	Pre-Sapling	Sapling	Immature	Mature	Late Successional
Jack Pine	`=SUMIFS(...)`	`=SUMIFS(...)`	…	…	…
Conifer	…	…	…	…	…
Upland Conifer	…	…	…	…	…
Lowland Mixedwood	…	…	…	…	…
Poplar	…	…	…	…	…
White Birch	…	…	…	…	…
Red and White Pine	…	…	…	…	…
Tolerant Hardwoods	…	…	…	…	…

Each cell is a `SUMIFS` formula like:

```
=SUMIFS(Data!$H:$H, Data!$F:$F, $A2, Data!$D:$D, B$1)
```

- `Data!$H:$H` is `SumOfTotalHa`.
- `Data!$F:$F` is `PFTName`.
- `Data!$D:$D` is `SERAL_NAME`.
- `$A2` is the row label (a forest type).
- `B$1` is the column label (a seral stage).

Notice the **mixed references**: `$A2` (column locked, row moves) and `B$1` (row locked, column moves). This is the cell-reference pattern you learned in Chapter 2 — it lets one formula copied across the whole grid produce the right answer in every cell.

The result is the same numbers as the PivotTable, but the formulas are explicit, auditable, and version-controllable.

When to use which

- For **exploration** — drag-and-drop PivotTable. Fast.
- For **production reports** that need to be reproducible — formula cross-tab with `SUMIFS`. Auditable.
- For **scripted analysis** — neither, use R (Chapter 7).

3.8 Exporting a chart with a caption

A chart in Excel is not a deliverable. A chart **outside** Excel — saved as a PNG with a caption — is a deliverable.

3.8.1 To save as PNG

Right-click the chart border → **Save as Picture**. Choose PNG. Save into your `outputs/` folder.

3.8.2 Caption format

A caption is one short sentence that:

- Tells the reader what the chart shows.
- Names the data source.
- Notes the date or version.

Example:

> **Figure 1.** BC reforestation has tracked total disturbance since 1999, with the exception of the 2017 wildfire spike. Source: Environmental Reporting BC, *Trends in Silviculture in B.C.*, retrieved 2026-05-24.

Put the caption in the report document, not in the PNG itself.

3.9 Connecting charts to the data dictionary

Every chart should be **traceable** back to the ReadMe sheet of the file it came from. Three habits make this easy:

1. **Name the chart** after the variable it shows. *"Total reforestation by year"* is better than *"Chart 3"*.
2. **Save the PNG** with a snake_case name that includes the variable. `reforestation_by_year.png` is better than `chart3.png`.
3. **In the caption**, reference the variable by its exact ReadMe name. *"`Reforestation_ha`"* is better than *"reforestation"*.

This sounds fussy but pays off the first time someone asks *"which column is this from?"*.

3.10 Active learning

3.10.1 Activity 1 — Choose the right chart

A list of forestry questions appears in the learner workbook. Match each one to its best chart type from this menu: line, bar, column, scatter, stacked column, pie. Defend your choices in pairs.

3.10.2 Activity 2 — Build the BC time-series chart

Open `bc_disturbance_reforestation_clean.xlsx` (your working copy). Select `Fiscal_Year`, `Total_Disturbance_ha`, and `Reforestation_ha`. **Insert → Charts → Line**. Add a title, axis labels, and a caption. Save as `outputs/bc_disturbance_reforestation_chart.png`.

3.10.3 Activity 3 — Build a PivotTable

Open `on_forest_statistics_1_clean.xlsx`. Insert a PivotTable with `PFTName` in Rows, `SERAL_NAME` in Columns, `SumOfTotalHa` in Values. Verify the totals match the table in the chapter prose above.

Table 3.2: Total area by forest type — should match the row totals of your PivotTable

PFTName	total_ha
Conifer Lowland	2,993,333
Conifer Upland	2,865,892
Mixedwood	1,343,933
Poplar	838,621
White Birch	511,064
Jack Pine	427,715
Red and White Pine	13,649
Tolerant Hardwoods	1,832

3.10.4 Activity 4 — Switch the aggregation

Right-click any value in your PivotTable → **Value Field Settings → Count**. The cell values change from total hectares to *number of age-class combinations* for that group. How does the picture change?

Table 3.3: Number of age-class combinations per forest type

PFTName	n
Conifer Upland	26
Poplar	25
White Birch	25
Conifer Lowland	23

PFTName	n
Mixedwood	23
Jack Pine	22
Red and White Pine	22
Tolerant Hardwoods	13

3.10.5 Activity 5 — Add a slicer

To your PivotTable, add a Slicer on `LGDS` (the seral-stage code). Click each button in turn. Which forest types disappear at the "Pre-Sapling" stage? Which dominate at "Mature"?

3.11 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.
2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?
4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

Lab: Build a chart-and-PivotTable deliverable together

Work in groups of 3 or 4. One submission per group.
Task 1 — Choose the question. As a group, choose one forestry question your PivotTable will answer. Options:

- *"Which forest type has the most mature area in Region 3E?"*
- *"Which age class has the most total area, across all types?"*
- *"How does conifer-vs-hardwood area compare across seral stages?"*

Task 2 — Build the PivotTable. Open `on_forest_statistics_1_clean.xlsx`. Insert a PivotTable that answers your question. Decide together what goes in Rows, Columns, and Values.
Task 3 — Build the PivotChart. From your PivotTable, insert a PivotChart of an appropriate type. Add a title, axis labels, and ensure units (hectares) appear in the y-axis label.
Task 4 — Add a slicer. Insert at least one slicer (any categorical field). Confirm that clicking a slicer button changes both the PivotTable and the PivotChart.
Task 5 — Export and caption. Save the PivotChart as a PNG into `outputs/`. Write a one-sentence caption that includes the data source.
Task 6 — Submit. Submit a short worksheet showing your question, your field choices, and the exported PNG + caption. Include all group members' names.
This lab is designed to fit comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 3 reference solution (HTML answer key) — the completed, task-by-task answer key — or **download the completed Excel workbook** with the formulas worked into the sheets (it recalculates when you open it in Excel). Open it **after** your own attempt and use it to check your work.
*This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

3.12 Exercises

These are the take-home exercises that go with this chapter.

1. Open `bc_disturbance_reforestation.xlsx`. Build a **column chart** of `Natural_Disturbance_ha` by `Fiscal_Year`. Add a title that names the 2017 wildfire spike. Export as PNG.
2. Open `on_forest_statistics_1.xlsx`. Build a PivotTable with `SERAL_NAME` in Rows and `SumOfTotalHa` in Values (using `Sum`). Sort the table by descending total area. Which seral stage covers the most area?
3. On the same PivotTable, change the aggregation from `Sum` to `Count`. Which seral stage has the most age-class combinations? Is it the same answer as Exercise 2?

4. Build a **manual SUMIFS cross-tab** with `PFTName` down the left and `SERAL_NAME` across the top. Verify your numbers match the PivotTable.
5. Add a **slicer** on `PFT` to your PivotTable. Take a screenshot showing the table when one forest-type slicer is active.
6. Write a one-page chart-caption sheet listing the captions you would use for the three exported PNG files (Exercises 1, 2, and the group lab output). Save as `outputs/chart_captions.docx` or `chart_captions.md`.
7. **Optional, harder.** Build a **dashboard sheet** containing your time-series chart from Exercise 1 and your PivotChart from Exercise 2 side by side, with a slicer that controls both. (Hint: a slicer can connect to multiple PivotTables built from the same data source.)

3.13 Optional, advanced

3.13.1 Sparklines

A sparkline is a tiny chart inside a single cell. Useful for trend-at-a-glance summaries inside a table.

To insert: **Insert → Sparklines → Line** (or Column, or Win/Loss). Select the data range, then choose the cell where the sparkline goes.

3.13.2 Conditional-formatted data bars

Inside a PivotTable, you can add **Data Bars** to a value column. Select the values → **Home → Conditional Formatting → Data Bars**. The cells now show a tiny in-cell horizontal bar proportional to the value — a quick way to add visual rank to a table.

3.13.3 Calculated fields

A **Calculated Field** is a column in a PivotTable derived from other PivotTable fields. To add one: **PivotTable Analyze → Fields, Items, & Sets → Calculated Field**.

Example: in the Ontario file you could add a `area_per_age_class` calculated field equal to `SumOfTotalHa / AC_10`. Use sparingly — calculated fields are powerful but easy to hide a typo inside.

3.14 A glimpse ahead: From Excel to `ggplot2`

In Chapter 9 you will rebuild the BC and Ontario charts from this chapter using `ggplot2` in R. The Excel moves you learned here map to R commands you will meet later:

What you did in Excel (Ch 3)	What you will do in R (Ch 9)
Insert → Line chart	`ggplot(bc, aes(Fiscal_Year, Reforestation_ha)) + geom_line()`
Insert → Column chart	`ggplot(...) + geom_col()`
Chart title in the chart area	`labs(title = "...")`
Axis title "Hectares"	`labs(y = "Hectares")`
Add caption in a text box	`labs(caption = "Source: ...")`
PivotTable with `PFTName` in Rows	`ontario |> group_by(PFTName) |> summarise(sum(SumOfTotalHa))`
PivotTable with `PFTName` × `SERAL_NAME`	`... |> group_by(PFTName, SERAL_NAME) |> summarise(...) |> pivot_wider(...)`
Slicer on `PFTName`	`... |> filter(PFTName == "Jack Pine")`
Refresh PivotTable (manual)	Automatic — re-running the code re-computes everything
PNG export via Save as Picture	`ggsave("outputs/bc_reforestation.png", width = 7, height = 4)`

The intent is identical. R is scriptable; Excel is visual. Chapter 9 will walk through each row of this table in detail. The point right now is that the chart vocabulary you learn here transfers directly.

3.15 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

3.16 AI as a debugging companion

Useful prompts for this chapter

- *"In Excel, my PivotTable shows only zeros for one row. The underlying Data sheet has values. What are the three most common reasons a PivotTable shows zeros?"*
- *"Explain step by step how to add a slicer to an existing PivotTable and link it to a second PivotTable on the same sheet."*
- *"I want to recreate this PivotTable cell with a formula — sum of `SumOfTotalHa` for `PFTName = 'Jack Pine'` and `SERAL_NAME = 'Mature'`. What is the SUMIFS formula?"*

3.16.1 Verifying AI's answers

After AI suggests a chart type or PivotTable layout, **check**:

1. Does the chart actually answer the question you asked?

2. Does the title state a one-sentence claim?
3. Do the units appear in the axis label?
4. Did the AI refresh the PivotTable in its instructions? (It often forgets.)

AI's chart suggestions are starting points. The interpretation is yours.

3.16.2 When *not* to use AI

- Do not let AI choose the chart type without you stating the question. The chart serves the question; the question serves you.
- Do not let AI invent a caption that names a source you have not verified.

3.17 Reading

- Microsoft Excel official documentation on PivotTables.
- Microsoft Excel official documentation on PivotCharts.
- Cole Nussbaumer Knaflic, *Storytelling with Data* (chapter on chart choice — short, opinionated, practical).
- BC Government silviculture indicator background: https://www.env.gov.bc.ca/soe/indicators/land/silviculture.html
- Ontario open-data dataset page: https://data.ontario.ca/dataset/forest-resources-of-ontario-2021

Instructor notes

Pacing and emphasis

- This chapter is the **bridge chapter** from raw Excel work into tidy-data thinking. Spend time on the PivotTable → R parallel in the "A glimpse ahead" section — it is the most important conceptual move of the first half of the book.
- The main path covers six skills: chart choice, chart anatomy, PivotTable construction, aggregation switching, PivotChart, and Slicers. The SUMIFS cross-tab section is essential for the Excel-to-R bridge.
- **Sparklines**, **data bars in pivots**, and **calculated fields** are moved to the *Optional, advanced* section. Mention them only if your group is ahead of pace.
- The group lab is the central hands-on deliverable. Plan for one full session.

3.17.1 Expected output checklist (matches the top-of-chapter callout)

1. One PivotTable + PivotChart on `on_forest_statistics_1.xlsx`, exported as PNG with caption.
2. One time-series chart on `bc_disturbance_reforestation.xlsx`, exported as PNG with caption.
3. A short chart-choice worksheet (from the learner workbook).

4. Group-lab worksheet or screenshot.

3.17.2 Materials provided alongside this chapter

File	Purpose
`bc_disturbance_reforestation.xlsx`	Raw BC silviculture workbook (carried forward from Chapter 1).
`on_forest_statistics_1.xlsx`	Raw Ontario forest stats workbook (carried forward from Chapter 2).
`frst232_ch03_pivot_learner.xlsx`	Learner-facing workbook with both datasets and chapter exercises.
`frst232_ch03_pivot_solutions.xlsx`	Instructor solutions. **Do not distribute to learners.**
`quiz_ch03.html`	Standalone interactive quiz. Embedded in the rendered chapter; can also be hosted separately.

3.17.3 Suggested facilitation guidance for the Practice Demo Lab

*These are optional facilitation and feedback suggestions for instructors running this book demo — **not** a grading key. The percentages below are relative emphasis, not Canvas marks; the graded lab assignment is on Canvas.*

Task	Weight	What to look for
Question + PivotTable (Tasks 1–2)	25 %	Question is specific and answerable; PivotTable layout matches the question.
PivotChart + axis labels (Task 3)	25 %	Chart type matches the question; axis labels include units.
Slicer (Task 4)	15 %	Slicer present and functional.
PNG export + caption (Task 5)	25 %	PNG file submitted; one-sentence caption includes source.
Submission (Task 6)	10 %	Worksheet is legible; group members named.

3.17.4 Common stumbling points

- Some learners build a PivotTable, change the Data sheet, and then export from the stale PivotTable. Emphasize the **Refresh before export** habit.
- Some forget that a chart title and an axis label are different things. A good chart has both.
- A few will choose a pie chart for 8-category comparisons. Redirect to a sorted bar chart.
- Some will compute "average" on a column already summed in a PivotTable, double-aggregating. Walk through what `Sum` vs `Average` actually does at the cell level.
- *Excel-to-R bridge:* The "A glimpse ahead" section is the single most important paragraph in Chapter 3 for the long arc of the course. Refer to it explicitly when learners ask why we are still in Excel.

Guided Case Study: From Excel to R

! Practice demo only

This activity is a **guided practice lab in the OER book**. It is designed to help you learn the workflow before completing the official lab assignment on Canvas. **The Canvas lab is the graded assignment.** Use this book activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

i **Data for this chapter**

This chapter uses the **BC silviculture** workbook (`bc_disturbance_reforestation.xlsx`) — the same file you used in the Excel chapters. Keep it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

What this chapter is

This is your **first, guided case study** — an early bridge from Excel (Chapters 1–3) to R (Chapters 4–5). You will answer one small forestry question **twice**: once with the Excel skills you already have, and once with the first R tools. Seeing the *same answer* on both sides is the point.

It deliberately uses **only the skills taught so far**: Excel inspection, cleaning, formulas, PivotTables, and charting, plus early R setup, import, inspection, basic functions, the native pipe `|>`, and Quarto. It does **not** need joins, reshaping, grouped `summarise()`, or `ggplot2` — those come later, and the full **independent** case study at the end of the course (Chapter Integrated Case Study) uses them all.

Chapter goals

By the end you will be able to:

- restate a small forestry question and decide what you need to answer it;
- answer it in **Excel** (inspect → formula / PivotTable → chart);
- answer the same question in **early R** (import → inspect → a basic calculation with the native pipe) and **confirm the two agree**;
- verify a result instead of trusting it, and write an AI prompt that *checks* your work rather than doing it for you.

The question

> *"What was the average annual reforested area in British Columbia over the whole record, and which single year was the highest?"*

Nothing here needs a join or a grouped summary — just an average, a maximum, and careful reading of the file.

Part A — answer it in Excel

You already know these moves from Chapters 1–3:

1. **Inspect.** Open the **Data** sheet. Confirm the columns (`Fiscal_Year`, `Reforestation_ha`, …), the units (hectares), and that there are no missing values.
2. **Average.** In an empty cell, `=AVERAGE(H2:H38)` on the `Reforestation_ha` column.
3. **Maximum + its year.** `=MAX(H2:H38)` for the largest value, and `=INDEX(A2:A38, MATCH(MAX(H2:H38), H2:H38, 0))` for the fiscal year it happened in.
4. **Chart (optional).** A quick line chart of `Reforestation_ha` by `Fiscal_Year` shows the shape behind the numbers.

Write your three answers down — you will check them against R next.

Part B — answer it in early R

> **AI prompt to check your work**
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"My question is `[question]`, and I plan to answer it with these steps: `[list your Excel and early-R steps]`. Check whether my plan uses the simplest tools that answer the question, whether the order makes sense, and which checks would confirm the result. Do not solve the case study for me."*

The same three answers, now reproducibly. This uses only Chapter 4–5 skills: `read_excel()`, inspection, base functions, and the pipe.

```
  bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                   sheet = "Data")
  dim(bc)         # rows and columns
```

```
[1] 37  8
```

```
  names(bc)       # column names - confirm Reforestation_ha is here
```

```
[1] "Fiscal_Year"                   "Clearcutting_ha"
[3] "Clearcutting_with_reserves_ha" "Partial_cutting_ha"
[5] "Harvested_ha"                  "Natural_Disturbance_ha"
[7] "Total_Disturbance_ha"          "Reforestation_ha"
```

```
  # Average reforested area (same as Excel =AVERAGE)
  bc$Reforestation_ha |> mean()
```

```
[1] 223353.1
```

```
  # The maximum, and the fiscal year it happened in
  bc$Reforestation_ha |> max()
```

```
[1] 274616.1
```

```r
bc$Fiscal_Year[which.max(bc$Reforestation_ha)]
```

```
[1] 2006
```

The pipe `|>` reads *"take the* ***`Reforestation_ha`*** *column, and then take its mean."* No new packages, no `group_by()` — just the moves you already have.

The bridge: the R average should match your Excel `=AVERAGE` to the cent, and the R maximum and its year should match your Excel `MAX` / `INDEX/MATCH`. If they *don't*, something is off in one of the two — that mismatch is a verification win, not a failure.

Practice Demo Lab

> **!** Practice demo only (not the graded Canvas lab)
>
> Work through the six steps below. The graded version is on Canvas.

1 — Your attempt. For a *different* column — **`Harvested_ha`** — compute in R: (a) the average harvested area, (b) the maximum, and (c) the fiscal year of the maximum. Then confirm each against an Excel formula.

2 — Reference solution.

```
  bc$Harvested_ha |> mean()                              # (a) average
```

```
[1] 199053.3
```

```
  bc$Harvested_ha |> max()                               # (b) maximum
```

```
[1] 251557.3
```

```
  bc$Fiscal_Year[which.max(bc$Harvested_ha)]             # (c) year of max
```

```
[1] 1988
```

(In Excel: `=AVERAGE(E2:E38)`*,* `=MAX(E2:E38)`*, and* `=INDEX(A2:A38, MATCH(MAX(E2:E38), E2:E38, 0))`*.)*

3 — Compare your work with the reference solution. Where did your numbers match? Where did they differ? What caused any difference (wrong column? a stray blank cell in the Excel range? the wrong sheet)? How did you fix it?

4 — Write your own AI verification prompt. Draft a prompt that asks an AI to *check* your reasoning and outputs — not to hand you the answer. Include the file name, the column, what you computed, and the checks you want it to confirm.

5 — Model AI verification prompt.

> *"I am using R in FRST 232. I have a tibble* `bc` *from* `bc_disturbance_reforestation.xlsx` *(Data sheet), columns including* `Fiscal_Year` *and* `Harvested_ha`*. I computed the mean, the max, and the year of the max of* `Harvested_ha`*. Here is my code and output:* `[paste]`*. Please explain what each line does, tell me whether it correctly answers the question, and list checks I can run to confirm it (for example, comparing to an Excel* `=AVERAGE/=MAX`*, and checking for blank cells). Do not give me a final assignment answer — help me verify my own."*

6 — Reflection: what changed after checking? In two or three sentences: did Excel and R agree the first time? If not, what was wrong? What will you check *first* next time?

Done

You have answered one forestry question in two tools and **verified they agree**. That habit — compute, then confirm — is the foundation of every later chapter, and of the full independent case study at the end of the course.

Part III

R foundations

4 Start of Computing in R: Install, Project, First Import

> **Data for this chapter**
>
> This chapter uses the **BC silviculture** (`bc_disturbance_reforestation.xlsx`) and **Ontario forest statistics** (`on_forest_statistics_1.xlsx`) workbooks. Click each file name to download it directly, then save it in your `data/` folder. For sources and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Explain the difference between **R**, **RStudio**, **Quarto**, and **Posit Cloud**, and choose which you will use.
- Install R and RStudio on your computer, **or** create a free Posit Cloud account (recommended for getting started).
- Create an **RStudio Project** for FRST 232 with a clean folder structure (`data/`, `outputs/`, `R/`, `notes/`).
- Install and load the **`tidyverse`**, **`readxl`**, and **`here`** packages.
- Use the **`here()`** function for portable, project-relative file paths.
- **Import** the BC silviculture file from Chapter 1 into R using `readxl::read_excel()`.
- Inspect the imported **tibble** (R's tidyverse data table — introduced fully in Chapter 5) with `dim()`, `nrow()`, `ncol()`, `names()`, `glimpse()`, `summary()`, and `head()`.
- **Confirm** the numbers R computes match the numbers you computed in Excel in Chapters 1–3.
- Use the **native pipe `|>`** to chain operations.
- Recognize and read common error messages.

Expected output for this chapter

What you will hand in

1. A **working RStudio Project** named `frst232/` (or `frst232-yourname/`), either on your local machine or on Posit Cloud, with this folder structure:

```
frst232/
  frst232.Rproj
  data/bc_disturbance_reforestation.xlsx
  outputs/
  R/
  notes/
```

2. A rendered Quarto HTML file (`frst232_ch04_learner.html`) showing the result of importing and inspecting the BC silviculture file. The file should include:
   - the working directory printed by `here::here()`,
   - `dim(bc)`, `names(bc)`, and `glimpse(bc)` output,
   - the five basic statistics (`sum`, `mean`, `min`, `max`, `median`) of `bc$Reforestation_ha`,
   - a one-line confirmation that **the R numbers match the Excel numbers** from Chapter 1.
3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

4.1 Why R, why now

For three chapters, every analysis lived inside Excel. You wrote formulas, dragged them across grids, built PivotTables, made charts. Real work.

But you also saw the limits:

- **PivotTable refresh.** Change the Data sheet → forget to refresh → silent stale data.
- **Cell-reference shifts.** Copy a formula one cell over → forget the `$` → silently wrong percentages.
- **Display vs value.** Two cells show `100,000` → one is `99999.999`. `=A1=B1` is `FALSE` and nobody notices.
- **No history.** Six months later, *"how did I get this number?"* is unanswerable.

R fixes these — not by being smarter, but by making every step **a written instruction**. The instruction is the audit trail. Run it again, get the same number. Send it to a colleague, get the same number. Edit one line, re-run, see what changed.

Chapter 4 is where the language changes. The data does not. The file you import today is the same `bc_disturbance_reforestation.xlsx` from Chapter 1. The numbers you compute will be identical. **What changes is that the steps are now scriptable.**

4.2 R, RStudio, Quarto, Posit Cloud — what each thing is

These four words show up everywhere. They are not the same thing.

Name	What it is	When you need it
R	The computing language and engine. The thing that does the math.	Always. R is the underlying tool.
RStudio	A friendly editor for R, with a console, file browser, and project manager.	Almost always. You *could* use R without it, but you do not want to.
Quarto	A document format that lets you mix R code, R output, and prose into one rendered HTML / PDF / Word file.	When you want to write a report (Chapter 5 onward).
Posit Cloud	A free, browser-based version of RStudio that runs in the cloud — no install needed.	When you do not want to (or cannot) install R locally.

The recommended path for this course

If you have never used R before, **start with Posit Cloud**. No installation. No version conflicts. Works on any laptop, any operating system, any course-lab computer.
Switch to a local install only if (a) you need to work offline or (b) you are comfortable troubleshooting R installation issues yourself.

4.3 Path A — Posit Cloud (recommended)

1. Go to https://posit.cloud.
2. Click **Sign Up** (free plan is fine for this course).
3. Once signed in, click **New Project → New RStudio Project**.
4. Rename the project `frst232` (top-left of the window).
5. You now have a full RStudio environment in your browser.

That **New RStudio Project** is the project you will use all term. Skip **Path B** (it is only for local installs) and continue reading below — the next sections (*Creating an RStudio Project*, *Folder structure*, *Installing packages*) still apply to you.

4.4 Path B — Install R and RStudio locally

If you prefer (or need) a local install:

4.4.1 Install R

- Windows: https://cran.r-project.org/bin/windows/base/
- macOS: https://cran.r-project.org/bin/macosx/
- Pick the latest release. Accept all defaults.

4.4.2 Install RStudio

- All platforms: https://posit.co/download/rstudio-desktop/
- Install **after** R is installed. RStudio finds R automatically.

4.4.3 Install Quarto

- https://quarto.org/docs/get-started/
- Quarto often comes bundled with recent RStudio; if so, you can skip this step.

4.4.4 Verify

Open RStudio. In the **Console** pane (bottom-left), type:

```
R.version.string
```

You should see something like `"R version 4.5.0 (2025-04-11)"`. If you do, you are ready.

4.5 Creating an RStudio Project

Whether on Posit Cloud or local, **always work inside an RStudio Project**. Never run R code without a project.

A project is just a folder with a `.Rproj` file inside it. The `.Rproj` file tells RStudio:

- where your working directory is,
- which files belong to this project,
- how to keep your R session separate from other projects.

> **i** Posit Cloud users: you already created your project
>
> If you followed **Path A**, the **New RStudio Project** you made in Posit Cloud (Path A, step 3) *is* your course project — you do not need to create another one. Read this section for the *why*, then continue at *Folder structure*. The **To create the project** steps just below are

written for a **local (Path B)** install.

4.5.1 To create the project

In RStudio: **File → New Project → New Directory → New Project**.

- **Directory name**: `frst232`
- **Create project as subdirectory of**: pick a sensible parent folder (e.g., your `Documents/` folder, or — on Posit Cloud — the default).
- Click **Create Project**.

RStudio opens a fresh session inside the new folder. The top-right corner of the window now reads `frst232`.

4.5.2 Why this matters

Without a project, your R session has **no anchor**. You end up typing absolute paths like `/Users/yourname/Desktop/FRST232/Lab1/data.csv` — which break the moment you move the folder, switch computers, or share with a classmate.

With a project, you type **relative paths** like `data/bc_disturbance_reforestation.xlsx`. R knows the project root and resolves the rest. Portable forever.

4.6 Folder structure (same as Chapters 1–3, now used by R)

Inside your `frst232` project folder, create these subfolders. In RStudio, use the **Files** pane (bottom-right) → **New Folder**.

```
frst232/
  frst232.Rproj                              ← created automatically
  data/                                      ← raw input files
     bc_disturbance_reforestation.xlsx
  outputs/                                   ← cleaned files, figures
  R/                                         ← reusable R scripts (.R files)
  notes/                                     ← scribbles, screenshots
  deliverables/                              ← what you submit
```

Same three rules from Chapter 1:

1. **Never edit raw data in place.** `data/` is read-only.
2. **Use lowercase, snake_case names.** No spaces, no capitals.
3. **One folder per chapter** (or per topic). Easier to find things later.

4.6.1 Place the BC file

If you do not have it already, download `bc_disturbance_reforestation.xlsx` from the BC Data Catalogue (https://catalogue.data.gov.bc.ca/dataset/indicator-summary-data-trends-in-silviculture-in-b-c-) and drop it into `data/`.

On Posit Cloud, you upload files through the **Files pane → Upload** button.

4.7 Installing packages

R on its own can do a lot, but most real work uses **packages** — pre-written collections of functions. You install a package once, then load it at the start of every R session that needs it.

The three packages we use this chapter:

Package	What it does
`tidyverse`	A bundle of packages for data manipulation, plotting, and reshaping. The backbone of modern R.
`readxl`	Reads `.xlsx` and `.xls` files.
`here`	Builds portable, project-relative file paths.

4.7.1 To install

In the RStudio Console, type:

```
install.packages(c("tidyverse", "readxl", "here"))
```

Press Enter. This will take a minute or two the first time — R downloads each package and its dependencies.

> ⚠ install vs library
>
> - `install.packages(...)` — runs **once per machine**. Downloads and installs.
> - `library(...)` — runs **once per R session**. Loads the installed package so its functions are available.
>
> Confusing these two is the most common first-day R bug. *"I installed it but R says it can't find the function"* — almost always means you forgot the `library()` call.

4.7.2 To load

At the top of every R script or Quarto document for this course, include:

```
library(tidyverse)
library(readxl)
library(here)
```

4.8 `here()` — portable paths

The `here()` function builds a path relative to your project root. Type this in the Console:

```
here("data", "bc_disturbance_reforestation.xlsx")
```

You should see a string like:

```
/cloud/project/data/bc_disturbance_reforestation.xlsx
```

(or, on a local install, something like `/Users/yourname/Documents/frst232/data/bc_disturbance_reforestati`

The point: **the same code works for everyone**. You do not need to know the full path; `here()` figures it out.

> Why this matters for collaboration
>
> When you share your project folder with a classmate (or with a TA for grading), they may unzip it to a completely different location. **Because every file path uses `here()`, the code still works** — no edits required.
> This is the reproducibility win that pure-Excel work cannot give you.

4.9 Your first R commands

Open a new R script: **File → New File → R Script**. Save it into `R/` as `01_first_import.R`.

Type these lines (or copy-paste). Run them one at a time by clicking on the line and pressing **Ctrl+Enter** (Cmd+Enter on Mac):

```
# Load packages - run these once at the start of every R session
library(tidyverse)   # dplyr, ggplot2, readr, and friends
library(readxl)      # read_excel() for .xlsx files
library(here)        # here() builds project-relative paths
library(knitr)       # kable() for tidy tables (used later this chapter)
library(scales)      # label_comma() and friends for number formatting

# Import the BC silviculture file (same one you used in Chapter 1)
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")
```

```r
# Inspect
dim(bc)
names(bc)
glimpse(bc)
```

What you should see:

```
dim(bc): 37 8
```

```
names(bc):
```

```
[1] "Fiscal_Year"                    "Clearcutting_ha"
[3] "Clearcutting_with_reserves_ha" "Partial_cutting_ha"
[5] "Harvested_ha"                   "Natural_Disturbance_ha"
[7] "Total_Disturbance_ha"           "Reforestation_ha"
```

```
glimpse(bc):
```

```
Rows: 37
Columns: 8
$ Fiscal_Year                  <dbl> 1987, 1988, 1989, 1990, 1991, 1992, 1993~
$ Clearcutting_ha              <dbl> 216304.92, 217836.07, 190635.52, 172360.~
$ Clearcutting_with_reserves_ha <dbl> 5800.400, 1842.952, 1456.666, 1400.013, ~
$ Partial_cutting_ha           <dbl> 20076.681, 31878.294, 23785.954, 18930.8~
$ Harvested_ha                 <dbl> 242182.0, 251557.3, 215878.1, 192691.6, ~
$ Natural_Disturbance_ha       <dbl> 20875.00, 9641.20, 10711.80, 28300.70, 1~
$ Total_Disturbance_ha         <dbl> 263057.0, 261198.5, 226589.9, 220992.3, ~
$ Reforestation_ha             <dbl> 201972.5, 221296.3, 237275.9, 260370.8, ~
```

37 rows × 8 columns — exactly the BC file you knew from Chapter 1, now living inside R.

> **!** Load your packages at the start of every session
>
> `library()` loads a package for the **current session only**. Run all five `library()` lines above before anything else each time you open RStudio or Posit Cloud. In particular, `library(tidyverse)` does **not** load `knitr` or `scales` — you need those two separate lines, or later examples (`kable()` in this chapter, `label_comma()` in Chapter 9) fail with *"could not find function"*.

4.9.1 Line by line — what each command did

Several new ideas appear at once above. Here is what each line does:

Code	What it does
`library(tidyverse)`	Loads the common data-analysis tools (dplyr, ggplot2, …).
`library(readxl)`	Loads the package that can read Excel files.
`library(here)`	Loads `here()`, which builds paths from the project root.
`here("data", "…xlsx")`	Builds the path from your project folder to the data file.
`read_excel(…, sheet = "Data")`	Reads the **Data** sheet (not About or ReadMe) into R.
`bc <- …`	Stores the result as a tibble named `bc`.
`dim(bc)`	Number of rows and columns.
`names(bc)`	The column names.
`glimpse(bc)`	A compact preview of every column's type and first values.

⚠ Keep the workbook as `.xlsx` — don't export it to CSV

If you open the workbook in Excel and use **File → Export / Save As → CSV**, Excel writes **only the active sheet** — you can end up with `bc_disturbance_reforestation(About).csv` containing just the About tab. The import needs the **Data** sheet, so keep the original `.xlsx` and read it with `read_excel(…, sheet = "Data")`.

4.9.2 The assignment operator <-

You probably noticed the `<-` symbol. It is the **assignment operator** — *"store this on the right into this name on the left"*. In R, `<-` is preferred over `=` for assignment because `=` has other uses (function arguments).

A keyboard shortcut: **Alt+-** (Windows / Linux) or **Option+-** (Mac) inserts `<-` with a space.

4.10 The Excel-to-R bridge — side by side

The following table is what Chapters 1, 2, and 3 have been previewing. Now you can actually run the right-hand column.

What you did in Excel	What you do in R	Result on the BC file
Open the file by double-clicking	`bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"), sheet = "Data")`	A tibble in memory called `bc`
Read the status-bar `Count`	`nrow(bc)`	37

What you did in Excel	What you do in R	Result on the BC file
Read the status-bar `Sum` for column H	`sum(bc$Reforestation_ha)`	8,264,064
Read the status-bar `Average` for column G	`mean(bc$Total_Disturbance_ha)`	228,089
Scan the Data sheet visually	`glimpse(bc)` and `summary(bc)`	text summaries
`=MIN(F2:F38)`	`min(bc$Natural_Disturbance_ha)`	1,448
`=MAX(F2:F38)`	`max(bc$Natural_Disturbance_ha)`	218,275

Every number on the right matches the value you saw in Chapter 1. The data is the same. The columns are the same. The only thing that changed is *how you ask for the answer*.

4.11 Accessing columns with $

`bc$Reforestation_ha` means *"the column called Reforestation_ha inside the tibble bc"*. The `$` is the column selector.

Try these in the Console:

```
bc$Fiscal_Year                  # the year column (37 numbers)
bc$Reforestation_ha             # the reforestation column
sum(bc$Reforestation_ha)        # total reforestation
mean(bc$Reforestation_ha)       # mean annual reforestation
length(bc$Reforestation_ha)     # how many values (= 37)
```

4.11.1 Tab completion saves typos

In RStudio, after you type `bc$`, press **Tab**. A dropdown shows every column name. Click one (or arrow-down + Tab) to insert it. This prevents the *"I spelled the column wrong"* class of bugs.

4.12 Computing the chapter's five statistics

```
sum(bc$Reforestation_ha)
mean(bc$Reforestation_ha)
min(bc$Reforestation_ha)
max(bc$Reforestation_ha)
median(bc$Reforestation_ha)
```

Statistic	Result
Total reforestation	8,264,064 ha

Statistic	Result
Mean annual reforestation	223,353 ha
Min annual reforestation	189,394 ha
Max annual reforestation	274,616 ha
Median annual reforestation	219,925 ha

Match against Chapter 1. Open `bc_silviculture_ch01_workbook.xlsx` from your Chapter 1 deliverables and compare. Total reforestation should match to the last decimal. If anything does not match, something is wrong — either with your import or with the file you handed in for Chapter 1.

4.13 The native pipe |>

The pipe operator `|>` takes the thing on its left and sends it into the first argument of the function on its right. It reads left-to-right, like a sentence.

Without the pipe:

```
round(mean(bc$Reforestation_ha), 0)
```

With the pipe:

```
bc$Reforestation_ha |> mean() |> round(0)
```

Same answer. The piped version reads top to bottom: *"take the reforestation column → take its mean → round to zero decimals"*.

This is the same logical pattern as **chaining slicers in PivotTables**: a sequence of operations applied one after the other. The pipe is how that sequence looks in R.

When pipes help, when they hurt

- Pipes **help** when you have 3 or more steps that read naturally as a sequence. `bc |> filter() |> group_by() |> summarise()`.
- Pipes **hurt** when there is only one step. Just write `mean(bc$Reforestation_ha)` directly.

Use the pipe when it makes the code easier to read. Skip it when it does not.

4.14 Reading error messages

Errors in R look intimidating. They are not.

A typical first-day error — **this example is intentional**. We misspelled the column name (`Refrestation_ha`, missing the second "o") on purpose so you can see what R prints. You do **not** need to type or run it:

```
> sum(bc$Refrestation_ha)
Error in `$.tbl_df`(bc, Refrestation_ha) :
  Column `Refrestation_ha` doesn't exist.
Did you mean `Reforestation_ha`?
```

Two facts from that message:

1. The error happens in `$` — that is the column-selector.
2. R suggests a fix — *"Did you mean `Reforestation_ha`?"*. You misspelled it.

R errors usually tell you **where** and **why**. Read every word. The fix is often in the message itself.

4.14.1 A second common error

```
> library(tidyverse)
Error in library(tidyverse) : there is no package called 'tidyverse'
```

R is saying *"you tried to load a package you haven't installed"*. The fix is `install.packages("tidyverse")`, then try again.

> How to read any R error in 30 seconds
>
> 1. **Read the function name** in the error (e.g., `$`, `library`, `read_excel`). That tells you where the problem is.
> 2. **Read the suggested fix** if there is one (R is good at this).
> 3. **Copy-paste the error** into your search engine if you cannot solve it. Someone else has had the same problem.
> 4. **Ask AI** with the full error text *and* the code that caused it. Vague *"my code doesn't work"* prompts give vague answers.

4.15 Active learning

4.15.1 Activity 1 — Spot-check the install

In the Console, type:

```
R.version.string
sessionInfo()
```

The first line should report your R version (4.5 or newer). The second should list `tidyverse`, `readxl`, and `here` among the loaded packages (after you have run `library(...)` on each).

4.15.2 Activity 2 — Verify your folder structure

In the Console, type:

```
list.files(here("data"))
```

You should see `"bc_disturbance_reforestation.xlsx"`. If you see an empty result, the file is not where you think it is.

4.15.3 Activity 3 — Import and inspect

Run the import code from the *Your first R commands* section. Then answer in your own words:

- What is the *type* of `bc`? (Hint: `class(bc)`)
- How many rows does it have?
- What are the column names?

4.15.4 Activity 4 — Compute and compare

Compute the five statistics from this chapter. Compare each value to the same statistic in your Chapter 1 deliverable workbook. Do they match?

4.15.5 Activity 5 — Write your first pipe

Take this sequence:

```
round(mean(bc$Total_Disturbance_ha), 0)
```

Rewrite it as a pipe with three steps. Confirm the answer is the same.

4.16 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.

2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?
4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

i Lab: Set up a working R project together

Work in groups of 3 or 4. Each member follows the same setup steps on their own computer (or in their own Posit Cloud project). Use the lab session to debug together.
Task 1 — Posit Cloud accounts. Every member creates a free Posit Cloud account (if they have not already).
Task 2 — Create the project. Each member creates a project named `frst232` and sets up the folder structure from the *Folder structure* section.
Task 3 — Install packages. Each member runs:

```
install.packages(c("tidyverse", "readxl", "here"))
```

While packages install, discuss: *what is a package, and why are there so many?*
Task 4 — Upload the data file. Each member uploads `bc_disturbance_reforestation.xlsx` into `data/`. Confirm with `list.files(here("data"))`.
Task 5 — First import. Each member runs the import code from the chapter. Confirm everyone sees the same `dim(bc)` output (`37 8`).
Task 6 — Compute and compare. Each member computes `sum(bc$Reforestation_ha)`. Confirm everyone gets the same number (8,264,064). Now compare this to the Chapter 1 workbook value. **They should match.**
Task 7 — Submit. Submit a single screenshot showing one group member's RStudio with the BC file loaded, the column names visible, and the `sum()` result. Include all group members' names on the screenshot or in the submission caption.
This lab is designed to fit comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 4 reference solution (rendered HTML) — the completed, rendered solution for this demo lab. Open it **after** your own attempt and use it to check your work. *This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

4.17 Exercises

These are the take-home exercises that go with this chapter.

1. Compute `mean(bc$Harvested_ha)` and confirm it equals the same number you computed in Excel for Chapter 1 (Activity 3, the `=AVERAGE(...)` formula).
2. Use the pipe to compute the median of `Natural_Disturbance_ha`, rounded to the nearest hectare.
3. Use `which.max()` to find the **index** of the largest value in `bc$Natural_Disturbance_ha`. Then use it to find the year of the maximum:

   ```
   bc$Fiscal_Year[which.max(bc$Natural_Disturbance_ha)]
   ```

   What year does R report? Does it match Chapter 1?
4. Use `summary(bc)` and explain in one sentence what each row of the summary shows (Min, 1st Qu, Median, Mean, 3rd Qu, Max).
5. Save your `R/01_first_import.R` script. Close the project. Re-open it. Run the whole script again from top to bottom (**Code → Run Region → Run All**). Confirm everything works without errors.
6. Open the Ontario file from Chapter 2 the same way:

   ```
   ontario <- read_excel(here("data", "on_forest_statistics_1.xlsx"),
                         sheet = "Data")
   dim(ontario)
   ```

   How many rows does R report? Does it match Chapter 2?
7. **Optional, harder.** Write a tiny R function that takes a numeric vector and returns a named list with `mean`, `median`, `min`, and `max`. Apply it to `bc$Reforestation_ha`.

4.18 Optional, advanced

4.18.1 The old pipe %>%

You may see `%>%` (the magrittr pipe) in older R code. It does almost the same thing as the native `|>` but is not built into base R. This course uses the native `|>` because it is now part of R itself (since R 4.1) and does not require a package.

If you see `%>%` in a tutorial, treat it as equivalent to `|>` for most purposes. The differences are minor and only matter in advanced cases.

4.18.2 RStudio Project options

In the project menu (top-right of RStudio), choose **Project Options → General**. Recommended settings for this course:

- **Restore .RData into workspace on startup**: **No**
- **Save workspace to .RData on exit**: **Never**

These two changes guarantee that your R session starts fresh every time. This sounds inconvenient but is the right discipline: if your code only works because of *leftover variables from yesterday*, your code is broken.

4.18.3 Reading multiple sheets at once

`readxl::excel_sheets()` returns the sheet names of a file:

```
excel_sheets(here("data", "bc_disturbance_reforestation.xlsx"))
```

You can then `purrr::map()` over them to read all sheets. We will get to `purrr` later in the course; just know that batch-import is possible.

4.19 A glimpse ahead: From Excel to Quarto

In Chapter 5 you will write your first **Quarto** document. Quarto mixes:

- prose (written in Markdown),
- R code (in chunks like ```` ```{r} ... ``` ````), and
- the output of that R code (numbers, tables, charts)

…into one rendered HTML, PDF, or Word file. The chapter you are reading right now is a Quarto document.

In Excel (Ch 1–3)	In Quarto (Ch 5+)
Submit a `.xlsx` workbook	Submit a rendered `.html` (or `.pdf`) file
Charts as PNG inserts	Charts as code chunks that re-render

In Excel (Ch 1–3)	In Quarto (Ch 5+)
Cleaning log on a separate sheet	Cleaning log as Markdown next to the code
Hand-edited values inside cells	Code-computed values that update on re-render
File is the deliverable	The `.qmd` source AND the rendered file together

The principle is the same as Excel's *About / ReadMe / Data* pattern: documentation lives with the data. The difference is that the documentation, the code, and the output are now one file.

4.20 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

4.21 AI as a debugging companion

Useful prompts for this chapter

- *"I am brand new to R. I just installed it and I'm seeing this error message when I run `library(tidyverse)`. What does it mean?"* (paste the full error)
- *"Explain step by step what this R code does:* `bc |> filter(Fiscal_Year >= 2010) |> summarise(total = sum(Reforestation_ha))`*"*
- *"I'm getting* `Error in$.tbl_df(bc, Refrestation_ha)`*. The column name in my Excel file is* `Reforestation_ha`*. What's wrong and how do I fix it?"*

4.21.1 Verifying AI's answers

After AI suggests a fix, **check**:

1. Does the fix run without error in your console?
2. Does the resulting number match what you expect from Chapter 1?
3. Did AI invent a function that doesn't exist? (Run `?function_name` — if the help page doesn't open, the function is fictional.)

AI's R suggestions are usually correct but occasionally include non-existent functions (called "hallucinations"). The two-second check is to look up the function in R's own help.

4.21.2 When *not* to use AI

- Do not paste an exercise question into AI and copy the answer back. The point of the exercise is the thinking.
- Do not let AI make code-style decisions for you (pipe vs nested, snake_case vs camelCase) without understanding why.

- Do not let AI invent a *replacement* for `here()` that uses absolute paths. Portable paths are non-negotiable for this course.

4.22 Reading

- The *R for Data Science* book by Wickham, Çetinkaya-Rundel, & Grolemund — free at https://r4ds.hadley.nz/. Chapter 2 ("Workflow: basics") and Chapter 8 ("Data import") are directly relevant.
- The `here` package vignette: https://here.r-lib.org/articles/here.html
- Jenny Bryan's classic *"Project-Oriented Workflow"*: https://www.tidyverse.org/blog/2017/12/workflow-vs-script/
- RStudio Education resources: https://education.rstudio.com/learn/beginner/

Instructor notes

Pacing and emphasis

- Chapter 4 is structurally different from Chapters 1–3. The first half is **installation and setup**, which can swallow an entire lab session if your group is mostly new to R.
- The **Excel-to-R bridge table** in the *Side by side* section is the single most important content in this chapter. If learners only remember one thing, it should be that the numbers they computed in Excel are the same numbers R computes, using different syntax for the same data.
- **The old pipe `%>%`** is briefly mentioned in *Optional, advanced*. Keep the main path on the native `|>` only — the difference rarely matters for beginners, and consistency makes the course easier to teach.

4.22.1 Posit Cloud vs local install

- **Strongly recommend Posit Cloud** for the first lab. It eliminates 90% of "my install isn't working" tickets.
- Learners can switch to a local install later in the term once they are comfortable with the workflow.
- For the lab session, have one TA ready to help with local installs and another with Posit Cloud account creation.

4.22.2 Expected output checklist (matches the top-of-chapter callout)

1. RStudio Project named `frst232/` with the correct folder structure (verify with `list.files(here("data"))`).
2. Rendered Quarto HTML showing `dim(bc)`, `names(bc)`, `glimpse(bc)`, and the five basic statistics.
3. Group-lab worksheet or screenshot.

4.22.3 Materials provided alongside this chapter

File	Purpose
`bc_disturbance_reforestation.xlsx`	Raw BC silviculture workbook (carried forward from Chapter 1).
`frst232_ch04_learner.qmd`	Quarto document with TODO chunks for learners to complete.
`frst232_ch04_solutions.qmd`	Completed Quarto document with all code chunks filled in. **Do not distribute to learners.**
`frst232_ch04_starter.zip`	A pre-built starter project containing the .Rproj file, folder structure, learner .qmd, and a placeholder data folder.
`quiz_ch04.html`	Standalone interactive quiz. Embedded in the rendered chapter; can also be hosted separately.

4.22.4 Suggested facilitation guidance for the Practice Demo Lab

*These are optional facilitation and feedback suggestions for instructors running this book demo — **not** a grading key. The percentages below are relative emphasis, not Canvas marks; the graded lab assignment is on Canvas.*

Task	Weight	What to look for
Posit Cloud accounts (Task 1)	10 %	Everyone has an account.
Project + folders (Task 2)	20 %	All four folders present; project named correctly.
Packages installed (Task 3)	15 %	No load errors for tidyverse, readxl, here.
Data file uploaded (Task 4)	15 %	`list.files(here("data"))` returns the .xlsx name.
Successful import (Task 5)	15 %	`dim(bc)` returns `37 8`.
Sum matches Excel (Task 6)	15 %	`sum(bc$Reforestation_ha)` matches the Chapter 1 deliverable to the cent.
Screenshot submission (Task 7)	10 %	Screenshot legible; all group members named.

4.22.5 Common stumbling points

- **install vs library** confusion. Walk through the distinction explicitly at the start of the lab.
- **Forgot to load `readxl`.** `read_excel()` errors with *"could not find function"* if `readxl` is not loaded.
- **Wrong working directory.** Learners who skip the RStudio Project step end up with absolute paths that work on their machine but break in submission. Insist on `here()` from day one.
- **Misspelled column names.** R is case-sensitive. `bc$Refor` is not `bc$Reforestation_ha`. Tab completion fixes most of these.
- **The "saved workspace" trap.** RStudio's default is to save your workspace on exit and restore it on startup. This means yesterday's variables are still around — until they aren't, and your code mysteriously breaks. Recommend the **No / Never** setting from *Optional, advanced*.
- *Excel-to-R bridge:* Refer back to the side-by-side table often. When a learner asks *"is this the same as the Excel formula?"*, the answer is always *"yes — here's the row in the bridge table."*

5 Import and Inspect Forestry Data with R and Quarto

Data for this chapter

This chapter uses the **BC silviculture** (`bc_disturbance_reforestation.xlsx`), **Ontario forest statistics** (`on_forest_statistics_1.xlsx`), and **BC air-quality** (`airdata3.csv`) files. Click each file name to download it directly, then save it in your `data/` folder. For sources and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Explain what a **vector** is in R and recognize the four atomic types: logical, integer, double, character.
- Use `c()` to combine values into a vector, and apply vectorized functions (`sum`, `mean`, `length`) directly to a vector.
- Distinguish a **tibble** from a **data.frame** and explain why the tidyverse uses tibbles.
- Identify column types from `glimpse()` output (`<int>`, `<dbl>`, `<chr>`, `<lgl>`, `<date>`).
- Import an Excel file with `readxl::read_excel()` and a CSV file with `readr::read_csv()`.
- Inspect a tibble with `dim()`, `names()`, `glimpse()`, `summary()`, `head()`, and `tail()`.
- Count missing values per column with `colSums(is.na(...))` and distinguish `NA` from `0`.
- Write a small **data dictionary** in Quarto next to the code that produced it.
- Render a complete **Quarto HTML report** that another reader could open without R.

Expected output for this chapter

What you will hand in

1. A **rendered Quarto HTML report** named `frst232_ch05_learner.html`, containing:
   - The result of importing both `bc_disturbance_reforestation.xlsx` and `on_forest_statistics_1.xlsx`.
   - The inspection output — `dim()`, `names()`, `glimpse()`, `summary()`, `head()`, and `tail()` — for each file (the inspection moves taught in this chapter).
   - A **missing-value summary** for each file (one cell per column showing how many

`NA` values it has).
- A **data dictionary table** for the BC file, written by you in Markdown.
- A short prose paragraph (two or three sentences) describing what is in each file.

2. The corresponding **`.qmd` source file**.
3. Notification that you have completed the **Excel midterm** in the lab session for this chapter (see *Lab session — Excel midterm* below).
4. A short **group peer-review worksheet** of another learner's rendered HTML.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

5.1 Where we are in the course

In Chapter 4 you:

- Installed R, RStudio, and Quarto (or set up Posit Cloud).
- Created the `frst232/` project with `data/`, `outputs/`, `R/`, `notes/`.
- Imported `bc_disturbance_reforestation.xlsx` with `readxl::read_excel()`.
- Computed five statistics (`sum`, `mean`, `min`, `max`, `median`) on `bc$Reforestation_ha`.
- **Confirmed those numbers match the values you saw in Excel for Chapter 1.**

Chapter 5 deepens the R fundamentals: vectors, tibbles, column types, missing values, and your first full **Quarto report**.

5.2 Excel-to-R bridge — the inspection moves

From Chapter 4 you already know:

Excel inspection	R command	Returns
Status-bar `Count`	`dim(bc)` / `nrow(bc)`	Row and column count
Status-bar `Sum`	`sum(bc$ColumnName)`	Sum of one column
Status-bar `Average`	`mean(bc$ColumnName)`	Mean of one column

This chapter adds three more — the *missing-value* and *type* moves that Excel cannot do cleanly:

Excel pattern	R command	Returns
"Cells: blank or text? Hard to tell."	`glimpse(bc)`	Each column with type **and** first values
"Count = 35 but 37 rows? Two cells are text."	`colSums(is.na(bc))`	Number of missing values per column
"Format Cells → Number / Text"	`class(bc$ColumnName)`	The type ("numeric", "character", etc.)

By the end of this chapter, every move you used to do by *looking* at Excel is now a scripted R call.

5.3 Vectors — the building block

Almost everything in R is a **vector**: an ordered sequence of values, all of the same type.

```
# A vector of numbers
years <- c(1987, 1988, 1989, 1990, 1991)
years
```

```
[1] 1987 1988 1989 1990 1991
```

```
# A vector of text
species <- c("Pl", "Fd", "Sx")
species
```

```
[1] "Pl" "Fd" "Sx"
```

```
# A vector of TRUE/FALSE values
is_old <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
is_old
```

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

The `c()` function (**c** for *combine*) is how you build a vector. Type the values, comma-separated, inside `c(...)`.

5.3.1 The four atomic types

Every value in R has a type. The four most common:

Type	Examples	R shorthand
logical	`TRUE`, `FALSE`, `NA`	`<lgl>`
integer	`1L`, `2L`, `2026L`	`<int>`
double	`1`, `3.14`, `-2.5`	`<dbl>`
character	`"Vancouver"`, `"Pl"`	`<chr>`

Numbers without an `L` suffix are `<dbl>` (double-precision) by default. Most numeric R work uses doubles.

5.3.2 Vectorized operations

Most R functions work on a *whole vector at once*, not one element at a time:

```
  ha <- c(220000, 180000, 250000, 190000)

  # Each of these acts on the whole vector
  sum(ha)
```

```
[1] 840000
```

```
  mean(ha)
```

```
[1] 210000
```

```
  min(ha)
```

```
[1] 180000
```

```
  max(ha)
```

```
[1] 250000
```

```
  length(ha)
```

```
[1] 4
```

```
  # Arithmetic is vectorized too - element-by-element
  ha / 1000      # convert each value to thousands of hectares
```

```
[1] 220 180 250 190
```

```
  ha + 1000      # add 1000 to every value
```

```
[1] 221000 181000 251000 191000
```

```
  ha > 200000    # comparison returns a logical vector
```

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

That last result — `ha > 200000` returns a vector of `TRUE`/`FALSE` — is the bedrock of every filter and condition you will write in R.

5.3.3 Subsetting with `[`

You can pull out specific positions from a vector with square brackets:

```
  ha[1]                # first value
```

```
[1] 220000
```

```
  ha[c(1, 3)]          # first and third values
```

```
[1] 220000 250000
```

```
  ha[ha > 200000]      # only values greater than 200,000
```

```
[1] 220000 250000
```

That third example reads: *"return the elements of `ha` where `ha > 200000` is TRUE"*. This is the **logical-vector subsetting** pattern — the conceptual ancestor of `dplyr::filter()` which you will meet in Chapter 6.

5.4 Tibbles — vectors arranged in columns

A **tibble** is a tidyverse-flavored data.frame: a rectangular table where every column is a vector and all columns have the same length.

When you ran:

```
  bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                   sheet = "Data")
```

…in Chapter 4, R built a tibble called `bc`. Each column of `bc` is a vector. The whole tibble is a list of vectors that R prints as a rectangle.

```
# A tibble: 37 x 8
  Fiscal_Year Clearcutting_ha Clearcutting_with_reserves_ha Partial_cutting_ha
        <dbl>           <dbl>                         <dbl>              <dbl>
1        1987         216305.                        5800.             20077.
2        1988         217836.                        1843.             31878.
3        1989         190636.                        1457.             23786.
4        1990         172361.                        1400.             18931.
5        1991         192005.                        1515.             19697.
6        1992         209444.                        2951.             21691.
# i 31 more rows
# i 4 more variables: Harvested_ha <dbl>, Natural_Disturbance_ha <dbl>,
#   Total_Disturbance_ha <dbl>, Reforestation_ha <dbl>
```

Look at the top of the printout. Just below the column names, you see things like `<dbl>`, `<dbl>`, … — the column types. **A tibble always shows its column types when printed.** That alone is reason enough to use tibbles over base R data.frames.

5.4.1 Tibble vs data.frame

For most purposes, a tibble *is* a data.frame. The differences that matter for this course:

Feature	data.frame	tibble
Prints	All rows by default (slow on big data)	First 10 rows only
Column types	Hidden until you ask	Always shown
Subset with `[, "col"]`	Returns a vector	Returns a tibble
Created by	`read.csv()`, `data.frame()`	`read_excel()`, `read_csv()`, `tibble()`

You can always convert **in either direction**: `as_tibble(df)` turns a data.frame *into* a tibble, and `as.data.frame(tib)` turns a tibble *back into* a data.frame. For tidyverse work, stay with tibbles.

5.5 Importing files

> **AI prompt to check your work**
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"I imported `[file]` and got a tibble of `[rows] x [cols]`. Check whether the column types look right, whether any column that should be numeric came in as text (or a date as character), and which inspection commands I should run next. Do not hand me a finished import script."*

Two import functions cover almost every real-world file:

5.5.1 `readxl::read_excel()`

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")
```

Arguments to remember:

- The first argument is the **file path** — always build it with `here()`.
- `sheet =` names the sheet to read. If you skip it, `read_excel` reads the first sheet.
- `skip = 3` skips the first 3 rows (useful when the file has a banner or title above the headers).
- `na = c("", "NA", "n/a")` tells `read_excel` which strings to treat as missing.

5.5.2 `readr::read_csv()`

```
# Example (you do not need to run this; we use it in Chapter 6).
airdata <- read_csv(here("data", "airdata3.csv"))
```

Arguments to remember:

- The first argument is the file path.
- `col_types =` can lock in column types if `read_csv` guesses wrong (advanced — leave it off for now).
- `na = c("", "NA", "-9999")` is useful for fields that use sentinel values for missing.

5.5.3 Other readr siblings (briefly)

- `read_tsv()` — tab-separated.
- `read_delim()` — generic, any delimiter.

All readr functions return a tibble. All accept a file path as the first argument.

5.6 Inspecting a tibble — the eight moves

Whenever you load a new file, run these eight commands. They take seconds and catch most problems.

```
dim(bc)          # rows and columns
```

```
[1] 37  8
```

```
nrow(bc)         # rows only
```

```
[1] 37
```

```
ncol(bc)         # columns only
```

```
[1] 8
```

```
names(bc)        # column names
```

```
[1] "Fiscal_Year"                   "Clearcutting_ha"
[3] "Clearcutting_with_reserves_ha" "Partial_cutting_ha"
[5] "Harvested_ha"                  "Natural_Disturbance_ha"
[7] "Total_Disturbance_ha"          "Reforestation_ha"
```

```
  head(bc, 3)     # first three rows
```

```
# A tibble: 3 x 8
  Fiscal_Year Clearcutting_ha Clearcutting_with_reserves_ha Partial_cutting_ha
        <dbl>           <dbl>                         <dbl>              <dbl>
1        1987         216305.                        5800.             20077.
2        1988         217836.                        1843.             31878.
3        1989         190636.                        1457.             23786.
# i 4 more variables: Harvested_ha <dbl>, Natural_Disturbance_ha <dbl>,
#   Total_Disturbance_ha <dbl>, Reforestation_ha <dbl>
```

```
  tail(bc, 3)     # last three rows
```

```
# A tibble: 3 x 8
  Fiscal_Year Clearcutting_ha Clearcutting_with_reserves_ha Partial_cutting_ha
        <dbl>           <dbl>                         <dbl>              <dbl>
1        2021          13177.                       122562.              8295.
2        2022           9237.                       100909.              8953.
3        2023           7300.                        65369.              6292.
# i 4 more variables: Harvested_ha <dbl>, Natural_Disturbance_ha <dbl>,
#   Total_Disturbance_ha <dbl>, Reforestation_ha <dbl>
```

```
  glimpse(bc)     # every column, type, and first values
```

```
Rows: 37
Columns: 8
$ Fiscal_Year                  <dbl> 1987, 1988, 1989, 1990, 1991, 1992, 1993~
$ Clearcutting_ha              <dbl> 216304.92, 217836.07, 190635.52, 172360.~
$ Clearcutting_with_reserves_ha <dbl> 5800.400, 1842.952, 1456.666, 1400.013, ~
$ Partial_cutting_ha           <dbl> 20076.681, 31878.294, 23785.954, 18930.8~
$ Harvested_ha                 <dbl> 242182.0, 251557.3, 215878.1, 192691.6, ~
$ Natural_Disturbance_ha       <dbl> 20875.00, 9641.20, 10711.80, 28300.70, 1~
$ Total_Disturbance_ha         <dbl> 263057.0, 261198.5, 226589.9, 220992.3, ~
$ Reforestation_ha             <dbl> 201972.5, 221296.3, 237275.9, 260370.8, ~
```

```
  summary(bc)     # min / quartiles / mean / max for each numeric column
```

```
  Fiscal_Year   Clearcutting_ha  Clearcutting_with_reserves_ha
 Min.   :1987   Min.   :  7300   Min.   :   923.9
 1st Qu.:1996   1st Qu.: 28099   1st Qu.: 14044.9
 Median :2005   Median : 76531   Median :121658.2
 Mean   :2005   Mean   : 89774   Mean   : 97152.6
```

```
 3rd Qu.:2014   3rd Qu.:166369   3rd Qu.:155763.9
 Max.   :2023   Max.   :217836   Max.   :200699.1
 Partial_cutting_ha  Harvested_ha    Natural_Disturbance_ha
 Min.   : 2502      Min.   : 78961   Min.   :  1448
 1st Qu.: 5603      1st Qu.:190393   1st Qu.:  7562
 Median : 9273      Median :204791   Median : 13908
 Mean   :12127      Mean   :199053   Mean   : 29035
 3rd Qu.:18931      3rd Qu.:216302   3rd Qu.: 28733
 Max.   :31878      Max.   :251557   Max.   :218275
 Total_Disturbance_ha Reforestation_ha
 Min.   :116942       Min.   :189394
 1st Qu.:207574       1st Qu.:207668
 Median :224147       Median :219925
 Mean   :228089       Mean   :223353
 3rd Qu.:254544       3rd Qu.:237276
 Max.   :422092       Max.   :274616
```

Each one answers a different question:

What you want to know	Use
How many rows × columns?	`dim()`
Just rows	`nrow()`
Just columns	`ncol()`
The column names	`names()`
What the first / last rows look like	`head()` / `tail()`
Column types + a preview of values	`glimpse()`
Distributions of numeric columns	`summary()`

`glimpse()` is the single most valuable function in this list. Make it your default first move on any tibble.

5.7 Column types — and why they matter

In the `glimpse(bc)` output above, you see types like `<dbl>` next to each column name. The type tells you *what R thinks this column contains.*

Type	Means	Example
`<int>`	integer (whole numbers, no decimals)	`1987L`, `2L`
`<dbl>`	double (any real number)	`216304.919`
`<chr>`	character (text)	`"Vancouver"`
`<lgl>`	logical	`TRUE` / `FALSE`
`<fct>`	factor (categorical with a fixed set of levels)	rare on imported files

Type	Means	Example
`<date>`	date	2024-09-15

If `read_excel` reads `Fiscal_Year` as `<dbl>` but you want it as `<int>`, you can coerce after import:

```
bc$Fiscal_Year <- as.integer(bc$Fiscal_Year)
class(bc$Fiscal_Year)
```

```
[1] "integer"
```

Most of the time the inferred types are fine. The only case where it matters is when something *looks* numeric but arrives as text (often because of an extra space or a stray character).

5.7.1 How to spot a type bug

If you compute `sum(bc$Reforestation_ha)` and get the wrong answer — or an error message like *"non-numeric argument to binary operator"* — the column is probably character. Check:

```
class(bc$Reforestation_ha)
```

```
[1] "numeric"
```

If it returns `"character"`, fix it:

```
bc$Reforestation_ha <- as.numeric(bc$Reforestation_ha)
```

5.8 Missing values — `NA` is its own thing

> **AI prompt to check your work**
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"`colSums(is.na(df))` shows `[which columns]` have missing values. Check whether my plan to handle them (`[drop / keep / na.rm = TRUE]`) is appropriate for this dataset, and explain what could go wrong if I treated `NA` as zero. Do not decide the whole cleaning strategy for me."*

In R, missing values are represented by `NA`. `NA` is **not** zero. It is **not** an empty string. It is its own thing: a flag meaning *"we do not know this value"*. If that distinction is new to you, skim **`NA` vs `0`** (below) first — treating an unknown value as `0` silently corrupts every sum and mean.

5.8.1 Counting NAs in one column

```
is.na(bc$Reforestation_ha)            # logical vector - TRUE for each NA
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE
```

```
sum(is.na(bc$Reforestation_ha))       # how many NAs in this column
```

```
[1] 0
```

`is.na()` returns a logical vector of the same length as its input — `TRUE` for each `NA`, `FALSE` for each real value. Sum that vector to count the missing values (`TRUE` is treated as `1`, `FALSE` as `0`).

5.8.2 Counting NAs in every column at once

```
colSums(is.na(bc))
```

```
                 Fiscal_Year               Clearcutting_ha
                           0                             0
Clearcutting_with_reserves_ha           Partial_cutting_ha
                           0                             0
               Harvested_ha       Natural_Disturbance_ha
                           0                             0
       Total_Disturbance_ha             Reforestation_ha
                           0                             0
```

This is the **fastest missing-value audit in R**. One line. Every column. Always run it after import.

column	n_missing
Fiscal_Year	0
Clearcutting_ha	0
Clearcutting_with_reserves_ha	0
Partial_cutting_ha	0
Harvested_ha	0
Natural_Disturbance_ha	0
Total_Disturbance_ha	0
Reforestation_ha	0

For the BC silviculture file, every column shows `0` — there are no missing values, which matches what you saw in Excel for Chapter 1. (The Ontario file is the same — we will check in the exercises.)

5.8.3 `NA` vs `0` — a habit worth forming

Situation	What to record
The variable was measured and the value really is zero	`0`
The variable was not measured	`NA`
The plot did not exist that year	`NA`
The measurement was below the detection limit	depends — see your data dictionary

Mixing these up is a silent bug class. *"Mean reforestation"* including 12 unmeasured years coded as `0` will be 12 zeros pulled into the average. Coded as `NA`, they would be skipped (if you use `na.rm = TRUE`).

5.9 Skipping `NA` in calculations

```
# If a column had NAs, the default behaviour is:
mean(c(1, 2, NA, 4))    # returns NA
```

```
[1] NA
```

```
# Add na.rm = TRUE to skip missing values:
mean(c(1, 2, NA, 4), na.rm = TRUE)   # returns 7/3 = 2.33
```

```
[1] 2.333333
```

`na.rm = TRUE` is available on every aggregation function (`sum`, `mean`, `min`, `max`, `median`, `sd`). Add it whenever your data might have missing values. Forgetting it is the most common silent bug in first-year R.

5.10 A worked inspection — the BC file end-to-end

Here is the canonical inspection sequence on a freshly imported file:

```
# 1. Import
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")
```

```r
  # 2. Dimensions and column names
  dim(bc)
```

```
[1] 37  8
```

```r
  names(bc)
```

```
[1] "Fiscal_Year"                    "Clearcutting_ha"
[3] "Clearcutting_with_reserves_ha" "Partial_cutting_ha"
[5] "Harvested_ha"                   "Natural_Disturbance_ha"
[7] "Total_Disturbance_ha"           "Reforestation_ha"
```

```r
  # 3. Type + preview
  glimpse(bc)
```

```
Rows: 37
Columns: 8
$ Fiscal_Year                  <dbl> 1987, 1988, 1989, 1990, 1991, 1992, 1993~
$ Clearcutting_ha              <dbl> 216304.92, 217836.07, 190635.52, 172360.~
$ Clearcutting_with_reserves_ha <dbl> 5800.400, 1842.952, 1456.666, 1400.013, ~
$ Partial_cutting_ha           <dbl> 20076.681, 31878.294, 23785.954, 18930.8~
$ Harvested_ha                 <dbl> 242182.0, 251557.3, 215878.1, 192691.6, ~
$ Natural_Disturbance_ha       <dbl> 20875.00, 9641.20, 10711.80, 28300.70, 1~
$ Total_Disturbance_ha         <dbl> 263057.0, 261198.5, 226589.9, 220992.3, ~
$ Reforestation_ha             <dbl> 201972.5, 221296.3, 237275.9, 260370.8, ~
```

```r
  # 4. Numeric summaries
  summary(bc)
```

```
  Fiscal_Year   Clearcutting_ha  Clearcutting_with_reserves_ha
 Min.   :1987   Min.   :  7300   Min.   :   923.9
 1st Qu.:1996   1st Qu.: 28099   1st Qu.: 14044.9
 Median :2005   Median : 76531   Median :121658.2
 Mean   :2005   Mean   : 89774   Mean   : 97152.6
 3rd Qu.:2014   3rd Qu.:166369   3rd Qu.:155763.9
 Max.   :2023   Max.   :217836   Max.   :200699.1
 Partial_cutting_ha  Harvested_ha    Natural_Disturbance_ha
 Min.   : 2502      Min.   : 78961   Min.   :  1448
 1st Qu.: 5603      1st Qu.:190393   1st Qu.:  7562
 Median : 9273      Median :204791   Median : 13908
 Mean   :12127      Mean   :199053   Mean   : 29035
 3rd Qu.:18931      3rd Qu.:216302   3rd Qu.: 28733
```

```
 Max.   :31878      Max.   :251557   Max.   :218275
 Total_Disturbance_ha Reforestation_ha
 Min.   :116942       Min.   :189394
 1st Qu.:207574       1st Qu.:207668
 Median :224147       Median :219925
 Mean   :228089       Mean   :223353
 3rd Qu.:254544       3rd Qu.:237276
 Max.   :422092       Max.   :274616
```

```
  # 5. Missing-value audit
  colSums(is.na(bc))
```

```
                 Fiscal_Year               Clearcutting_ha
                           0                             0
Clearcutting_with_reserves_ha           Partial_cutting_ha
                           0                             0
               Harvested_ha       Natural_Disturbance_ha
                           0                             0
       Total_Disturbance_ha             Reforestation_ha
                           0                             0
```

```
  # 6. Spot-check the verification identity from Chapter 1
  identical_check <- all.equal(
    bc$Harvested_ha + bc$Natural_Disturbance_ha,
    bc$Total_Disturbance_ha
  )
  identical_check
```

```
[1] TRUE
```

Six lines. Six confirmations. You now know:

- The file has 37 rows and 8 columns (matches Chapter 1).
- Column names match the ReadMe (you can compare them by eye).
- Every column is `<dbl>` — no surprise text-as-number issues.
- The `summary()` table tells you each column's range.
- No missing values anywhere.
- The arithmetic identity from Chapter 1 still holds: `Harvested + Natural = Total` for every row.

That sequence is **what professional R analysts do every time they open a new file**. Do it on every dataset you touch.

5.11 First Quarto report

You have been *reading* Quarto documents through this whole book. Now you write one.

5.11.1 The four parts of a `.qmd` file

A Quarto file has four ingredients:

Part	Looks like	What it does
YAML header	`---` block at the top	Tells Quarto how to render
Markdown text	Plain prose, headings, bullets	The reading content
Code chunks	```` ```{r} ... ``` ````	R code that gets executed
Inline code	`` `r expression` ``	One-line R inside prose

A minimal `.qmd` file:

````
---
title: "My first Quarto report"
format: html
---

# Introduction

This report inspects the BC silviculture file.


::: {.cell}

```{.r .cell-code}
library(readxl)
library(here)
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")
dim(bc)
```

::: {.cell-output .cell-output-stdout}

```
[1] 37  8
```


:::
:::


The file has 37 rows.
````

Save as `my_first_report.qmd`. Click **Render** (or **Ctrl+Shift+K** / **Cmd+Shift+K**). You get a single self-contained HTML file.

5.11.2 Common chunk options

Chunk options go right after the ```{r} line, prefixed with #|:

```
::: {.cell}

:::
```

Option	What it does
`label:`	Names the chunk (helpful when one errors)
`echo: false`	Hide the code in the rendered output (just show the result)
`include: false`	Run the code but hide both code and output (useful for setup)
`message: false`	Hide messages from R (e.g., the tidyverse load message)
`warning: false`	Hide warnings
`eval: false`	Show the code but do not run it
`fig.width: 6`	Plot width in inches (Chapter 9)
`fig.cap: "..."`	Caption for figures

5.11.3 What rendering does

When you click **Render**, Quarto:

1. Reads your `.qmd` file.
2. Runs every R chunk in order, from top to bottom, in a fresh R session.
3. Captures the output (text, tables, charts).
4. Stitches code + text + output into a single HTML file.
5. Saves the HTML alongside the `.qmd`.

The **fresh session** part is important: rendering does not use your interactive console. If a chunk depends on a variable that was only created in your console (and not in an earlier chunk), rendering will fail.

A reproducibility test

If your `.qmd` renders cleanly from a fresh R session, your work is reproducible. If it does not, your work depends on state that no one else can see — and that is the bug.
Restart R often. In RStudio: **Session → Restart R** (Ctrl+Shift+F10 / Cmd+Shift+F10). Then render. If it still works, you are good.

5.12 Writing a data dictionary in Quarto

A data dictionary lives next to your code. In Quarto, it is just a Markdown table:

```
| Column | Type | Unit | Description |
|---|---|---|---|
| `Fiscal_Year` | integer | year | Fiscal year (April-March) |
| `Harvested_ha` | double | hectares | Total area harvested |
| ... | | | |
```

That renders as:

Column	Type	Unit	Description
`Fiscal_Year`	integer	year	Fiscal year (April–March)
`Harvested_ha`	double	hectares	Total area harvested
…			

The point: **the dictionary travels with the analysis**. When you hand your `.qmd` to a colleague, they see the table and the code that built the numbers, in the same file.

Starter for the BC file

The BC silviculture file has these eight columns. Copy this table into your report and fill in the **Type**, **Unit**, and **Description** yourself (use `glimpse(bc)` for the types):

Column	Type	Unit	Description
`Fiscal_Year`			
`Clearcutting_ha`			
`Clearcutting_with_reserves_ha`			
`Partial_cutting_ha`			
`Harvested_ha`			
`Natural_Disturbance_ha`			
`Total_Disturbance_ha`			
`Reforestation_ha`			

File-summary template. After the dictionary, add two or three sentences that turn your inspection output into prose, e.g.:

> *"The BC silviculture file has 37 rows and 8 columns. The variables are numeric areas in hectares plus a fiscal-year integer, and the missing-value audit shows no missing values. The main variables describe harvested, natural-disturbance, total-disturbance, and reforested area by fiscal year."*

5.13 Active learning

5.13.1 Activity 1 — Vectors

```
ha <- c(220000, 180000, 250000, 190000, 210000)
```

In the console, compute:

- the sum,
- the mean,
- the count of values greater than 200,000 (hint: `sum(ha > 200000)`),
- the values greater than 200,000 themselves.

5.13.2 Activity 2 — Inspect two files

Run the six-move inspection on the BC silviculture file, then on the Ontario forest stats file. Note the differences:

```
ontario <- read_excel(here("data", "on_forest_statistics_1.xlsx"),
                      sheet = "Data")
dim(ontario); glimpse(ontario)
```

How are the column types different from the BC file? Which file has any character columns?

5.13.3 Activity 3 — Missing-value audit

Run `colSums(is.na(bc))` and `colSums(is.na(ontario))`. Are there any missing values? Confirm against what you saw in Excel for Chapters 1 and 2.

5.13.4 Activity 4 — Render your first Quarto report

Open the chapter learner file `frst232_ch05_learner.qmd`. Render it. Confirm the HTML opens and shows all expected output.

5.13.5 Activity 5 — Break and fix

In `frst232_ch05_learner.qmd`, deliberately misspell `Reforestation_ha` somewhere. Render. Read the error message carefully. Fix it. Re-render. The error message taught you something useful — what?

5.14 Lab session — Excel midterm

! The lab session for Chapter 5 is the Excel midterm

There is no separate practice lab this chapter. The lab session is given over to the **Excel midterm**, covering material from Chapters 1–3:

- Importing and inspecting an Excel workbook (Chapter 1).
- Cleaning, summarizing, and writing formulas (Chapter 2).
- Visualization and PivotTables (Chapter 3).

The midterm is closed-AI but open-notes. Bring a laptop with Excel installed (or use the lab computers). The teaching team will provide a midterm-specific workbook at the start of the session.

What to do before the lab:

- Re-read the Chapter 1, 2, and 3 *Expected output* callouts.
- Make sure your laptop has Excel working.
- Bring your `bc_disturbance_reforestation.xlsx` and `on_forest_statistics_1.xlsx` files in case the midterm references them.

What is NOT on the midterm:

- R code.
- Quarto rendering.
- Anything from Chapter 4 or Chapter 5.

The R material is assessed through homework exercises, the chapter quizzes, and the final exam at the end of the term.

5.15 Group activity — Peer review of Quarto reports

Outside the midterm session, run this peer-review activity at any time during the chapter (e.g., during a lecture session).

i Group: Peer review a rendered HTML report

Work in pairs (or threes). Each member should have a rendered `frst232_ch05_learner.html` to share.

Task 1 — Swap. Each person sends their rendered HTML to the person on their left. Open it in a browser.

Task 2 — Score each other's report using this rubric:

Criterion	0	1	2
Renders without error	Did not render	Renders with warnings	Renders cleanly
`dim()`, `names()`, `glimpse()` visible	Missing	Present but unclear	Clearly visible
Missing-value summary present	Missing	Present but unexplained	Present with note
Data dictionary table	Missing	Incomplete	Complete and readable
Two-sentence prose summary	Missing	Vague	Specific to the file

Task 3 — Give one piece of constructive feedback in writing. What is the strongest part of the report? What would you change?
Task 4 — Submit. Submit your peer-review rubric (with your own name and the author's name) to the course site.
This activity is designed to fit inside a single lecture session.

5.16 Exercises

These are the take-home exercises that go with this chapter.

1. Run the **six-move inspection** sequence on the BC silviculture file. Copy the output of each function into your `.qmd` file inside R chunks.
2. Run the same six-move inspection on the **Ontario file**. Compare the two outputs. Which file has character columns? How many?
3. Compute the **missing-value summary** for both files using `colSums(is.na(...))`. Are there any missing values?
4. Build a **data dictionary table** for the BC silviculture file in your `.qmd`, in the format shown in this chapter. Each row should have *Column*, *Type*, *Unit*, *Description*.
5. Use **vectorized arithmetic** to compute the ratio `Reforestation_ha / Total_Disturbance_ha` for every year. Save the result as a new vector. How many years had a ratio greater than 1?
6. Use `summary(bc)` to find the **median** of `Total_Disturbance_ha`. Confirm it matches the value you would compute with `median(bc$Total_Disturbance_ha)`.
7. **Restart R** (Session → Restart R). Re-render your `frst232_ch05_learner.qmd` from scratch. Confirm it renders without errors. (If it does not, fix what depends on hidden console state.)
8. **Optional, harder.** Write a tiny R function called `inspect_file()` that takes a tibble and prints `dim`, `names`, `glimpse`, and `colSums(is.na(...))` all at once. Apply it to `bc` and `ontario`.

5.17 Optional, advanced

5.17.1 Lists — the catch-all data structure

A **list** in R is like a vector except its elements can be of different types. A tibble is technically a list of equal-length vectors.

```
my_list <- list(name = "BC silviculture",
                rows = 37,
                cols = 8,
                tibble = bc)
my_list$name
my_list$tibble
```

You will use lists in the **optional** spatial-data material, where each "row" is a complex geometry.

5.17.2 Factors — categorical variables with order

A **factor** is a vector with a fixed set of allowed values (levels). Useful when you want ordered categories (`small` < `medium` < `large`) or when a categorical variable has a known set of valid values.

```
size <- factor(c("small", "large", "medium", "large"),
               levels = c("small", "medium", "large"))
size
```

We will use factors more in Chapter 9 (ggplot2 axis ordering).

5.17.3 Locking column types at import time

If `read_excel` or `read_csv` guesses a column type wrong, you can override it:

```
read_csv("file.csv",
         col_types = cols(
           plot_id = col_character(),
           dbh_cm = col_double(),
           date_measured = col_date(format = "%Y-%m-%d")
         ))
```

This is overkill for clean BC and Ontario files. It becomes useful when you import messy real-world CSVs in Chapter 6.

5.18 A glimpse ahead: From inspection to cleaning

In Chapter 6 you move from *inspecting* to *cleaning*. The core moves of `dplyr`:

What you want to do	Excel (Ch 2)	dplyr verb (Ch 6)
Keep some rows	AutoFilter	`filter()`
Add a new column	`=A2+B2` in a new column	`mutate()`
Sort	Data → Sort	`arrange()`
Keep only some columns	Delete columns	`select()`
Group and summarise	PivotTable	`group_by() |> summarise()`
Look up from a reference	`VLOOKUP`	`left_join()`
Recode values	`IFS`	`case_when()`

Every Excel cleaning move you learned in Chapter 2 has a one-line dplyr equivalent. The names are short, the pattern is consistent, and once you know the six verbs above, you can clean almost any real-world dataset.

Chapter 6 walks through each verb on the BC air-quality dataset (`airdata3.csv`) — a messier file than what you have seen so far, with actual missing values, mixed date formats, and the kind of problems that real forestry data presents.

5.19 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

5.20 AI as a debugging companion

Useful prompts for this chapter

- *"My Quarto document fails to render with the error `object 'bc' not found`. The variable works fine in my console. What might be wrong?"* (Hint: fresh-session rendering.)
- *"Explain this output of `glimpse(bc)` line by line:"* (paste the glimpse output)
- *"I imported a CSV with `read_csv` and one of my numeric columns was read as character. What's the most common cause and how do I fix it?"*

5.20.1 A prompt template for the rest of the course

When you ask AI for help with R code, include four things:

1. Your R version (`R.version.string`).
2. The packages you have loaded (the top of your `.qmd`).
3. The exact code you ran.

4. The exact error message (copy-paste, do not paraphrase).

Without these four, the answer will be too generic to use.

5.20.2 Verifying AI's answers

After AI suggests a fix, **check**:

1. Run the suggested code in your console. Does it work?
2. Does the result match what you expect from `glimpse()`, `summary()`, or a manual count?
3. If AI rewrites a chunk for you, render the whole document afterwards to confirm nothing else broke.

5.20.3 When *not* to use AI

- Do not let AI write your data dictionary. The dictionary describes *your* analysis decisions.
- Do not let AI hide an unexplained behaviour with a `try()` or `suppressWarnings()`. Warnings exist for a reason — read them.

5.21 Reading

- *R for Data Science* — chapter 2 *Workflow: basics* and chapter 8 *Data import*. Free at https://r4ds.hadley.nz/.
- *Quarto official guide* — https://quarto.org/docs/guide/. Start with *Authoring* → *Markdown Basics* and *Computations* → *R*.
- Jenny Bryan, *"Naming things"* — short lecture on file naming conventions: https://speakerdeck.com/jennybc/how-to-name-files.
- The `readxl` documentation: https://readxl.tidyverse.org/.

Instructor notes

Pacing and emphasis

- The chapter has two big new ideas: **vectors / tibbles** (the R type system) and **Quarto** (the new deliverable format). Spend time on both; do not rush past the YAML header.
- The **missing-value audit** (`colSums(is.na(bc))`) is the single most useful one-line R command in this chapter. Use it in your live demo.
- **Quarto rendering must succeed before submission.** Many learners will hand in a `.qmd` that does not render. Build the "render from a fresh session" check into the grading rubric.

5.21.1 Excel midterm logistics

- The lab session for this chapter is the midterm.

- Material is Chapter 1, 2, 3 Excel only.
- Recommended midterm structure: one 90-minute Excel deliverable (single workbook) with sections on importing, cleaning, summarizing, and a PivotTable. Provide a starter file at the start of the session.
- Closed-AI, open-notes. Learners may use the chapter materials but not generative-AI tools.
- TAs should be prepared for *"my Excel won't open"* support questions at the start of the session — have a backup computer ready.

5.21.2 Expected output checklist (matches the top-of-chapter callout)

1. `frst232_ch05_learner.html` (rendered) with imports for both BC and Ontario files.
2. `frst232_ch05_learner.qmd` (source).
3. Excel midterm completed in the lab session.
4. Peer-review worksheet (during the lecture peer-review activity).

5.21.3 Materials provided alongside this chapter

File	Purpose
`bc_disturbance_reforestation.xlsx`	Raw BC silviculture (from Ch 1, unchanged).
`on_forest_statistics_1.xlsx`	Raw Ontario forest stats (from Ch 2, unchanged).
`frst232_ch05_learner.qmd`	Quarto starter with TODO chunks for each exercise.
`frst232_ch05_solutions.qmd`	Completed Quarto document. **Do not distribute to learners.**
`quiz_ch05.html`	Standalone interactive quiz.

5.21.4 Common stumbling points

- **"Object not found on render."** Almost always: the variable was created in the console, not in a chunk. Solution: Restart R, then re-render.
- **YAML errors.** Quarto is fussy about the YAML header. Indentation must use spaces (no tabs); the `---` fences must be on their own lines.
- **`library(tidyverse)` not run in the setup chunk.** Then the first chunk that uses a tidyverse function errors. Make sure the setup chunk has `include: false` (runs but is hidden) and loads every package the document needs.
- **NA confusion.** Some learners conflate NA with 0 when they see a "missing value count of 0" — they think *"so there are zero values?"*. Walk through the distinction explicitly.
- *Excel-to-R bridge:* The most important sentence in this chapter is *"every move you used to do by looking at Excel is now a scripted R call"*. Refer back to it whenever a learner asks why a particular function exists.

Working with AI on this chapter

Use AI as a **verification and debugging partner**, not an answer machine. The goal is not to make AI do the work — it is to use AI to help you **understand, check, and improve your own work.**

AI prompt to check your work

Fill in the brackets and paste this into your AI assistant. It should *explain and check*, not hand you a final assignment answer:

> *"I am working on FRST 232, this chapter. My dataset is `[file name]` with columns `[columns]`. My task is `[what you are trying to do]`. I tried `[paste your code or steps]` and got `[paste the exact output or error]`. Explain what this does line by line, tell me whether it answers the question, and list the checks I should run to verify it. Do not give me the final answer."*

Verify it yourself — before trusting any result (yours *or* an AI's):

- the rows, columns, and units are what you expect;
- the row count changes sensibly after a `filter()` or a join;
- group sizes and summary values are plausible;
- the figure answers the question and axis labels include units.

Common AI mistakes to watch for: inventing a column, function, or value that is not in your data; choosing a tool before reading the question; and (in Excel) suggesting the wrong cell range or forgetting to refresh a PivotTable.

Compare your solution with the reference

When a reference solution is available, do **not** just copy it — use it to check your own work:

- **Expected result** — does your output match the target shape and values?
- **Required components** — did you include every step the task asked for?
- **If your result differs** — say *where* it matched, *where* it differed, *what* caused the difference, *how* you fixed it, and *what* you learned.

AI replication challenge

Write a prompt that would guide an AI *toward* the reference solution without handing over the final answer. Include the dataset, the columns, the task goal, the output format you need, and the verification checks. Then paste (or summarise) the AI's response and answer: did its approach match the book's? did it miss a verification step? did it invent a column, function, or value? what did your group change after checking it?

6 Clean and Prepare Forestry Data in R

Data for this chapter

This chapter uses the **BC air-quality** file (`airdata3.csv`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Recognize the **six core dplyr verbs**: `filter()`, `select()`, `mutate()`, `arrange()`, `group_by()`, `summarise()`.
- Drop unwanted columns with **`select()`** and drop unwanted rows with **`filter()`**.
- Add new columns or transform existing ones with **`mutate()`**.
- Sort tibbles with **`arrange()`** (ascending) and `desc()` (descending).
- Parse messy text dates into proper R dates with **`lubridate::mdy()`** and friends.
- Recode values into categories with **`case_when()`** (the R equivalent of `IFS` from Chapter 2).
- Look up reference values with **`left_join()`** (the R equivalent of `VLOOKUP`).
- Chain operations into a single readable pipeline with the native pipe `|>`.
- Maintain a **cleaning log** inside a Quarto document — text and code side by side, every change documented.

Expected output for this chapter

What you will hand in

1. A **cleaned tibble** of `airdata3.csv`, saved as `outputs/airdata3_clean.csv`, containing only the useful columns, with the date parsed properly, and missing-value rows removed where appropriate.
2. A **rendered Quarto HTML report** (`frst232_ch06_learner.html`) showing the full cleaning sequence and a cleaning log.
3. A **station-level summary table** of mean O_3 by monitoring station, sorted in descending order.
4. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

6.1 Why dplyr

In Chapter 2 you cleaned data in Excel using AutoFilter, sorts, helper columns with `IF/IFS`, `VLOOKUP`, and `SUMIFS`. Every move was visible — but every move was also a click that left no record. *"How did you get this number?"* months later was hard to answer.

dplyr is the tidyverse package for data cleaning in R. Its verbs do exactly what your Excel moves did, with three changes:

1. Every move is a **written instruction** that can be re-run.
2. Every move is **named after the operation** (`filter` not "AutoFilter"; `mutate` not "add helper column").
3. Moves **chain together** with the pipe `|>`, so a multi-step cleaning is one block of code that reads top to bottom.

6.2 Excel-to-dplyr bridge

This is the cleaning bridge that Chapter 2 previewed and Chapter 5 introduced. Now you write the right-hand column for real.

What you did in Excel (Ch 2)	What you do in R (Ch 6)
AutoFilter to hide rows	`filter(data, condition)`
Sort A→Z	`arrange(data, column)`
Sort Z→A	`arrange(data, desc(column))`
Delete unwanted columns	`select(data, -unwanted_col)`
Helper column with `=A2+B2`	`mutate(data, new_col = A + B)`
`IFS(x<1000, "small", ...)`	`case_when(x < 1000 ~ "small", ...)`
`VLOOKUP(...)`	`left_join(data, reference, by = "key")`
PivotTable (sum by group)	`group_by(col) |> summarise(sum = sum(x))`
Cleaning log on a separate sheet	Markdown prose between code chunks

Every cell of the right-hand column is one line of code. After this chapter, you will be able to clean a dataset in R as fluently as you cleaned one in Excel — with the bonus that the work is now reproducible.

6.3 The dataset for this chapter

This chapter uses one openly licensed file from the **BC Government — Environmental Reporting BC** program:

Property	Value
File	`airdata3.csv`
Source	BC Government — Environmental Reporting BC, BC Air Data

Property	Value
Catalogue	https://envistaweb.env.gov.bc.ca/
Licence	Open Government Licence — British Columbia
Rows	11,904
Columns	27 (most of them empty)
Coverage	16 air-quality monitoring stations across Metro Vancouver
Time period	January 2000 (one month of hourly readings)

6.3.1 Why this dataset matters for forestry

Forest health and air quality are tightly linked. **Ground-level ozone (O_3)** is one of the most damaging air pollutants for vegetation: it enters leaves and needles through the stomata, injures foliage, and reduces photosynthesis and growth over a season. Unlike a pollutant that is emitted directly, ozone *forms* when sunlight reacts with pollutants from vehicles, industry, and — during fire season — wildfire emissions. The 16 stations in this file are part of the network BC uses to monitor air quality across the Lower Mainland.

A working analyst at a BC ministry, regional district, or consultancy will use exactly this format — hourly readings from many stations — to assess pollutant exposure. In this chapter we focus on O_3, which is recorded at eight of these stations, so that later chapters can compare readings across stations and regions.

> **!** The file is intentionally messy
>
> Unlike the BC silviculture and Ontario forest stats files (which were pre-cleaned by their publishers), `airdata3.csv` arrives in a state much closer to **real-world raw data**:
>
> - Several columns are **entirely missing** (every row is `NA`). We need to detect and drop these.
> - Two **redundant index columns** (`"Unnamed: 0"` and similar) were added by an upstream tool — we drop these too.
> - **Dates are stored as text** in `M/D/YYYY` format (`"1/1/2000"`). R does not recognise these as dates until we parse them.
> - **Many rows have no O_3 measurement** because not every station measures every pollutant.
>
> Real forestry data looks like this. The cleaning work in this chapter is the cleaning work you will do all the time as a working analyst.

6.4 Importing the file

In your `frst232/` project (from Chapter 4 or 5), the file should live at `data/airdata3.csv`. If you do not have it yet, download it from the BC Government link above and place it there.

```
airdata_raw <- read_csv(here("data", "airdata3.csv"),
                        show_col_types = FALSE)
```

```
New names:
* `` -> `...1`
```

```
dim(airdata_raw)
```

```
[1] 11904    27
```

Notice the variable name: **airdata_raw**. We never modify this in place. Every cleaning step creates a new variable. The raw tibble stays untouched in memory in case we need to start over.

6.5 Inspect first — the six-move check from Chapter 5

```
dim(airdata_raw)
```

```
[1] 11904    27
```

```
glimpse(airdata_raw)
```

```
Rows: 11,904
Columns: 27
$ ...1          <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
$ `Unnamed: 0`  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1~
$ WSPD_VECT     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ WDIR_SCLR     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ WSPD_SCLR     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ PM10          <dbl> 13, 6, 2, 2, 4, 2, 4, 6, 6, 8, 7, 4, 3, 3, 6, 6, 6, 5, 8~
$ SO2           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ NOx           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ WDIR_VECT     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ TEMP_MEAN     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ CO            <dbl> 0.5, 0.4, 0.4, 0.3, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.4, 0~
$ NO            <dbl> 10, 4, 2, 1, 1, 1, 1, 1, 2, 5, 4, 3, 3, 3, 3, 2, 3, 2, 5~
$ NO2           <dbl> 22, 21, 19, 8, 9, 9, 12, 19, 20, 23, 17, 10, 9, 8, 8, 10~
$ PM25          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ PRECIP_TOTAL  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ O3            <dbl> 1, 3, 18, 32, 31, 32, 28, 19, 17, 13, 15, 20, 19, 21, 21~
$ ATM_PRESS_1HR <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ HUMIDITY      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
$ location      <chr> "Burnaby South", "Burnaby South", "Burnaby South", "Burn~
$ Date          <chr> "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000", "1/1/200~
$ Time          <time> 01:00:00, 02:00:00, 03:00:00, 04:00:00, 05:00:00, 06:00~
$ RAD_TOTAL     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ NH3           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ PM25_5030i    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ TRS           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ RAD_NET       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ PM25_SHARP    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
colSums(is.na(airdata_raw)) |> sort(decreasing = TRUE) |> head(15)
```

```
  WSPD_VECT     WDIR_SCLR     WSPD_SCLR           NOx     WDIR_VECT
      11904         11904         11904         11904         11904
 TEMP_MEAN  PRECIP_TOTAL ATM_PRESS_1HR      HUMIDITY     RAD_TOTAL
      11904         11904         11904         11904         11904
       NH3    PM25_5030i       RAD_NET    PM25_SHARP           TRS
      11904         11904         11904         11904         11173
```

What we see:

- 11,904 rows × 27 columns.
- Many `<dbl>` columns. Some `<chr>` columns (`location`, `Date`, `Time`).
- **Several columns are 100% missing** — the count of NAs equals the total row count.

The 100%-missing columns are an artefact of how this dataset is distributed: the schema reserves room for variables that *some* stations might measure, but the stations in this extract do not all measure them. Our first cleaning move is to drop them.

6.6 Verb 1 — `select()` (keep or drop columns)

`select()` chooses which columns to keep. The simplest form is to list the columns you want:

```
airdata_raw |>
  select(location, Date, Time, O3, PM10, NO2, SO2, CO) |>
  head(3)
```

```
# A tibble: 3 x 8
  location      Date     Time      O3  PM10   NO2   SO2    CO
  <chr>         <chr>    <time> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Burnaby South 1/1/2000 01:00      1    13    22     0   0.5
2 Burnaby South 1/1/2000 02:00      3     6    21     0   0.4
3 Burnaby South 1/1/2000 03:00     18     2    19     0   0.4
```

The pipe `|>` reads: *"take* ***`airdata_raw`****, then pass it into* ***`select()`****, then take just the first 3 rows".*

6.6.1 Dropping columns with -

To drop named columns, prefix them with -:

```
  # Drop the redundant index columns added by an upstream tool.
  airdata_step1 <- airdata_raw |>
    select(-`...1`, -starts_with("Unnamed"))
  ncol(airdata_raw)        # columns before
```

```
[1] 27
```

```
  ncol(airdata_step1)      # columns after - a few fewer
```

```
[1] 25
```

```
  names(airdata_step1)     # and here is exactly what remains
```

```
 [1] "WSPD_VECT"     "WDIR_SCLR"     "WSPD_SCLR"     "PM10"
 [5] "SO2"           "NOx"           "WDIR_VECT"     "TEMP_MEAN"
 [9] "CO"            "NO"            "NO2"           "PM25"
[13] "PRECIP_TOTAL"  "O3"            "ATM_PRESS_1HR" "HUMIDITY"
[17] "location"      "Date"          "Time"          "RAD_TOTAL"
[21] "NH3"           "PM25_5030i"    "TRS"           "RAD_NET"
[25] "PM25_SHARP"
```

`starts_with("Unnamed")` is a **selection helper** — a pattern match that grabs every column whose name starts with `"Unnamed"`. Other helpers: `ends_with()`, `contains()`, `matches()` (regex), and `where()` (predicate).

6.6.2 Dropping all-NA columns programmatically

We saw above that several columns are entirely missing. We can drop them in one move with `where()`:

```
  airdata_step2 <- airdata_step1 |>
    select(where(\(col) !all(is.na(col))))
  ncol(airdata_step2)
```

```
[1] 11
```

```
  names(airdata_step2)
```

```
 [1] "PM10"     "SO2"      "CO"       "NO"       "NO2"      "PM25"
 [7] "O3"       "location" "Date"     "Time"     "TRS"
```

Read it: *"keep columns where it is **not** the case that all values are NA"*. We just removed twelve dead-weight columns with one line.

6.7 Verb 2 — `filter()` (keep or drop rows)

`filter()` keeps rows where a condition is `TRUE`:

```
# Keep only rows where O3 was actually measured.
airdata_step3 <- airdata_step2 |>
  filter(!is.na(O3))
nrow(airdata_step3)
```

```
[1] 5677
```

`!is.na(O3)` reads *"O3 is NOT missing"*. The `!` is the logical NOT operator.

6.7.1 Combining conditions

You can combine conditions with `&` (AND), `|` (OR), or by adding multiple conditions separated by commas (which dplyr treats as AND):

```
# Keep only Burnaby South rows with measured O3
burnaby_o3 <- airdata_step3 |>
  filter(location == "Burnaby South",
         !is.na(O3))
nrow(burnaby_o3)
```

```
[1] 732
```

Note the **double-equals `==`** for "is equal to" — single `=` would be an assignment.

6.7.2 Common filter patterns

You want	Filter expression
One specific value	`filter(col == "Burnaby South")`
Not equal	`filter(col != "Burnaby South")`
One of several	`filter(col %in% c("A", "B", "C"))`
Numeric threshold	`filter(O3 > 10)`

You want	Filter expression
Between two values	`filter(O3 >= 10, O3 < 20)`
Non-missing	`filter(!is.na(O3))`
Date range	`filter(date >= "2000-01-15")`

The right-hand column reads naturally in English. *"Filter the data where location is in 'A', 'B', or 'C'."* That's the dplyr design.

6.8 Verb 3 — `mutate()` (add or change columns)

`mutate()` adds new columns or replaces existing ones. The most common use in this chapter is **parsing the text date column into a real R date**.

```
airdata_step4 <- airdata_step3 |>
  mutate(date_parsed = mdy(Date))
glimpse(airdata_step4 |> select(Date, date_parsed))
```

```
Rows: 5,677
Columns: 2
$ Date        <chr> "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000"~
$ date_parsed <date> 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-0~
```

`mdy()` from the **lubridate** package reads a text date in *month/day/year* order and returns a proper `<date>` column. Other parsers:

Source format	lubridate function
`"1/15/2000"` (US)	`mdy()`
`"15/1/2000"` (UK / Canadian common)	`dmy()`
`"2000-01-15"` (ISO 8601)	`ymd()`
`"15-Jan-2000"`	`dmy()` (handles abbreviations)

> ⚠ Date ambiguity
>
> A string like `"3/4/2000"` could be March 4 or April 3 depending on convention. **lubridate trusts the function name** — call `mdy()` and it interprets as March 4; call `dmy()` and it interprets as April 3.
> Always check: pick one row, look at the original `Date` string, and confirm the parsed date is what you expected. A wrong date parser is a silent bug that ruins every later analysis.

6.8.1 Multiple new columns at once

```
airdata_step5 <- airdata_step4 |>
  mutate(date_parsed = mdy(Date),
         year = year(date_parsed),
         month = month(date_parsed),
         day = day(date_parsed),
         o3_kg_per_m3 = O3 / 1000)
glimpse(airdata_step5 |> select(Date, date_parsed, year, month, day, o3_kg_per_m3))
```

```
Rows: 5,677
Columns: 6
$ Date         <chr> "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000~
$ date_parsed  <date> 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-~
$ year         <dbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 200~
$ month        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ day          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ o3_kg_per_m3 <dbl> 0.001, 0.003, 0.018, 0.032, 0.031, 0.032, 0.028, 0.019, 0~
```

`year()`, `month()`, `day()` are lubridate accessors. Each returns a vector of the same length.

6.9 Verb 4 — `arrange()` (sort rows)

`arrange()` sorts the tibble:

```
airdata_step5 |>
  arrange(O3) |>
  select(location, date_parsed, O3) |>
  head(5)
```

```
# A tibble: 5 x 3
  location      date_parsed    O3
  <chr>         <date>      <dbl>
1 Burnaby South 2000-01-02      0
2 Burnaby South 2000-01-02      0
3 Burnaby South 2000-01-02      0
4 Burnaby South 2000-01-04      0
5 Burnaby South 2000-01-04      0
```

Ascending by default. For descending, wrap in `desc()`:

```
airdata_step5 |>
  arrange(desc(O3)) |>
```

```
    select(location, date_parsed, O3) |>
    head(5)
```

```
# A tibble: 5 x 3
  location                    date_parsed    O3
  <chr>                       <date>      <dbl>
1 Port Moody Rocky Point Park 2000-01-09     39
2 Langley Central             2000-01-09     38
3 Port Moody Rocky Point Park 2000-01-09     38
4 Port Moody Rocky Point Park 2000-01-09     38
5 Langley Central             2000-01-09     37
```

The top of the list is the highest-O_3 hourly readings — useful for spotting the most polluted moments.

6.9.1 Multi-key sort

```
  airdata_step5 |>
    arrange(location, desc(O3)) |>
    select(location, date_parsed, O3) |>
    head(8)
```

```
# A tibble: 8 x 3
  location      date_parsed    O3
  <chr>         <date>      <dbl>
1 Burnaby South 2000-01-17     35
2 Burnaby South 2000-01-02     33
3 Burnaby South 2000-01-09     33
4 Burnaby South 2000-01-09     33
5 Burnaby South 2000-01-09     33
6 Burnaby South 2000-01-09     33
7 Burnaby South 2000-01-17     33
8 Burnaby South 2000-01-01     32
```

Sorts by `location` (alphabetical), then within each location by `O3` descending. The equivalent in Excel was **Data → Sort → Add Level**.

6.10 Verb 5 — `case_when()` (the R `IFS`)

`case_when()` recodes values into categories based on rules. It is the direct R equivalent of Excel's `IFS`.

```r
airdata_step6 <- airdata_step5 |>
  mutate(o3_band = case_when(
    is.na(O3)     ~ "missing",
    O3 < 10       ~ "low",
    O3 < 20       ~ "moderate",
    O3 < 30       ~ "high",
    O3 >= 30      ~ "very high",
    TRUE            ~ "other"
  ))
airdata_step6 |> count(o3_band)
```

```
# A tibble: 4 x 2
  o3_band       n
  <chr>     <int>
1 high        905
2 low        3327
3 moderate   1013
4 very high   432
```

Read it line by line: *"if O3 is missing, return 'missing'; otherwise if it is less than 10, return 'low'; otherwise less than 20, 'moderate'; …"*. The first match wins. The trailing `TRUE ~ "other"` is the catch-all, like the `TRUE` in Excel's `IFS`.

> **Why `case_when` beats nested `if_else`**
>
> You could write the same logic with nested `if_else()`:
>
> ```r
> if_else(O3 < 10, "low",
>   if_else(O3 < 20, "moderate",
>     if_else(O3 < 30, "high", "very high")))
> ```
>
> It works, but it nests deeply and is hard to maintain. `case_when` is flat — one row per rule — and easy to extend.

6.11 Verb 6 — `summarise()` overall, `group_by()` for groups

`summarise()` collapses a tibble to **one row** of summary statistics:

```r
airdata_step6 |>
  summarise(
    n_obs = n(),
    mean_o3 = mean(O3, na.rm = TRUE),
    median_o3 = median(O3, na.rm = TRUE),
    max_o3 = max(O3, na.rm = TRUE)
  )
```

```
# A tibble: 1 x 4
  n_obs mean_o3 median_o3 max_o3
  <int>   <dbl>     <dbl>  <dbl>
1  5677    10.3         6     39
```

The function `n()` (with no arguments, only used inside `summarise()` or `mutate()`) returns the row count.

Add `group_by()` *before* `summarise()` to get one row per group:

```
airdata_step6 |>
  group_by(location) |>
  summarise(
    n_obs = n(),
    n_with_o3 = sum(!is.na(O3)),
    mean_o3 = mean(O3, na.rm = TRUE)
  ) |>
  arrange(desc(mean_o3))
```

```
# A tibble: 8 x 4
  location                          n_obs n_with_o3 mean_o3
  <chr>                             <int>     <int>   <dbl>
1 Langley Central                     732       732   17.7
2 Maple Ridge Golden Ears School      564       564   11.3
3 Richmond South                      732       732   10.5
4 North Delta                         731       731   10.3
5 Vancouver International Airport #2  727       727   10.2
6 Burnaby South                       732       732    9.05
7 Vancouver Kitsilano                 731       731    7.49
8 Port Moody Rocky Point Park         728       728    6.13
```

This is the **PivotTable equivalent** from Chapter 3. Same operation. Same answer. Now scriptable.

> Always remember `na.rm = TRUE`
>
> When summarising a column that *might* have missing values, add `na.rm = TRUE` to every aggregation function: `mean()`, `sum()`, `min()`, `max()`, `median()`, `sd()`. Forgetting it returns `NA` for any group with even one missing value, which is rarely what you want.

6.11.1 `count()` — the most common summarise

`count(col)` is a shortcut for `group_by(col) |> summarise(n = n())`:

```
airdata_step6 |>
  count(location, sort = TRUE)
```

```
# A tibble: 8 x 2
  location                              n
  <chr>                             <int>
1 Burnaby South                       732
2 Langley Central                     732
3 Richmond South                      732
4 North Delta                         731
5 Vancouver Kitsilano                 731
6 Port Moody Rocky Point Park         728
7 Vancouver International Airport #2  727
8 Maple Ridge Golden Ears School      564
```

`sort = TRUE` orders the result descending by count. Useful for spotting which categories dominate.

6.12 Verb 7 — `left_join()` (the R `VLOOKUP`)

A real-world cleaning task often needs a **lookup**: take a code column, attach a description from a reference table. `left_join` does this without the silent failures of `VLOOKUP`.

Suppose we have a small reference tibble mapping stations to regions:

```
station_regions <- tribble(
  ~location,                          ~region,
  "Burnaby South",                    "Burrard Peninsula",
  "Coquitlam Douglas College",        "North-East Sector",
  "Langley Central",                  "Fraser Valley West",
  "Maple Ridge Golden Ears School",   "North-East Sector",
  "New Westminster Sapperton Park",   "Burrard Peninsula",
  "North Delta",                      "South of Fraser",
  "North Vancouver Second Narrows",   "North Shore",
  "Pitt Meadows Airport",             "North-East Sector",
  "Port Coquitlam North",             "North-East Sector",
  "Port Moody Rocky Point Park",      "North-East Sector"
)
```

Attach the `region` column to the data:

```
airdata_step7 <- airdata_step6 |>
  left_join(station_regions, by = "location")
airdata_step7 |>
  select(location, region, date_parsed, O3) |>
  head(5)
```

```
# A tibble: 5 x 4
  location      region            date_parsed    O3
```

```
  <chr>          <chr>              <date>      <dbl>
1 Burnaby South Burrard Peninsula 2000-01-01      1
2 Burnaby South Burrard Peninsula 2000-01-01      3
3 Burnaby South Burrard Peninsula 2000-01-01     18
4 Burnaby South Burrard Peninsula 2000-01-01     32
5 Burnaby South Burrard Peninsula 2000-01-01     31
```

Every row gets its region. The `by = "location"` argument names the column to match on. Stations not in the reference table get `region = NA` (and the join does *not* drop them — that is what `left` means).

`left_join` vs `inner_join` vs `right_join`

Join	Keeps
`left_join(x, y)`	**All rows of `x`**, plus matched rows from `y` (NA otherwise)
`inner_join(x, y)`	Only rows that match in both
`right_join(x, y)`	All rows of `y`, plus matched rows from `x`
`full_join(x, y)`	All rows from both, NA where no match

For lookups, `left_join` is the default — it keeps every row in your main data, even if the lookup misses.

6.13 Putting it all together — the full cleaning pipeline

Now the entire chapter in one pipeline:

```
airdata_clean <- read_csv(here("data", "airdata3.csv"),
                          show_col_types = FALSE) |>
  # Drop redundant index columns
  select(-starts_with("Unnamed"), -any_of("...1")) |>
  # Drop columns that are 100% missing
  select(where(\(col) !all(is.na(col)))) |>
  # Parse the date and add date parts
  mutate(date_parsed = mdy(Date),
         year = year(date_parsed),
         month = month(date_parsed),
         day = day(date_parsed)) |>
  # Keep only rows with a measured O3
  filter(!is.na(O3)) |>
  # Classify O3 into bands
  mutate(o3_band = case_when(
    O3 < 10  ~ "low",
```

```
      O3 < 20  ~ "moderate",
      O3 < 30  ~ "high",
      TRUE       ~ "very high"
    )) |>
    # Attach region info
    left_join(station_regions, by = "location") |>
    # Final sort
    arrange(date_parsed, location)
```

```
New names:
* `` -> `...1`
```

```
  dim(airdata_clean)
```

```
[1] 5677   17
```

```
  glimpse(airdata_clean)
```

```
Rows: 5,677
Columns: 17
$ PM10        <dbl> 13, 6, 2, 2, 4, 2, 4, 6, 6, 8, 7, 4, 3, 3, 6, 6, 6, 5, 8, ~
$ SO2         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ CO          <dbl> 0.5, 0.4, 0.4, 0.3, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.4, 0.4~
$ NO          <dbl> 10, 4, 2, 1, 1, 1, 1, 1, 2, 5, 4, 3, 3, 3, 3, 2, 3, 2, 5, ~
$ NO2         <dbl> 22, 21, 19, 8, 9, 9, 12, 19, 20, 23, 17, 10, 9, 8, 8, 10, ~
$ PM25        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ O3          <dbl> 1, 3, 18, 32, 31, 32, 28, 19, 17, 13, 15, 20, 19, 21, 21, ~
$ location    <chr> "Burnaby South", "Burnaby South", "Burnaby South", "Burnab~
$ Date        <chr> "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000"~
$ Time        <time> 01:00:00, 02:00:00, 03:00:00, 04:00:00, 05:00:00, 06:00:0~
$ TRS         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ date_parsed <date> 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-0~
$ year        <dbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000~
$ month       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ day         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ o3_band     <chr> "low", "low", "moderate", "very high", "very high", "very ~
$ region      <chr> "Burrard Peninsula", "Burrard Peninsula", "Burrard Peninsu~
```

One pipeline. Eight steps. Reads top-to-bottom in English. Reproducible. Documented. This is what cleaning data in R looks like once you are fluent.

6.13.1 Save the cleaned tibble

```
write_csv(airdata_clean,
          here("outputs", "airdata3_clean.csv"))
```

The cleaned file is your **Expected output** for the chapter assignment. The raw `data/airdata3.csv` stays untouched.

6.14 The cleaning log

In Excel you kept a cleaning log on a separate sheet. In Quarto you keep it as **Markdown text between the code chunks**.

For this chapter, the log might look like:

```
## Cleaning log

| Date | Step | What I did | Why |
|---|---|---|---|
| 2026-09-22 | 1 | Dropped 2 redundant index columns (`Unnamed: 0`, etc.) | Added by an upst
| 2026-09-22 | 2 | Dropped 12 columns that were 100% missing | No measurements at any statio
| 2026-09-22 | 3 | Parsed `Date` from text to `<date>` using `lubridate::mdy()` | Original w
| 2026-09-22 | 4 | Added `year`, `month`, `day` columns | Convenience for later grouping. |
| 2026-09-22 | 5 | Filtered to rows where O3 was measured | Reduced rows from 11,904 to 742.
| 2026-09-22 | 6 | Recoded O3 into 4 bands using `case_when` | For the air-quality summary t
| 2026-09-22 | 7 | Left-joined a station→region reference table | To enable regional summari
| 2026-09-22 | 8 | Arranged by date then location | Final sort. |
```

That table goes in your `.qmd` directly above the cleaning pipeline. Anyone reading the rendered report sees what you did, why, and the code that did it.

6.15 Active learning

6.15.1 Activity 1 — Find the dirty columns

Run `colSums(is.na(airdata_raw)) |> sort(decreasing = TRUE)`. How many columns are 100% missing? How many are partially missing?

6.15.2 Activity 2 — Filter and count

How many rows did the file have before filtering for non-missing O3? How many after? What fraction of rows did you keep?

6.15.3 Activity 3 — Try a different case_when

Replace the four-band O3 classification with a three-band version (`good` / `moderate` / `unhealthy`). Re-run the pipeline. How does the count of each category change?

6.15.4 Activity 4 — Add a new station to the lookup

Add one row to `station_regions` for a station you noticed in the data but is missing from the reference. Re-run the `left_join`. Confirm the new station now has a region.

6.15.5 Activity 5 — Save and re-import

Save your cleaned tibble to `outputs/airdata3_clean.csv`. Then in a fresh R session, import it with `read_csv()`. Confirm the date column is still `<date>` (not `<chr>`). If it is, your earlier parsing held.

6.16 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.
2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?
4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

Lab: Clean the air-quality file together

Work in groups of 3 or 4. Each member should be inside their own `frst232/` RStudio Project. The file `airdata3.csv` must be in each member's `data/` folder.
Task 1 — Inspect together. Each member runs the six-move inspection (`dim`, `names`, `glimpse`, `summary`, `colSums(is.na(...))`). Compare what you see. How many columns have any data? How many are 100% missing?
Task 2 — Drop redundant columns. As a group, decide which columns to drop. Write the `select()` call together.
Task 3 — Parse the date. Try `mdy(Date)`. Did it work for every row, or did some fail? (Check with `summary(parsed_date)` — any `NA` count from parsing.)
Task 4 — Filter and group. Write a `filter()` + `group_by()` + `summarise()` pipeline that returns mean O_3 per station. **Sort descending** by mean O_3.
Task 5 — Discuss the result. Which station has the highest mean O_3 in this month? Does that make sense given what the group members know about Metro Vancouver geography?
Task 6 — Submit. Each group submits: - the final pipeline code (paste into a Quarto chunk in your shared learner file), - the station-ranked summary table, - a one-sentence interpretation of the result.
This lab fits comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 6 reference solution (rendered HTML) — the completed, rendered solution for this demo lab. Open it **after** your own attempt and use it to check your work. *This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

6.17 Exercises

These are the take-home exercises for this chapter.

1. **Read and inspect.** Import `airdata3.csv`. Report the row count, column count, and the number of all-NA columns.
2. **Clean step by step.** Build the pipeline yourself, one step at a time, in separate chunks. After each step, print `nrow()` and `ncol()`. The final cleaned tibble should have 742 rows (rows with O_3 measured).
3. **Per-station summary.** Group by `location` and compute `n_obs`, `mean_o3`, `median_o3`, `max_o3`. Sort descending by `mean_o3`.
4. **Per-day summary.** Group by your parsed `date_parsed` column and compute mean O_3 across all stations per day. Which day in January 2000 had the highest provincial mean O_3?

5. **Add a category.** Use `case_when()` to add a column `aq_alert` that is `"alert"` if O_3 25 and `"normal"` otherwise. How many `"alert"` hours are in the file?
6. **Lookup challenge.** Build a small `station_lat_lon` tibble with at least 5 stations and made-up latitude/longitude pairs. `left_join` it onto your cleaned tibble. Confirm the join did not drop any rows.
7. **Cleaning log.** Write a cleaning log Markdown table in your `.qmd`, with one row per step you did in Exercise 2.
8. **Optional, harder.** Use `pivot_wider()` to reshape your per-station summary so each station is a column. (We will meet `pivot_wider` formally in Chapter 8 — try it now if you want to look ahead.)

6.18 Optional, advanced

6.18.1 Renaming columns

`rename()` lets you give a column a new name:

```
airdata_step1 |>
  rename(station = location,
         o3_ugm3 = O3)
```

6.18.2 `if_else()` for binary recodes

When you only need two outcomes, `if_else()` is briefer than `case_when()`:

```
mutate(aq_alert = if_else(O3 >= 25, "alert", "normal"))
```

6.18.3 `across()` for applying a function to many columns

If you want the mean of *every* numeric column grouped by station, you can use `across()`:

```
airdata_clean |>
  group_by(location) |>
  summarise(across(where(is.numeric), \(x) mean(x, na.rm = TRUE)))
```

This is one of the most powerful patterns in dplyr — but also one of the most confusing. We will use it sparingly in this course.

6.19 A glimpse ahead: From cleaning to summarizing

In Chapter 7 we go deeper into `summarise()`. The verbs are already familiar; what we add is:

- Multiple summary statistics in one call.
- `group_by()` with two or more grouping variables.
- `count()` and `tally()` shortcuts.
- Reshaping summary output with `pivot_wider()` for cross-tabs.
- The connection to the Excel **PivotTable** from Chapter 3 — same answer, now scriptable.

```
airdata_clean |>
  group_by(region, o3_band) |>
  summarise(n = n(), .groups = "drop") |>
  pivot_wider(names_from = o3_band, values_from = n, values_fill = 0)
```

That's a cross-tabulation: rows = region, columns = O_3 band, values = count. Equivalent to a PivotTable with `region` in Rows, `o3_band` in Columns, and `Count of n_obs` in Values — but as one line of dplyr.

Chapter 7 walks through this in detail.

6.20 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

6.21 AI as a debugging companion

Useful prompts for this chapter

- *"In R, my `case_when()` returns NA for some rows even though I have a TRUE catch-all at the end. What might be wrong?"*
- *"Explain step by step what this pipeline does: `data |> filter(!is.na(O3)) |> group_by(location) |> summarise(mean_o3 = mean(O3))`"*
- *"I parsed a date with `mdy()` but the result is `NA` for some rows. The original strings look like '1/15/2000'. What are the most common reasons mdy() returns NA?"*

6.21.1 A cleaning prompt template

When you ask AI for cleaning help, include four things:

1. The output of `glimpse(your_tibble)`.
2. The cleaning step you are trying to do (in plain English).
3. The dplyr code you tried.

4. The output (or error) you got.

Without these four, AI's suggestion will be too generic to use directly.

6.21.2 When *not* to use AI

- Do not let AI write your cleaning log. The log records *your* decisions about *your* data.
- Do not let AI invent column names that are not in your data (a common hallucination). Cross-check every suggested column name against `names(your_tibble)`.

6.22 Reading

- *R for Data Science* — chapter 4 *Data transformation*, the canonical dplyr reference. Free at https://r4ds.hadley.nz/.
- The dplyr cheatsheet (PDF): https://rstudio.github.io/cheatsheets/data-transformation.pdf
- The lubridate cheatsheet (PDF): https://rstudio.github.io/cheatsheets/lubridate.pdf
- BC Air Data background: https://www2.gov.bc.ca/gov/content/environment/air-land-water/air/air-quality/air-data

Instructor notes

Pacing and emphasis

- The chapter has **seven new R concepts** (the six core verbs plus `case_when` / `left_join`). Plan to teach the verbs in two clusters: row-and-column (filter, select, arrange) in one session, transform-and-summarize (mutate, case_when, group_by + summarise, left_join) in the next.
- The **Excel-to-dplyr bridge** table at the top of the chapter is the most important content. Refer to it whenever a learner asks why R has a particular verb.
- The **cleaning log** habit is the chapter's most transferable skill. Insist learners write one for every cleaning task, starting with this chapter.

6.22.1 Expected output checklist (matches the top-of-chapter callout)

1. `outputs/airdata3_clean.csv` (saved tibble).
2. `frst232_ch06_learner.html` (rendered Quarto report).
3. Station-ranked summary table.
4. Group-lab worksheet or screenshot.

6.22.2 Materials provided alongside this chapter

File	Purpose
`airdata3.csv`	The raw BC air-quality file. Place in `data/`.

`frst232_ch06_learner.qmd`	Quarto starter with TODO chunks for each verb.
`frst232_ch06_solutions.qmd`	Completed solutions. **Do not distribute to learners.**
`quiz_ch06.html`	Standalone interactive quiz.

6.22.3 Common stumbling points

- **Confusing `filter` with `select`.** Walk through "rows vs columns" explicitly. *"`filter` chooses some of the rows. `select` chooses some of the columns."*
- **`==` vs `=`.** Learners write `filter(location = "Burnaby")` and get an error. Insist on `==` for comparison.
- **`%in%` confusion.** Some learners try `filter(location == c("A", "B"))` and get unexpected results (it does element-wise comparison). Teach `%in%` explicitly.
- **Date parsing failures.** Calling `mdy()` on a column that's in `dmy()` format silently returns wrong dates. Walk through the format-checking habit.
- **Forgetting `na.rm = TRUE`.** Resulting in summaries full of `NA`. Emphasise that for any column that might be missing, add `na.rm = TRUE` to every aggregation.
- *Excel-to-R bridge:* When a learner asks how to do an Excel cleaning move, point them to the bridge table at the top of this chapter. Almost every Excel cleaning question has a one-line dplyr answer.

Part IV

Summarise, combine, visualize

7 Summarize Forestry Data Overall and by Group

Data for this chapter

This chapter uses the **BC air-quality** file (`airdata3.csv`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Write an **overall summary** of one or more columns with `summarise()` — counts, means, medians, standard deviations, ranges.
- Write a **by-group summary** by adding `group_by()` before `summarise()`.
- Group by **multiple columns** at once (e.g., region × day) to build cross-tabulations.
- Use the **shortcut functions** `count()`, `tally()`, and `n_distinct()` for common patterns.
- Reshape a summary with **`pivot_wider()`** so each group level becomes a column — the closest R has to a PivotTable.
- Recognize that a `group_by() |> summarise()` pipeline is the conceptual equivalent of an Excel **PivotTable** from Chapter 3 — and produce identical numbers.
- Use **`across()`** to apply the same summary to many columns at once.
- Write summary tables that read cleanly in a rendered Quarto report.

Expected output for this chapter

What you will hand in

1. A **rendered Quarto HTML report** (`frst232_ch07_learner.html`) containing:
   - an **overall summary** of O_3 across the whole cleaned air-quality file,
   - a **by-station** summary table (one row per station),
   - a **by-region × by-day** summary table reshaped with `pivot_wider()` so each region is a column,
   - one short prose paragraph interpreting which station and which day had the highest mean O_3.
2. The corresponding **`.qmd` source file**.
3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

This list is the same as the rubric for the chapter assignment. Keep it in view while you work.

7.1 Where we are

In Chapter 6 you cleaned `airdata3.csv` with the six dplyr verbs. The cleaned tibble (`airdata_clean`) is the starting point for this chapter. Chapter 7 picks up where Chapter 6 leaves off: **now that the data is clean, how do you turn it into numbers your reader can use?**

Chapter 6 introduced `group_by() |> summarise()` briefly. This chapter is that pattern at full depth.

7.2 Excel-to-R bridge — summarising

What you did in Excel	What you do in R
`=AVERAGE(C2:C743)` over a column	`summarise(mean = mean(O3, na.rm = TRUE))`
`=MEDIAN(C2:C743)`	`summarise(median = median(O3, na.rm = TRUE))`
`=COUNT(C2:C743)`	`summarise(n_obs = n())`
`=SUMIF(loc, "Burnaby", o3)`	`filter(location == "Burnaby") |> summarise(sum = sum(O3))`
`=AVERAGEIF(loc, "Burnaby", o3)`	`filter(...) |> summarise(mean = mean(O3))`
`=SUMIFS(...)` multiple criteria	`filter(...) |> summarise(...)` with multiple conditions
PivotTable (rows = station)	`group_by(location) |> summarise(...)`
PivotTable (rows × columns)	`group_by(region, day) |> summarise(...) |> pivot_wider(...)`
PivotTable count by category	`count(category)`
PivotTable refresh after data change	Re-render the document

The right-hand column is the rest of this chapter. **Every Excel summary move from Chapters 2 and 3 has a one-line dplyr equivalent.**

7.3 The starting point — the cleaned tibble

This chapter continues directly from Chapter 6. **Before you run anything else, run the setup block below.** It loads the packages this chapter needs — including `knitr` (for `kable()`) and `scales`, which `library(tidyverse)` does *not* load — and rebuilds the cleaned, region-joined

`airdata_clean` tibble from the Chapter 6 pipeline. If you skip it, later code fails with *"could not find function"* or fills the `region` column with `NA`.

```
library(tidyverse)
library(lubridate)
library(here)
library(knitr)
library(scales)

# Rebuild the cleaned tibble from Chapter 6 --------------------------------
airdata_raw <- read_csv(here("data", "airdata3.csv"),
                        show_col_types = FALSE)

# Station-to-region lookup (same table used in Chapter 6)
station_regions <- tribble(
  ~location,                          ~region,
  "Burnaby South",                    "Burrard Peninsula",
  "Coquitlam Douglas College",        "North-East Sector",
  "Langley Central",                  "Fraser Valley West",
  "Maple Ridge Golden Ears School",   "North-East Sector",
  "New Westminster Sapperton Park",   "Burrard Peninsula",
  "North Delta",                      "South of Fraser",
  "North Vancouver Second Narrows",   "North Shore",
  "Pitt Meadows Airport",             "North-East Sector",
  "Port Coquitlam North",             "North-East Sector",
  "Port Moody Rocky Point Park",      "North-East Sector",
  "Richmond South",                   "South of Fraser",
  "Tsawwassen",                       "South of Fraser",
  "Vancouver Clark Drive",            "Vancouver",
  "Vancouver International Airport #2","South of Fraser",
  "Vancouver Kitsilano",              "Vancouver",
  "West Vancouver Lions Gate",        "North Shore"
)

airdata_clean <- airdata_raw |>
  select(-starts_with("Unnamed"), -any_of("...1")) |>
  select(where(\(col) !all(is.na(col)))) |>
  mutate(date_parsed = mdy(Date),
         year = year(date_parsed),
         month = month(date_parsed),
         day = day(date_parsed)) |>
  filter(!is.na(O3)) |>
  mutate(o3_band = case_when(
    O3 < 10  ~ "low",
    O3 < 20  ~ "moderate",
    O3 < 30  ~ "high",
    TRUE       ~ "very high"
```

```
  )) |>
  left_join(station_regions, by = "location")
```

Confirm it is in memory:

```
glimpse(airdata_clean)
```

```
Rows: 5,677
Columns: 17
$ PM10        <dbl> 13, 6, 2, 2, 4, 2, 4, 6, 6, 8, 7, 4, 3, 3, 6, 6, 6, 5, 8, ~
$ SO2         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ CO          <dbl> 0.5, 0.4, 0.4, 0.3, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.4, 0.4~
$ NO          <dbl> 10, 4, 2, 1, 1, 1, 1, 1, 2, 5, 4, 3, 3, 3, 3, 2, 3, 2, 5, ~
$ NO2         <dbl> 22, 21, 19, 8, 9, 9, 12, 19, 20, 23, 17, 10, 9, 8, 8, 10, ~
$ PM25        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ O3          <dbl> 1, 3, 18, 32, 31, 32, 28, 19, 17, 13, 15, 20, 19, 21, 21, ~
$ location    <chr> "Burnaby South", "Burnaby South", "Burnaby South", "Burnab~
$ Date        <chr> "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000", "1/1/2000"~
$ Time        <time> 01:00:00, 02:00:00, 03:00:00, 04:00:00, 05:00:00, 06:00:0~
$ TRS         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ date_parsed <date> 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-01, 2000-01-0~
$ year        <dbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000~
$ month       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ day         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ o3_band     <chr> "low", "low", "moderate", "very high", "very high", "very ~
$ region      <chr> "Burrard Peninsula", "Burrard Peninsula", "Burrard Peninsu~
```

5677 rows × 17 columns, all with a measured O_3 value, parsed dates, a O_3 band, and a region.

7.4 Overall summary — one row

The simplest case: collapse the whole tibble to one row of summary statistics:

```
airdata_clean |>
  summarise(
    n_obs       = n(),
    mean_o3   = mean(O3, na.rm = TRUE),
    median_o3 = median(O3, na.rm = TRUE),
    sd_o3     = sd(O3, na.rm = TRUE),
    min_o3    = min(O3, na.rm = TRUE),
    max_o3    = max(O3, na.rm = TRUE)
  )
```

```
# A tibble: 1 x 6
```

```
  n_obs mean_o3 median_o3 sd_o3 min_o3 max_o3
  <int>   <dbl>     <dbl> <dbl>  <dbl>  <dbl>
1  5677    10.3         6  10.8      0     39
```

Every column in the result is a single number. The arguments to `summarise()` are named: `<name> = <expression>`.

7.4.1 The functions you will use most

Function	Returns	Notes
`n()`	row count	No arguments. Only inside `summarise()` / `mutate()`.
`sum()`	total	Add `na.rm = TRUE` if missing values are possible.
`mean()`	arithmetic mean	Same.
`median()`	middle value	Robust to outliers.
`min()` / `max()`	extremes	Sensitive to outliers.
`sd()`	standard deviation	Spread around the mean.
`n_distinct()`	unique values	Useful as `n_distinct(location)`.
`quantile()`	any percentile	`quantile(O3, 0.9)` for the 90th.
`IQR()`	interquartile range	Robust spread.

i `mutate()` vs `summarise()` — don't mix them up

Chapter 6 used `mutate()` to add or change columns; this chapter uses `summarise()` to collapse rows into summary values.

Function	What it does	Rows out
`mutate()`	Adds or changes a column	**Same** number of rows
`summarise()`	Computes summary values	**Fewer** rows — one overall, or one per group

7.5 By-group summary — one row per group

i AI prompt to check your work

Use this **after** your own attempt. It should check your reasoning, not hand you the answer:

> *"I wrote `df |> group_by([group]) |> summarise([stat])`. Check whether `group_by()` + `summarise()` is the efficient choice here, whether I should report*

> *`n()` beside each summary, and how to confirm the group sizes are plausible. Do not write the final pipeline for me."*

Add `group_by()` *before* `summarise()`:

```
station_summary <- airdata_clean |>
  group_by(location) |>
  summarise(
    n_obs       = n(),
    mean_o3   = mean(O3, na.rm = TRUE),
    median_o3 = median(O3, na.rm = TRUE),
    max_o3    = max(O3, na.rm = TRUE),
    .groups = "drop"
  ) |>
  arrange(desc(mean_o3))

station_summary
```

```
# A tibble: 8 x 5
  location                          n_obs mean_o3 median_o3 max_o3
  <chr>                             <int>   <dbl>     <dbl>  <dbl>
1 Langley Central                     732   17.7         19     38
2 Maple Ridge Golden Ears School      564   11.3         10     35
3 Richmond South                      732   10.5          5     37
4 North Delta                         731   10.3          6     37
5 Vancouver International Airport #2  727   10.2          4     37
6 Burnaby South                       732    9.05         5     35
7 Vancouver Kitsilano                 731    7.49         2     37
8 Port Moody Rocky Point Park         728    6.13         2     39
```

This is the dplyr equivalent of an Excel PivotTable with `location` in Rows and `Average of O3` in Values. Same operation. Same answer.

> Always add `.groups = "drop"` (or your code will warn)
>
> dplyr keeps the grouping after `summarise()` by default for backward compatibility, which causes confusing behaviour in downstream pipelines. Explicitly say `.groups = "drop"` to clear the grouping after summarise. Treat it as a habit, not a choice.

7.5.1 Multi-key grouping

Group by two or more variables at once. Each unique combination becomes one row:

```
region_day_summary <- airdata_clean |>
  group_by(region, day) |>
```

```
    summarise(
      mean_o3 = mean(O3, na.rm = TRUE),
      .groups = "drop"
    )

  region_day_summary |> head(10)
```

```
# A tibble: 10 x 3
   region              day mean_o3
   <chr>             <int>   <dbl>
 1 Burrard Peninsula     1   17.2
 2 Burrard Peninsula     2   16.1
 3 Burrard Peninsula     3   14.1
 4 Burrard Peninsula     4   17.4
 5 Burrard Peninsula     5   12.6
 6 Burrard Peninsula     6    5.39
 7 Burrard Peninsula     7    4.21
 8 Burrard Peninsula     8   22.2
 9 Burrard Peninsula     9   23.7
10 Burrard Peninsula    10    6.59
```

The result is in **long format**: one row per (region, day) combination. Useful for plotting (Chapter 9) but hard to scan.

7.6 Reshaping with `pivot_wider()` — the PivotTable equivalent

For human-readable cross-tabulations, reshape the long-format summary into a wide table:

```
  region_day_summary |>
    pivot_wider(
      names_from  = region,
      values_from = mean_o3
    ) |>
    arrange(day)
```

```
# A tibble: 31 x 6
     day `Burrard Peninsula` `Fraser Valley West` `North-East Sector`
   <int>               <dbl>                <dbl>               <dbl>
 1     1               17.2                 24.6                15.1
 2     2               16.1                 21.6                11.9
 3     3               14.1                 15.7                 9.44
 4     4               17.4                 30.7                18.5
 5     5               12.6                 18.7                 3.51
 6     6                5.39                 8.96                9.15
```

```
 7     7                4.21                15.3                5.40
 8     8               22.2                 30.8               21.6
 9     9               23.7                 31.5               21.9
10    10                6.59                19.1               10.4
# i 21 more rows
# i 2 more variables: `South of Fraser` <dbl>, Vancouver <dbl>
```

What we just built is a true cross-tab: one row per day, one column per region, values = mean O_3. **This is what an Excel PivotTable produces** when you put `day` in Rows, `region` in Columns, and `Average of O3` in Values.

7.6.1 `pivot_wider()` arguments

Argument	What it does
`names_from`	the column whose values become new column names
`values_from`	the column whose values fill the cells
`values_fill`	what to put in cells where no data exists (default: NA; often `0` for counts)

When to use long vs wide

- **Long format** (one row per combination) is what `summarise()` and `ggplot2` prefer. Use it for analysis.
- **Wide format** (one column per group) is what humans prefer to read. Use it for tables in a report.

Convert between them with `pivot_longer()` and `pivot_wider()`. The two are inverses.

7.7 Shortcut: `count()` and `tally()`

The two most common summarise patterns:

```
airdata_clean |> count(region, sort = TRUE)
```

```
# A tibble: 5 x 2
  region                n
  <chr>             <int>
1 South of Fraser    2190
2 North-East Sector  1292
3 Burrard Peninsula   732
4 Fraser Valley West  732
5 Vancouver           731
```

```
  airdata_clean |> count(region, o3_band)
```

```
# A tibble: 20 x 3
   region             o3_band       n
   <chr>              <chr>     <int>
 1 Burrard Peninsula  high        110
 2 Burrard Peninsula  low         459
 3 Burrard Peninsula  moderate    136
 4 Burrard Peninsula  very high    27
 5 Fraser Valley West high        248
 6 Fraser Valley West low         189
 7 Fraser Valley West moderate    194
 8 Fraser Valley West very high   101
 9 North-East Sector  high        170
10 North-East Sector  low         845
11 North-East Sector  moderate    232
12 North-East Sector  very high    45
13 South of Fraser    high        301
14 South of Fraser    low        1301
15 South of Fraser    moderate    373
16 South of Fraser    very high   215
17 Vancouver          high         76
18 Vancouver          low         533
19 Vancouver          moderate     78
20 Vancouver          very high    44
```

`count(x)` is exactly `group_by(x) |> summarise(n = n())`. Use it for tallies.

`count(x, y)` cross-tabulates. To pivot it wide:

```
  airdata_clean |>
    count(region, o3_band) |>
    pivot_wider(names_from = o3_band, values_from = n, values_fill = 0)
```

```
# A tibble: 5 x 5
  region             high   low moderate `very high`
  <chr>             <int> <int>    <int>       <int>
1 Burrard Peninsula   110   459      136          27
2 Fraser Valley West  248   189      194         101
3 North-East Sector   170   845      232          45
4 South of Fraser     301  1301      373         215
5 Vancouver            76   533       78          44
```

That's a complete cross-tab in three lines. The Excel equivalent was a PivotTable with `region` in Rows, `o3_band` in Columns, `Count of n` in Values, plus the "Show items with no data" toggle to fill empty cells with 0. Three clicks in Excel; three lines in R; same answer.

7.8 Counting unique values with `n_distinct()`

```
airdata_clean |>
  summarise(
    n_stations = n_distinct(location),
    n_regions  = n_distinct(region),
    n_days     = n_distinct(day)
  )
```

```
# A tibble: 1 x 3
  n_stations n_regions n_days
       <int>     <int>  <int>
1          8         5     31
```

Useful when you want *"how many unique stations report O_3?"* in one line.

To *see* the actual unique values — not just how many — use `unique()` (handy when inspecting a column in Chapter 5 too):

```
unique(airdata_clean$region)
```

```
[1] "Burrard Peninsula"  "Fraser Valley West" "North-East Sector"
[4] "South of Fraser"    "Vancouver"
```

7.9 Multiple statistics with `across()`

When you want the same summary (e.g., mean) on **several columns**, `across()` is the cleaner pattern:

```
airdata_clean |>
  group_by(region) |>
  summarise(
    across(c(O3, PM10, NO2),
           \(x) mean(x, na.rm = TRUE)),
    .groups = "drop"
  )
```

```
# A tibble: 5 x 4
  region                O3  PM10   NO2
  <chr>              <dbl> <dbl> <dbl>
1 Burrard Peninsula   9.05 10.6   26.0
2 Fraser Valley West 17.7   8.15  10.2
3 North-East Sector   8.40 10.9   18.3
4 South of Fraser    10.4  12.4   23.4
5 Vancouver           7.49 12.0   29.8
```

This summarises three pollutant columns at once, one mean per column per region. Far less repetitive than writing three `mean(...)` calls manually.

7.9.1 Multiple statistics on the same column

You can also use `across()` to apply multiple statistics to one column:

```
airdata_clean |>
  group_by(region) |>
  summarise(
    across(O3,
           list(mean = \(x) mean(x, na.rm = TRUE),
                sd   = \(x) sd(x, na.rm = TRUE),
                max  = \(x) max(x, na.rm = TRUE))),
    .groups = "drop"
  )
```

```
# A tibble: 5 x 4
  region             O3_mean O3_sd O3_max
  <chr>                <dbl> <dbl>  <dbl>
1 Burrard Peninsula     9.05  9.54     35
2 Fraser Valley West   17.7  10.3      38
3 North-East Sector     8.40  9.50     39
4 South of Fraser      10.4  11.3      37
5 Vancouver             7.49 10.0      37
```

The result has columns `O3_mean`, `O3_sd`, `O3_max`.

> **`across()` is advanced — use it when it helps**
>
> For two or three columns, plain `summarise(mean1 = mean(x), mean2 = mean(y), ...)` is more readable. For five or more, `across()` shines. Use whichever is clearer for your reader.

7.10 Putting it together — a worked summary report

Here is what a complete summary section of a forestry data report might look like.

7.10.1 Overall

```
overall <- airdata_clean |>
  summarise(
    n_obs       = n(),
    n_stations  = n_distinct(location),
```

```
    n_regions   = n_distinct(region),
    mean_o3   = round(mean(O3, na.rm = TRUE), 1),
    median_o3 = round(median(O3, na.rm = TRUE), 1),
    max_o3    = round(max(O3, na.rm = TRUE), 1)
  )
overall |> kable()
```

n_obs	n_stations	n_regions	mean_o3	median_o3	max_o3
5677	8	5	10.3	6	39

7.10.2 By station

```
airdata_clean |>
  group_by(location) |>
  summarise(
    n_obs     = n(),
    mean_o3 = round(mean(O3, na.rm = TRUE), 1),
    max_o3  = max(O3, na.rm = TRUE),
    .groups = "drop"
  ) |>
  arrange(desc(mean_o3)) |>
  kable()
```

location	n_obs	mean_o3	max_o3
Langley Central	732	17.7	38
Maple Ridge Golden Ears School	564	11.3	35
Richmond South	732	10.5	37
North Delta	731	10.3	37
Vancouver International Airport #2	727	10.2	37
Burnaby South	732	9.1	35
Vancouver Kitsilano	731	7.5	37
Port Moody Rocky Point Park	728	6.1	39

7.10.3 By region × day (wide)

```
airdata_clean |>
  group_by(region, day) |>
  summarise(mean_o3 = round(mean(O3, na.rm = TRUE), 1),
            .groups = "drop") |>
  pivot_wider(names_from = region, values_from = mean_o3,
              values_fill = NA) |>
  arrange(day) |>
```

```
  head(10) |>
  kable()
```

day	Burrard Peninsula	Fraser Valley West	North-East Sector	South of Fraser	Vancouver
1	17.2	24.6	15.1	18.9	15.0
2	16.1	21.6	11.9	19.9	19.2
3	14.1	15.7	9.4	11.3	5.1
4	17.4	30.7	18.5	24.4	17.2
5	12.6	18.7	3.5	16.1	17.8
6	5.4	9.0	9.1	2.4	2.5
7	4.2	15.3	5.4	5.7	1.4
8	22.2	30.8	21.6	27.7	22.1
9	23.7	31.5	21.9	28.3	19.7
10	6.6	19.1	10.4	7.4	2.5

7.10.4 By O_3 band — count cross-tab

```
airdata_clean |>
  count(region, o3_band) |>
  pivot_wider(names_from = o3_band, values_from = n, values_fill = 0) |>
  kable()
```

region	high	low	moderate	very high
Burrard Peninsula	110	459	136	27
Fraser Valley West	248	189	194	101
North-East Sector	170	845	232	45
South of Fraser	301	1301	373	215
Vancouver	76	533	78	44

That entire summary section — four tables — is **fifteen lines of dplyr code** and renders to a polished HTML report. The same output in Excel would be three or four PivotTables, each requiring a manual refresh after every data update. **The dplyr version updates automatically when you re-render.**

7.11 Active learning

7.11.1 Activity 1 — Overall summary

Write a single `summarise()` call that returns the count, mean, median, sd, min, and max of O_3 across the whole cleaned tibble. Compare with the table in the chapter.

7.11.2 Activity 2 — By-station ranking

Group by `location`, summarise `mean_o3`, and arrange descending. Which station has the highest mean O_3? Which has the lowest?

7.11.3 Activity 3 — `count()` shortcut

Use `count(region)` and `count(region, sort = TRUE)`. What is the difference?

7.11.4 Activity 4 — Cross-tabulation

Build a cross-tab where rows are `region`, columns are `o3_band`, and cells are counts. Use `pivot_wider()` with `values_fill = 0` so empty cells show `0` instead of `NA`.

7.11.5 Activity 5 — `across()` for many means

Use `across(c(O3, PM10, NO2), \(x) mean(x, na.rm = TRUE))` inside a `group_by(region) |> summarise(...)` pipeline. How does the output differ from writing three separate mean calls?

7.11.6 Activity 6 — Interpret one summary

Pick any one row of your by-group summary and write **one sentence** that names the group, the statistic, the value, and the unit — e.g. *"The Fraser Valley West region had the highest mean O_3, at about 39 ppb."* A number is only useful once you can say what it means.

7.12 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

This demo lab has **six parts** — work them in order:

1. **Your attempt.** Work through the tasks below.
2. **Reference solution.** Compare against the **worked examples earlier in this chapter** (your public reference). The graded Canvas lab has its own private answer key — do not copy demo solutions into a Canvas submission.
3. **Compare your work with the reference.** Where did it match? Where did it differ? What caused the difference, and how did you fix it?

4. **Write your own AI verification prompt.** Ask an AI to *check* your reasoning, code, and outputs — not to produce the answer for you.
5. **Model AI verification prompt.**

   > *"I am doing a FRST 232 practice demo lab. My dataset is `[file name]` (columns `[columns]`). I attempted `[paste your steps or code]` and got `[paste the output]`. Explain what my work does line by line, tell me whether it answers the task, and list the checks I can run to verify it against the chapter's worked example. Do not give me the final answer."*

6. **Reflection: what changed after checking?** In two or three sentences — did your attempt hold up? what did you fix? what will you check first next time?

Lab: Produce a summary report together

Work in groups of 3 or 4. Each member should already have `airdata_clean` in memory (from Chapter 6, or by re-running the Chapter 6 pipeline at the top of their `frst232_ch07_learner.qmd` setup chunk).
Task 1 — Agree on the question. As a group, pick **one** forestry question your summary report will answer:

- Which BC region had the highest mean O_3 in January 2000?
- Which day in January 2000 was the worst air-quality day, on average, across all stations?
- Which monitoring station had the most "very high" O_3 hours?

Task 2 — Build the summary. Write a single dplyr pipeline that answers your question. Use `group_by()`, `summarise()`, and `arrange()` as needed.
Task 3 — Reshape if useful. If your summary has two grouping variables, use `pivot_wider()` to produce a cross-tab readable in a single screen. If it has one grouping variable, sort it.
Task 4 — Caption it. Write a one-sentence caption above your table. The caption should state the **headline finding** (*"Region X had the highest mean O_3…"*).
Task 5 — Submit. Submit: - the dplyr pipeline (in a Quarto chunk), - the rendered table, - the one-sentence caption, - all group members' names.
This lab fits comfortably inside a single hands-on session.

Reference solution — download

Open the Chapter 7 reference solution (rendered HTML) — the completed, rendered solution for this demo lab. Open it **after** your own attempt and use it to check your work. *This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

7.13 Exercises

These are the take-home exercises for this chapter.

1. **Overall summary.** Write a single `summarise()` call that returns `n_obs`, `n_stations`, `mean_o3`, `median_o3`, `sd_o3`, and `max_o3` for the cleaned air-quality data.
2. **By region.** Group by `region`, summarise mean O_3, arrange descending. Which region has the highest mean? Does the result make sense given Metro Vancouver geography?
3. **By day.** Group by `day` (the day-of-month integer), then summarise `n_obs` and `mean_o3` per day. Which day in January 2000 had the highest provincial mean O_3?
4. **Cross-tabulation.** Build the region × o3_band cross-tab from Activity 4. Use `pivot_wider()` with `values_fill = 0`. Submit the rendered table.
5. **Top station per region.** Group by `region`, then within each region find the station with the highest mean O_3. Hint: `group_by(region) |> slice_max(mean_o3, n = 1)` after you have already computed per-station means.
6. **`across()` for multiple pollutants.** Group by `region` and compute the mean of every pollutant column (`O3`, `PM10`, `O3`, `NO2`, `SO2`, `CO`). Use `across()`.
7. **PivotTable parity.** Save your cleaned tibble to `outputs/airdata3_clean.csv`. Open it in Excel. Build a PivotTable with `region` in Rows, `o3_band` in Columns, `Count of O3` in Values. **Confirm the numbers match your dplyr cross-tab to the last unit.**
8. **Optional, harder.** Compute the 90th percentile of O_3 per station: `summarise(p90 = quantile(O3, 0.9, na.rm = TRUE))`. Which station has the highest 90th-percentile O_3?

7.14 Optional, advanced

7.14.1 `slice_max()` and `slice_min()` — top-N per group

```
airdata_clean |>
  group_by(location) |>
  slice_max(O3, n = 3, with_ties = FALSE)
```

Returns the three highest-O_3 hours at each station.

7.14.2 `summarise()` with custom functions

You can define a helper function and call it inside `summarise()`:

```
robust_mean <- function(x) mean(x, na.rm = TRUE, trim = 0.05)

airdata_clean |>
  group_by(region) |>
  summarise(rmean_o3 = robust_mean(O3), .groups = "drop")
```

The 5% trimmed mean drops the top 5% and bottom 5% before averaging — more robust to outliers than a plain mean.

7.14.3 `summary()` of a tibble vs `summarise()` of a tibble

These two are easily confused:

Function	Returns
`summary(tibble)`	A base-R text dump: five-number summary per column
`summarise(tibble, ...)`	A new tibble with the columns you asked for

Use `summary()` for quick interactive inspection. Use `summarise()` for output that goes into a report.

7.15 A glimpse ahead: From summarising to reshaping and combining

Chapter 7 used `pivot_wider()` for the first time. In Chapter 8 you will meet:

- `pivot_longer()` — the inverse of `pivot_wider()`. Useful when data arrives in wide format but you need long format for analysis.
- `inner_join()`, `right_join()`, `full_join()` — the other three join types, alongside `left_join()` from Chapter 6.
- `bind_rows()` and `bind_cols()` for stacking tibbles.
- The principles of **tidy data** — why one row per observation and one column per variable matter so much.

The verbs in Chapter 8 are the *last* dplyr verbs you need for this course. After Chapter 8, every cleaning, summarising, reshaping, and joining task you face has a one-line dplyr answer.

7.16 Self-assessment quiz

The self-assessment quiz is interactive: it appears in the online (HTML) book, and you can also take it on Canvas. This PDF does not display the interactive quiz questions.

7.17 AI as a debugging companion

Useful prompts for this chapter

- *"In R, my `group_by(region) |> summarise(mean(O3))` returns NA for some groups, even though the column has values. Why?"*
- *"Show me how to convert a long-format summary tibble into a wide-format cross-tab with `pivot_wider()`. Use these column names: …"*
- *"What is the difference between `group_by() |> summarise()` and `group_by() |> mutate()`?"* (Answer: summarise collapses to one row per group; mutate keeps every

row and broadcasts the result across the group.)

7.17.1 When *not* to use AI

- Do not let AI choose what to summarise. The choice of mean vs median vs 90th percentile is an analytical decision — yours to make.
- Do not let AI invent column names that are not in your tibble. Always cross-check with `names(your_tibble)`.

7.18 Reading

- *R for Data Science* — chapter 5 *Data tidying* (covers pivot_wider and pivot_longer), chapter 13 *Quantitative exploratory data analysis*. Free at https://r4ds.hadley.nz/.
- The dplyr cheatsheet (PDF): https://rstudio.github.io/cheatsheets/data-transformation.pdf
- The tidyr cheatsheet (PDF): https://rstudio.github.io/cheatsheets/tidyr.pdf

Instructor notes

Pacing and emphasis

- This chapter is shorter on new verbs than Chapter 6 — only `summarise()` (deeper), `pivot_wider()`, `count()`, `n_distinct()`, and `across()`. Spend the saved time on the **PivotTable parity exercise** (Exercise 7), which makes the Excel-to-R bridge concrete by computing the same number both ways.
- The `.groups = "drop"` habit is worth repeating. Without it, downstream pipelines silently inherit grouping and produce surprising results.
- `across()` is powerful but overwhelming for first-time R users. Mention it once, demo it once, then let learners use whichever pattern (plain summarise vs across) feels clearer.

7.18.1 Expected output checklist (matches the top-of-chapter callout)

1. `frst232_ch07_learner.html` (rendered).
2. `frst232_ch07_learner.qmd` (source).
3. Group-lab worksheet or screenshot.

7.18.2 Materials provided alongside this chapter

File	Purpose
`airdata3.csv`	Same raw file from Ch 6. Already in `data/`.
`frst232_ch07_learner.qmd`	Quarto starter with TODO chunks.
`frst232_ch07_solutions.qmd`	Completed solutions. **Do not distribute.**
`quiz_ch07.html`	Standalone interactive quiz.

7.18.3 Sample numbers (reference)

Quantity	Approximate value
Overall mean O_3 (cleaned tibble)	~10 ppb
Overall median O_3	~6 ppb
Number of unique stations with O_3	6
Number of unique regions in cleaned tibble	4
Highest single-hour O_3 reading	~39 ppb

7.18.4 Common stumbling points

- **Confusing `summarise()` with `summary()`.** Walk through the distinction explicitly in the lecture: one is a verb that returns a new tibble; the other is a base R function that prints a text summary.
- **Forgetting `na.rm = TRUE`.** Causes group-mean columns full of NA. The fix is one argument; the habit takes a few weeks to form.
- **`pivot_wider()` syntax confusion.** `names_from` vs `values_from` is easy to swap. The mnemonic: *names_from* is the column whose **names** become headers; *values_from* is the column whose **values** fill the cells.
- *Excel-to-R bridge:* Exercise 7 (PivotTable parity) is the most important exercise. Insist that learners actually open Excel and verify the numbers match. The visceral "they're identical" moment is the chapter's pedagogical payoff.

8 Reshape and Combine Forestry Datasets Safely

Data for this chapter

This chapter uses the **BC silviculture** (bc_disturbance_reforestation.xlsx), **Ontario forest statistics** (on_forest_statistics_1.xlsx), **BC air-quality** (airdata3.csv), and **ECCC weather** (inter_air_van_weather.csv) files. Click each file name to download it directly, then save it in your `data/` folder. For sources and details, see the Datasets Used in This Book page.

Chapter goals

By the end of this chapter you will be able to:

- Explain **tidy data**: one row per observation, one column per variable, one cell per value.
- Use `pivot_longer()` to reshape a wide tibble (one column per group) into long format (one row per observation).
- Use `pivot_wider()` as the inverse (you met this in Chapter 7) and choose between long and wide thoughtfully.
- Combine two tibbles **side by side** with the four join verbs: `left_join()`, `inner_join()`, `right_join()`, `full_join()`.
- Use the **filtering joins** `semi_join()` and `anti_join()` to keep or drop rows based on whether they match a reference.
- Combine two tibbles **end to end** with `bind_rows()` (stack) and `bind_cols()` (side-by-side without matching).
- Diagnose join problems: missing matches, duplicate keys, mismatched data types.
- Build a multi-source forestry analysis that combines the BC silviculture, Ontario forest stats, BC air-quality, and ECCC weather files.

Expected output for this chapter

What you will hand in

1. A **rendered Quarto HTML report** (`frst232_ch08_learner.html`) containing:
    - A `pivot_longer()` example reshaping a wide BC summary into long format.
    - All four join types demonstrated on a small worked example, with a one-sentence explanation of what each kept or dropped.

- A `bind_rows()` example combining the BC silviculture and Ontario forest stats files into a single national tibble.
- A **many-to-one join** of hourly air quality to daily weather, with the pre- and post-join row counts shown.
- A short **join diagnostic** table showing key match counts.

2. The corresponding **`.qmd` source file**.
3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

8.1 Where we are

Through Chapter 7 you cleaned, summarised, and reshaped one dataset at a time. **Chapter 8 is about combining multiple datasets** — joining a main table to a reference table, stacking files from different provinces, reshaping summary tables for plotting.

These are the verbs that turn a folder of files into a single analysis.

8.2 Tidy data, briefly

Three rules:

1. **One row per observation.**
2. **One column per variable.**
3. **One cell per value.**

Most cleaning headaches come from violations of these rules. The BC silviculture file (Ch 1) is tidy — each row is a fiscal year, each column is a measured quantity. The Ontario file (Ch 2) is also tidy — each row is a region × ownership × seral × forest-type × age combination.

A file where each *column* is a year (e.g., `ha_2018`, `ha_2019`, `ha_2020`) is **not tidy** — the column name encodes a variable (year). You convert it to tidy with `pivot_longer()`.

8.3 Excel-to-R bridge — reshape and combine

Excel pattern	dplyr / tidyr verb
`VLOOKUP` or `XLOOKUP`	`left_join()`
Copy-paste-append from two sheets	`bind_rows()`
Copy-paste-side-by-side from two sheets (same row count)	`bind_cols()` (rarely used)
Unpivot in Power Query	`pivot_longer()`
Pivot in Power Query	`pivot_wider()`
"Keep only matched rows"	`inner_join()`
"Show me rows in A that have no match in B"	`anti_join()`

Excel pattern	dplyr / tidyr verb
"Show me rows in A that DO have a match in B (but don't join)"	`semi_join()`
Compare two sheets to find new entries	`anti_join(new, old, by = "key")`
Find duplicates	`count(key) |> filter(n > 1)`

8.4 `pivot_longer()` — the inverse of `pivot_wider()`

In Chapter 7 you used `pivot_wider()` to turn a summary tibble into a cross-tab. `pivot_longer()` is the reverse: take a wide table and convert it to long format.

8.4.1 A worked example with the BC silviculture file

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")

# Build a small wide summary: rows = fiscal year, columns = disturbance types
bc_wide <- bc |>
  select(Fiscal_Year, Harvested_ha, Natural_Disturbance_ha, Total_Disturbance_ha)
head(bc_wide)
```

```
# A tibble: 6 x 4
  Fiscal_Year Harvested_ha Natural_Disturbance_ha Total_Disturbance_ha
        <dbl>        <dbl>                  <dbl>                <dbl>
1        1987      242182                  20875               263057
2        1988      251557.                  9641.              261199.
3        1989      215878.                 10712.              226590.
4        1990      192692.                 28301.              220992.
5        1991      213217.                 14424.              227640.
6        1992      234086.                  5635.              239720.
```

This is **wide format**: each disturbance type is its own column. To plot all three on the same chart (Chapter 9), you need long format:

```
bc_long <- bc_wide |>
  pivot_longer(
    cols      = c(Harvested_ha, Natural_Disturbance_ha, Total_Disturbance_ha),
    names_to  = "disturbance_type",
    values_to = "area_ha"
  )
head(bc_long, 9)
```

```
# A tibble: 9 x 3
  Fiscal_Year disturbance_type       area_ha
        <dbl> <chr>                    <dbl>
1        1987 Harvested_ha            242182
2        1987 Natural_Disturbance_ha   20875
3        1987 Total_Disturbance_ha    263057
4        1988 Harvested_ha            251557.
5        1988 Natural_Disturbance_ha    9641.
6        1988 Total_Disturbance_ha    261199.
7        1989 Harvested_ha            215878.
8        1989 Natural_Disturbance_ha   10712.
9        1989 Total_Disturbance_ha    226590.
```

What we did:

- `cols` lists which columns to pivot.
- `names_to` is the name of the new column that will hold the original column names.
- `values_to` is the name of the new column that will hold the values.

Result: 3 columns × 37 rows became 3 columns × 111 rows. The data is identical, the shape is different.

When to use long vs wide

- **Long format** is what `ggplot2`, `dplyr`, and most statistics functions want. Use it for analysis.
- **Wide format** is what humans want to read. Use it for tables in a report and for cross-tabs.

Convert between them with `pivot_longer()` and `pivot_wider()`. They are inverses: `data |> pivot_longer(...) |> pivot_wider(...)` returns the original (modulo column ordering).

Which verb should I use?

Reshaping changes the **shape of one table**; joining **combines two tables**. When you meet a new task, match it here first:

Your task	Verb
Wide → long (fold many columns into key/value rows)	`pivot_longer()`
Long → wide (spread a key column across new columns)	`pivot_wider()`
Add columns from another table by a matching key	a **join** (`left_join()`, …)
Stack two tables with the same columns	`bind_rows()`
Glue two tables side by side (same row order)	`bind_cols()`

8.5 The four mutating joins

> **AI prompt to check your work**
>
> Use this **after** your own attempt. It should check your reasoning, not hand you the answer:
>
> > *"I plan to `left_join([A], [B], by = "[key]")` to add `[columns]`. Check whether `left_join` is the right join type, whether my key is unique in `[B]`, and which pre/post-join row-count checks I should run so the join does not silently fan out. Do not give me the merged table."*

8.5.1 Set up a small worked example

```
# Five fictitious silviculture plots
plots <- tribble(
  ~plot_id, ~district,        ~area_ha,
  "P001",   "Vancouver",      12.0,
  "P002",   "Vancouver",       8.5,
  "P003",   "Squamish",       21.2,
  "P004",   "Squamish",       15.0,
  "P005",   "Nanaimo",         9.7
)

# Reference table - district → biogeoclimatic zone
districts <- tribble(
  ~district,    ~bec_zone,
  "Vancouver",  "CWHdm",
  "Squamish",   "CWHvm",
  "Sunshine Coast", "CWHvh"      # district NOT in plots
)

plots
```

```
districts
```

8.5.2 `left_join()` — keep all rows of the left tibble

```
left_join(plots, districts, by = "district")
```

8.5.3 `inner_join()` — keep only matched rows

```
inner_join(plots, districts, by = "district")
```

8.5.4 `right_join()` — keep all rows of the right tibble

```
right_join(plots, districts, by = "district")
```

8.5.5 `full_join()` — keep everything

```
full_join(plots, districts, by = "district")
```

8.5.6 A picture of all four

8.6 The filtering joins

8.6.1 `semi_join()` — "keep rows in A that have a match in B"

```
semi_join(plots, districts, by = "district")
```

8.6.2 `anti_join()` — "keep rows in A that have NO match in B"

```
anti_join(plots, districts, by = "district")
```

8.7 Joining on multiple keys

```
left_join(measurements, reference, by = c("plot_id", "year"))
```

```
left_join(measurements, reference,
          by = c("plot_id" = "plot", "year" = "yr"))
```

8.8 `bind_rows()` — stack tibbles end to end

```
# Pretend we received the BC file as two separate halves
bc_pre2005 <- bc |> filter(Fiscal_Year < 2005)
bc_post2005 <- bc |> filter(Fiscal_Year >= 2005)

nrow(bc_pre2005); nrow(bc_post2005)
```

```
# Stack them
bc_recombined <- bind_rows(bc_pre2005, bc_post2005)
nrow(bc_recombined)
```

```
identical(arrange(bc, Fiscal_Year), arrange(bc_recombined, Fiscal_Year))
```

8.8.1 Tagging the source

```
bc_tagged <- bc |>
  select(Fiscal_Year, Harvested_ha) |>
  mutate(province = "BC")
```

```r
# Build a tiny Ontario-style sample
ontario_sample <- tribble(
  ~Fiscal_Year, ~Harvested_ha,
  2018,         160000,
  2019,         158000,
  2020,         155000
) |>
  mutate(province = "ON")

combined <- bind_rows(bc_tagged, ontario_sample)
combined |> tail(6)
```

8.9 `bind_cols()` — stack side by side (rarely the right answer)

```r
bind_cols(tibble_a, tibble_b)
```

> ⚠ **`bind_cols()` vs `left_join()`**
>
> `bind_cols()` glues by row position; `left_join()` matches by key. **Always prefer `left_join()`** for combining columns from two sources — even if you "know" the rows are in the same order. Order assumptions silently break the day someone re-sorts one of the files.

8.10 Join diagnostics — what every join should print

```r
# 1. How many rows did the join produce vs the original?
nrow(plots)
nrow(left_join(plots, districts, by = "district"))

# 2. How many rows in A failed to match in B?
anti_join(plots, districts, by = "district") |> nrow()
```

8.11 A real-world many-to-one join: air quality + weather

```r
# Hourly air quality at the YVR station (one row per hour)
air_yvr <- read_csv(here("data", "airdata3.csv"),
                    show_col_types = FALSE) |>
  filter(location == "Vancouver International Airport #2",
         !is.na(O3)) |>
  mutate(date = mdy(Date)) |>
  select(location, date, time = Time, O3)
```

```r
# Daily weather at the same airport (one row per day).
# The column names arrive messy, so we rename as we select.
weather <- read_csv(here("data", "inter_air_van_weather.csv"),
                    show_col_types = FALSE) |>
  mutate(date = mdy(`Date/Time`)) |>
  select(date,
         mean_temp    = `Mean Temp (°C)`,
         total_precip = `Total Precip (mm)`)

# Pre-join row counts - always check these first
```

```r
nrow(air_yvr)    # hourly: many rows
```

```r
nrow(weather)    # daily: one row per day
```

```r
# Many-to-one left join: each hour gets its day's weather
air_weather <- air_yvr |>
  left_join(weather, by = "date")

nrow(air_weather)   # post-join row count
```

```r
# Is the weather key unique? (If not, the join can fan out.)
weather |> count(date) |> filter(n > 1) |> nrow()      # expect 0
```

```r
# Did every air reading find a weather match?
air_yvr |> anti_join(weather, by = "date") |> nrow()   # expect 0
```

8.12 A complete worked example — multi-source forestry analysis

```r
# Read all three
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data") |>
  mutate(province = "BC", source = "silviculture")

ontario <- read_excel(here("data", "on_forest_statistics_1.xlsx"),
                      sheet = "Data") |>
  mutate(province = "ON", source = "forest_stats")

# Stack the two harvest-area summaries from different schemas
# (we need to pick comparable columns first)
bc_harvest <- bc |>
  select(province, year = Fiscal_Year, area_ha = Harvested_ha)

ontario_harvest <- ontario |>
  group_by(year = 2021) |>
  summarise(area_ha = sum(SumOfTotalHa, na.rm = TRUE)) |>
  mutate(province = "ON")

combined_harvest <- bind_rows(bc_harvest, ontario_harvest)
combined_harvest |> tail(5)
```

8.13 Active learning

8.13.1 Activity 1 — Pivot longer

8.13.2 Activity 2 — All four joins

8.13.3 Activity 3 — Filtering joins

8.13.4 Activity 4 — Bind rows with tags

8.13.5 Activity 5 — Detect duplicate keys

8.14 Practice Demo Lab

! Practice demo only

This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

1.
2.
3.
4.
5.
6.

i Lab: Combine three forestry sources

Work in groups of 3 or 4. The lab builds one combined tibble from the BC silviculture, Ontario forest stats, and BC air-quality files.

Task 1 — Inventory. Each member loads all three files. As a group, list which columns are comparable across files (e.g., `year`, `area_ha`).
Task 2 — Select comparable columns. Write `select()` calls that produce a uniform schema (e.g., `province`, `year`, `area_ha`) for each of the BC and Ontario files. The air-quality file is the odd one out; pick a different summary axis for it.
Task 3 — Stack with `bind_rows()`. Combine the BC and Ontario harvest summaries into one tibble. Tag each row with the province.
Task 4 — Diagnose. Run `count(province)` to confirm both provinces appear. Run `colSums(is.na(combined))` to check for unexpected NAs after the bind.
Task 5 — Reflect. Discuss: what did you have to *decide* during the column-matching that the code itself cannot decide for you? (Hint: units, time aggregation, scope.)
Task 6 — Submit. Each group submits: - the combined tibble (printed as a `kable` table in the qmd), - a one-paragraph note describing the decisions made in Task 5, - all group members' names.

Reference solution — download

Open the Chapter 8 reference solution (rendered HTML) — the completed, rendered solution for this demo lab. Open it **after** your own attempt and use it to check your work. *This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

8.15 Exercises

1.

2.

3.

4.

5.

6.

7.

8.

8.16 Optional, advanced

8.16.1 `left_join` with multiple matches — fan-out

```
districts_dup <- tribble(
  ~district,    ~bec_zone,
  "Vancouver",  "CWHdm",
  "Vancouver",  "CWHxm"           # duplicate key!
)

left_join(plots, districts_dup, by = "district")
# Now Vancouver plots appear TWICE in the result.
```

```
districts_dedup <- districts_dup |> distinct(district, .keep_all = TRUE)
```

8.16.2 `relationship` argument in dplyr 1.1

```
left_join(plots, districts, by = "district",
          relationship = "many-to-one")
```

8.17 A glimpse ahead: From combined data to charts

8.18 Self-assessment quiz

8.19 AI as a debugging companion

Useful prompts for this chapter

- *"My `left_join` returned 50 rows but the left tibble has 30 rows. What does that mean and how do I fix it?"*
- *"How do I find which rows in tibble A have no matching key in tibble B? Show me with `anti_join`."*
- *"My pivot_longer call returns an error: 'columns must be numeric'. What is wrong?"*

8.20 Reading

- https://r4ds.hadley.nz/
- https://dplyr.tidyverse.org/articles/two-table.html
- https://stat545.com/join-cheatsheet.html

Instructor notes

Pacing and emphasis

- Spend most time on **left_join + anti_join** — together they cover 80% of real join needs. The other join types are easier once those two are solid.
- The **join diagnostics** habit is the most transferable skill in this chapter. Insist learners run row counts before trusting any join.
- `bind_cols()` is mentioned for completeness but **almost always wrong**. Highlight the warning callout.

8.20.1 Materials provided

File	Purpose
`frst232_ch08_learner.qmd`	Quarto starter with TODO chunks.
`frst232_ch08_solutions.qmd`	Completed solutions. **Do not distribute.**
`quiz_ch08.html`	Standalone interactive quiz.

8.20.2 Sample numbers

Quantity	Approximate value
`nrow(bc_wide)`	37
`nrow(bc_long)` after `pivot_longer` on 3 cols	111
Plot–district `left_join` row count	5
Plot–district `inner_join` row count	4
Plot–district `full_join` row count	6
Plot–district `anti_join` row count (plots→districts)	1 (Nanaimo)

8.20.3 Common stumbling points

- **Confusing `left_join` and `bind_rows`.** Walk through *side-by-side (more columns) vs end-to-end (more rows).*
- **Forgetting `by = "..."`.** dplyr will try to guess the join key from common column names — sometimes wrong. Insist on explicit `by` for every join.
- **Fan-out from duplicate keys.** Teach the row-count check before trusting any join.
- **`bind_rows` column-name mismatch.** Same data with different column names will produce extra columns full of NA. Use `rename()` first to harmonise.

9 Visualize Forestry Data with ggplot2

> **Data for this chapter**
>
> This chapter uses the **BC silviculture** (`bc_disturbance_reforestation.xlsx`) and **BC air-quality** (`airdata3.csv`) files. Click each file name to download it directly, then save it in your `data/` folder. For sources and details, see the Datasets Used in This Book page.

Setup — run this first

```
library(tidyverse)   # ggplot2, dplyr, readr, ...
library(readxl)      # read_excel()
library(lubridate)   # date helpers
library(here)        # project-relative paths
library(knitr)       # kable()
library(scales)      # label_comma(), label_percent()
```

Chapter goals

- aesthetic mappings
-
-
-
-
-

-

Expected output for this chapter

What you will hand in

1. A **rendered Quarto HTML report** (`frst232_ch09_learner.html`) containing at least:
   - One **chart** of BC reforestation by fiscal year — a bar or line chart; pick the type that best shows change over time.
   - One **line chart** with two or more series overlaid (e.g., harvested vs natural disturbance vs reforestation across years).
   - One **histogram** of hourly O_3 readings.
   - One **box plot** of O_3 by region.
   - One **faceted** plot (small multiples) showing O_3 over the days of January 2000, faceted by station.
   - At least one chart with **custom labels** (`labs(title, subtitle, x, y, caption)`).
2. One chart saved to `outputs/` as a `.png` using `ggsave()`.
3. A short **group-lab worksheet** (or screenshot) submitted to the course site.

9.1 Where we are

9.2 Excel-chart-to-ggplot bridge

9.3 The grammar of graphics — three sentences

1.
2.

3.

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")

ggplot(data = bc, mapping = aes(x = Fiscal_Year, y = Reforestation_ha)) +
  geom_line()
```

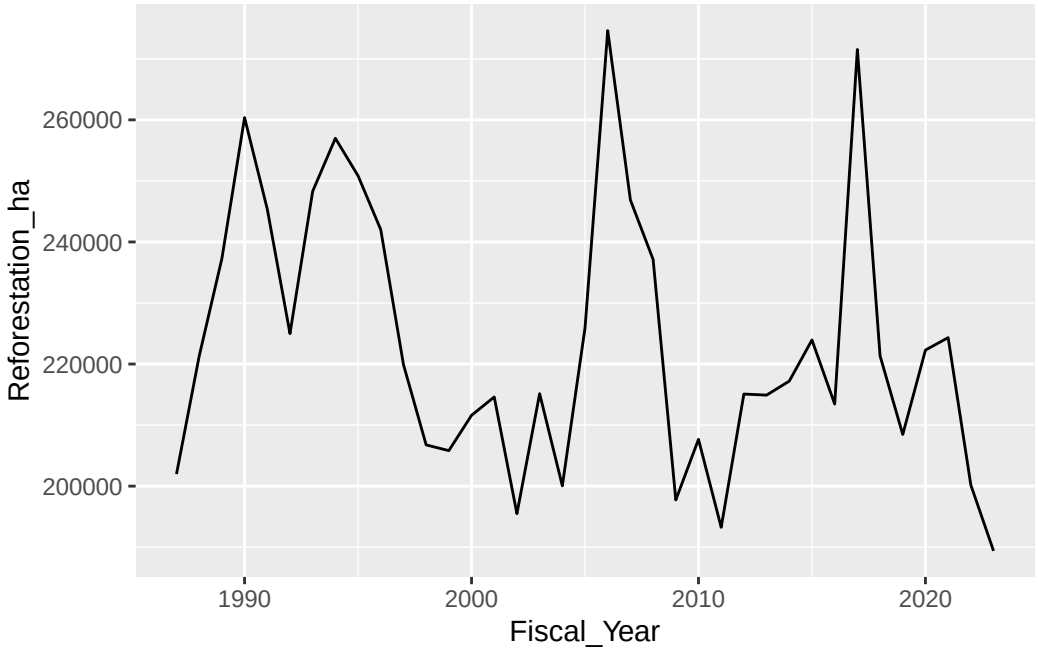

Figure 9.1: BC reforested area (ha) by fiscal year, 1987–2023.

-
-
-
-

9.4 `geom_point()` — scatter plot

```
ggplot(bc, aes(x = Harvested_ha, y = Natural_Disturbance_ha)) +
  geom_point()
```

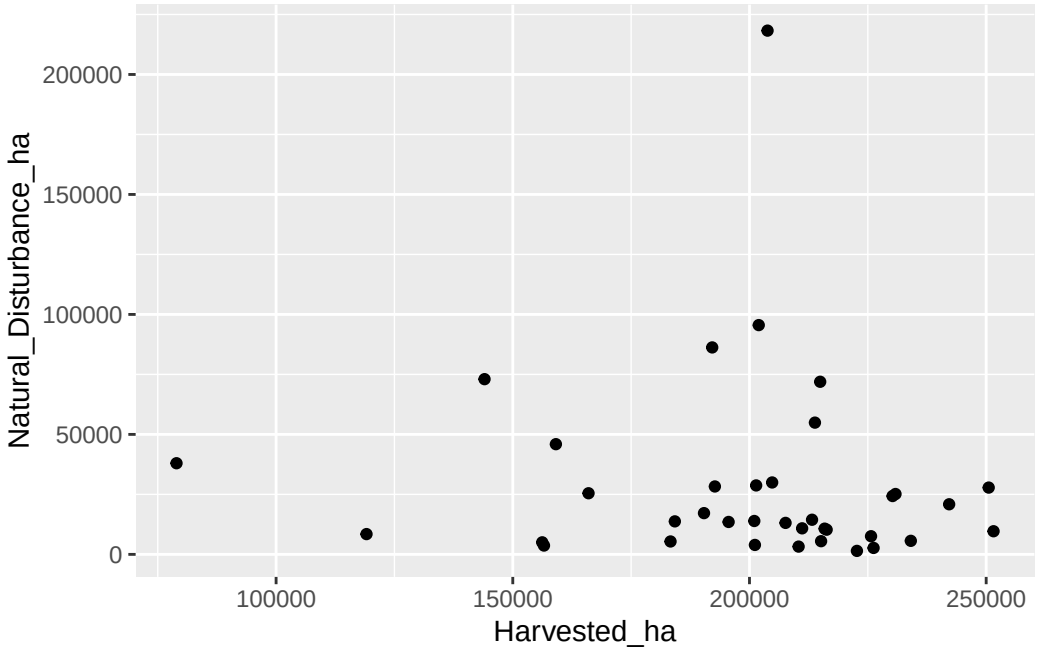

Figure 9.2: BC annual harvested area versus natural-disturbance area (ha), one point per year.

9.4.1 Mapping a third variable to colour

```
ggplot(bc, aes(x = Harvested_ha,
               y = Natural_Disturbance_ha,
               colour = Fiscal_Year)) +
  geom_point(size = 3)
```

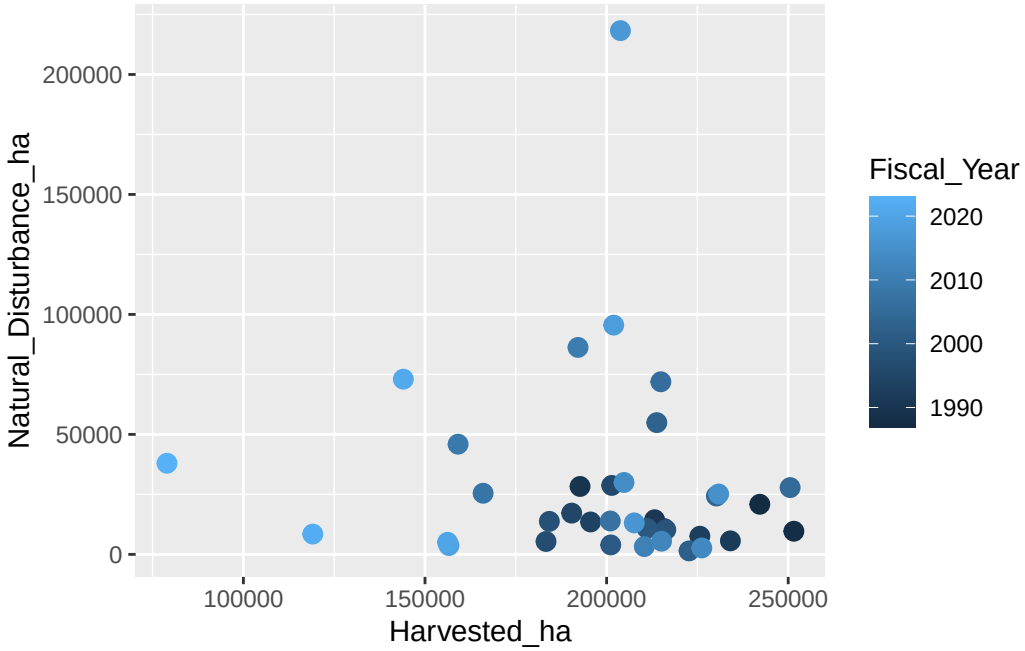

Figure 9.3: Harvested versus natural-disturbance area (ha), points coloured by fiscal year.

9.5 `geom_line()` — time series

```
# First reshape to long format so each series is a row
bc_long <- bc |>
  select(Fiscal_Year, Harvested_ha,
         Natural_Disturbance_ha, Reforestation_ha) |>
  pivot_longer(cols = -Fiscal_Year,
               names_to = "series",
               values_to = "area_ha")

ggplot(bc_long, aes(x = Fiscal_Year, y = area_ha, colour = series,
                    linetype = series)) +
  geom_line(linewidth = 1)
```

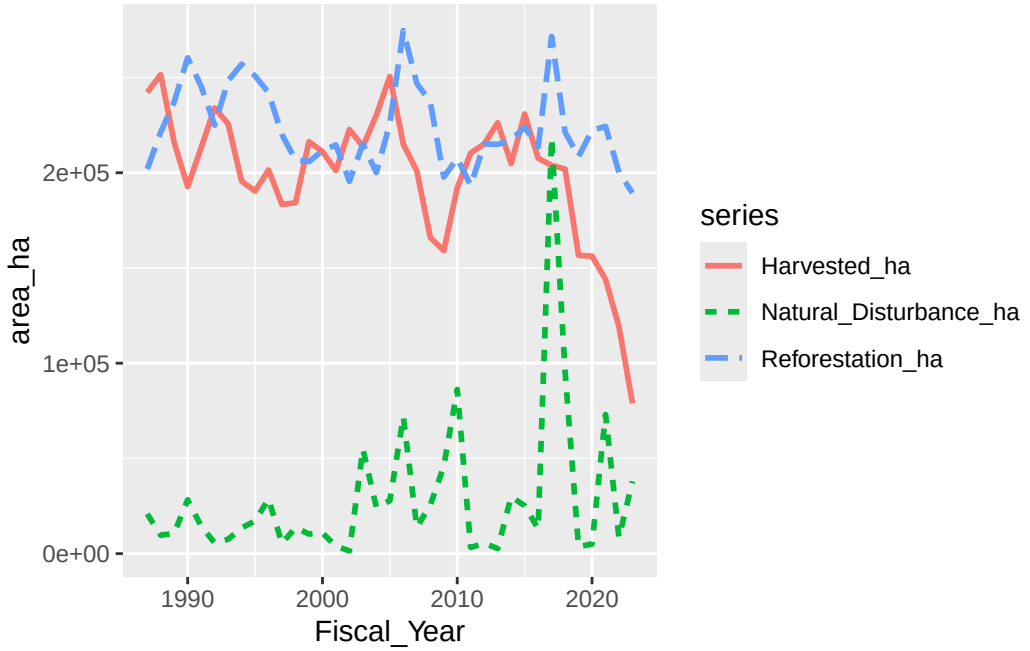

Figure 9.4: Harvested, natural-disturbance, and reforested area (ha) by fiscal year.

9.6 `geom_col()` vs `geom_bar()`

i AI prompt to check your work

Use this **after** your own attempt. It should check your reasoning, not hand you the answer:

> *"I want a chart that shows `[change over time / a distribution / a comparison across groups]` from `[columns]`, and I picked `[geom_*]`. Check whether that geom is the clearest choice for my message, suggest a better alternative if there is one, and list the labels and units the figure needs. Do not produce the final chart code for me."*

-
-

```r
# geom_col - you supply the y value
ggplot(bc, aes(x = Fiscal_Year, y = Reforestation_ha)) +
  geom_col(fill = "#2D6A4F")
```

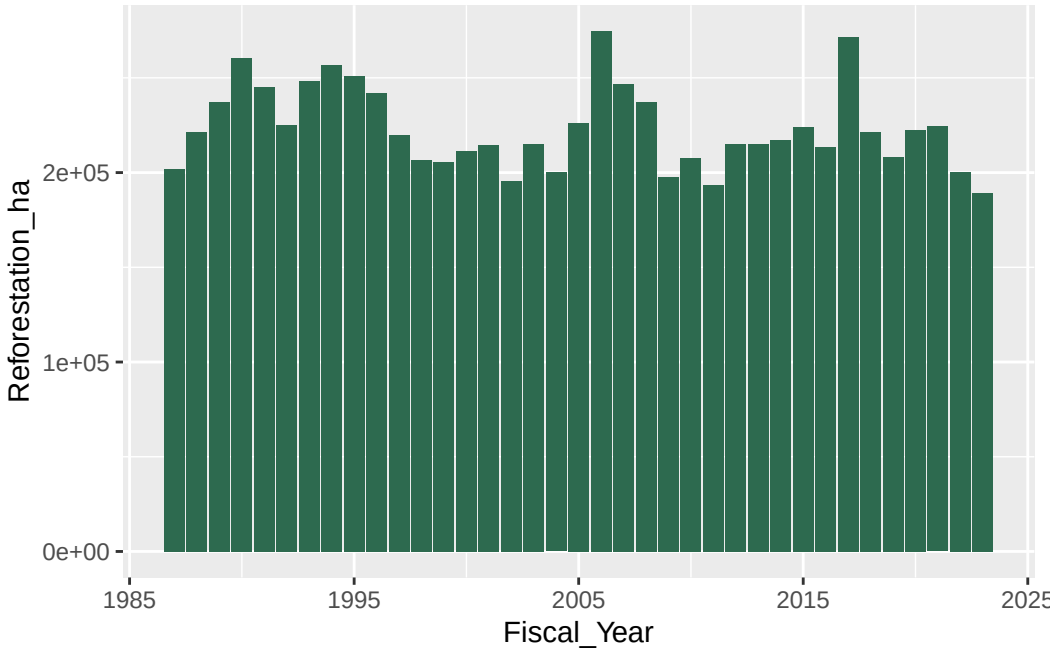

Figure 9.5: BC reforested area (ha) by fiscal year (geom_col).

```r
# geom_bar - it counts for you
# Build a small tibble with one row per disturbance event
events <- tibble(
  year = c(2017, 2017, 2017, 2018, 2019, 2019, 2020),
  type = c("fire", "fire", "pest", "fire", "fire", "wind", "fire")
)
ggplot(events, aes(x = type)) +
  geom_bar(fill = "#2D6A4F")
```

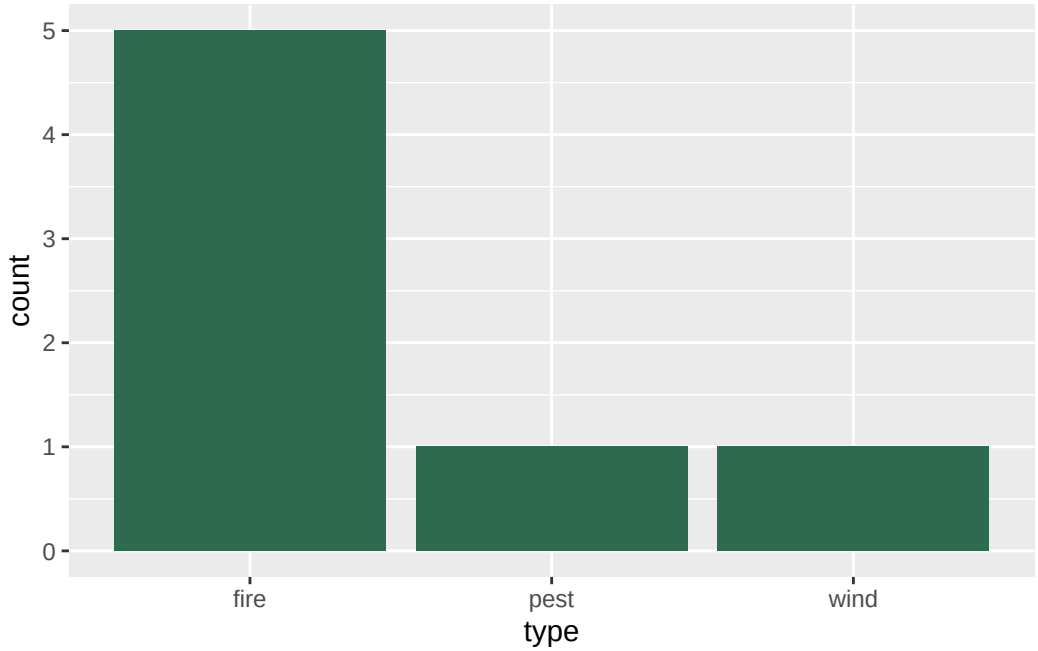

Figure 9.6: Count of disturbance events by type in a small sample (geom_bar).

> **A fresh import for this chapter**
>
> The `airdata_clean` object below is a **new import made here** for plotting — it is *not* the cleaned, region-joined tibble of the same name from Chapter 6. The `region` column is added later, in the box-plot example (via the station → region lookup). If you copy code that uses `region`, run that `left_join()` first, or you will see `object 'region' not found`.

9.7 `geom_histogram()` — distribution of one variable

```
airdata_clean <- read_csv(here("data", "airdata3.csv"),
                          show_col_types = FALSE) |>
  filter(!is.na(O3))
```

```
ggplot(airdata_clean, aes(x = O3)) +
  geom_histogram(binwidth = 2, fill = "#2D6A4F", colour = "white")
```

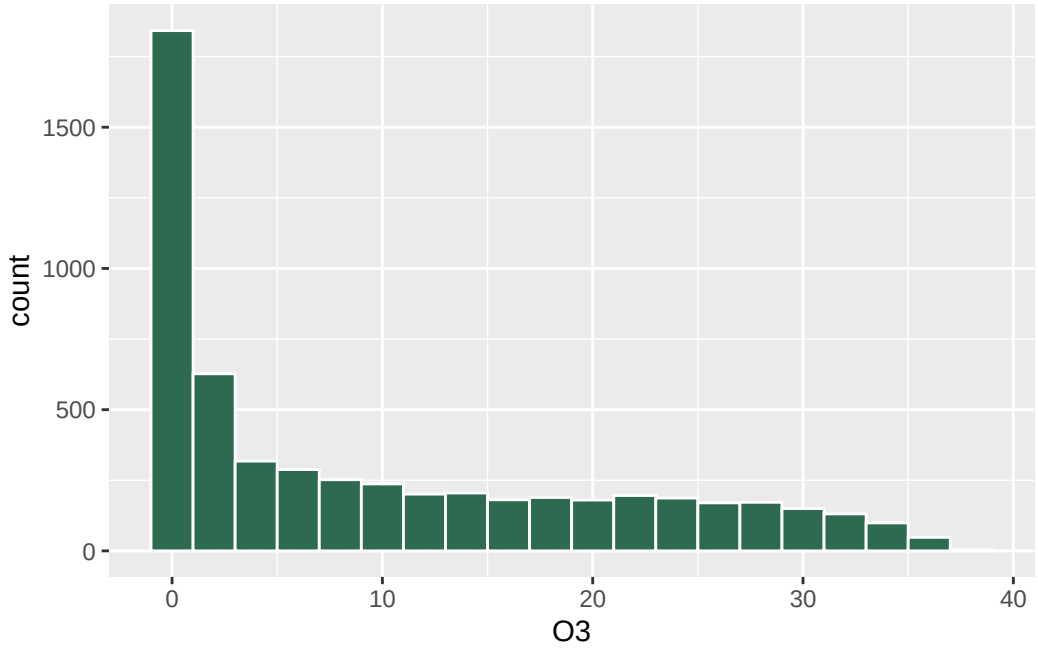

Figure 9.7: Distribution of hourly ozone (O_3, ppb).

3

9.8 `geom_boxplot()` — distributions by group

```
# First add a region column (from the Ch 6 lookup)
station_regions <- tribble(
  ~location,                          ~region,
  "Burnaby South",                    "Burrard Peninsula",
  "Coquitlam Douglas College",        "North-East Sector",
  "Langley Central",                  "Fraser Valley West",
  "Maple Ridge Golden Ears School",   "North-East Sector",
  "New Westminster Sapperton Park",   "Burrard Peninsula",
  "North Delta",                      "South of Fraser",
  "North Vancouver Second Narrows",   "North Shore",
  "Pitt Meadows Airport",             "North-East Sector",
  "Port Coquitlam North",             "North-East Sector",
  "Port Moody Rocky Point Park",      "North-East Sector",
  "Richmond South",                   "South of Fraser",
  "Tsawwassen",                       "South of Fraser",
  "Vancouver Clark Drive",            "Vancouver",
  "Vancouver International Airport #2","South of Fraser",
```

```
  "Vancouver Kitsilano",              "Vancouver",
  "West Vancouver Lions Gate",        "North Shore"
)

airdata_region <- airdata_clean |>
  left_join(station_regions, by = "location") |>
  filter(!is.na(region))

ggplot(airdata_region, aes(x = region, y = O3)) +
  geom_boxplot(fill = "#52B788")
```

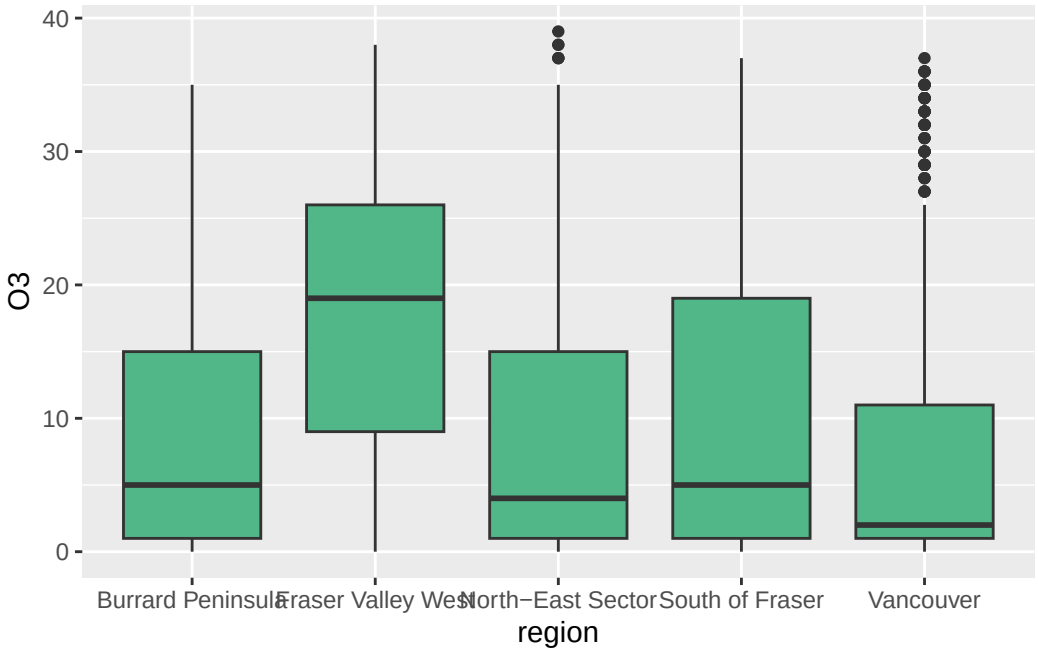

Figure 9.8: Hourly ozone (O_3, ppb) by Metro Vancouver region.

3

9.9 `geom_smooth()` — a fitted trend

```
ggplot(bc, aes(x = Fiscal_Year, y = Reforestation_ha)) +
  geom_point() +
  geom_smooth(method = "loess", se = TRUE)
```

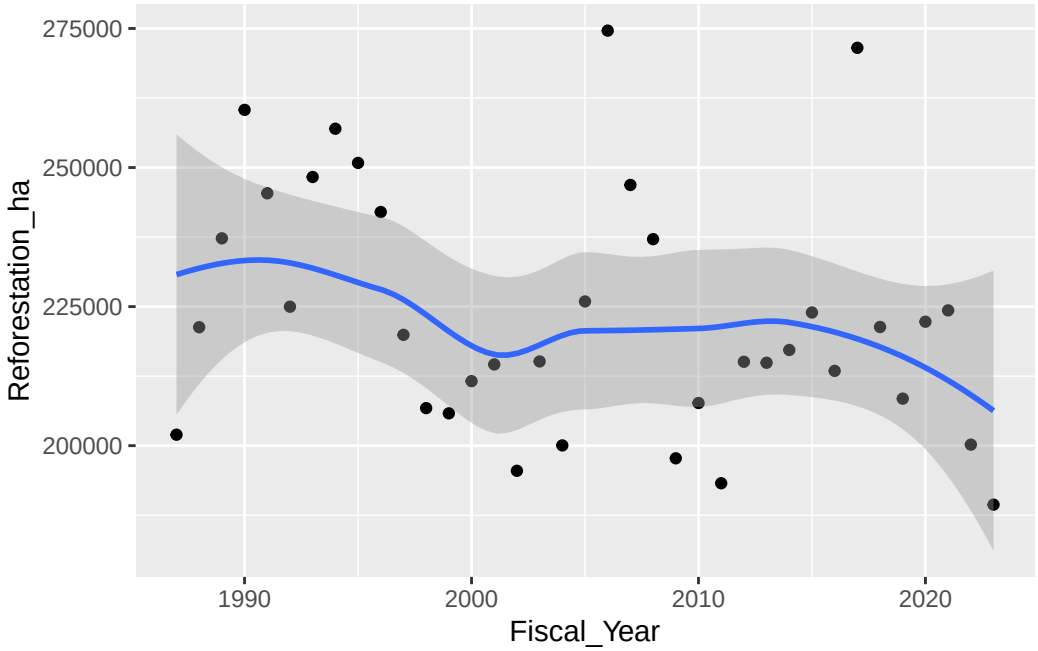

Figure 9.9: Reforested area (ha) by fiscal year with a loess trend line.

> **Layers draw in order**
>
> Every `+ geom_*()` adds a layer on top of the previous one. In the example above, `geom_point()` draws the dots first, then `geom_smooth()` draws the line on top. Reversing the order would hide the dots behind the ribbon.
> Read `+` as *"and also draw"*, in order.

9.10 Faceting — small multiples

```
ggplot(airdata_region, aes(x = O3)) +
  geom_histogram(binwidth = 2, fill = "#2D6A4F", colour = "white") +
  facet_wrap(~ region)
```

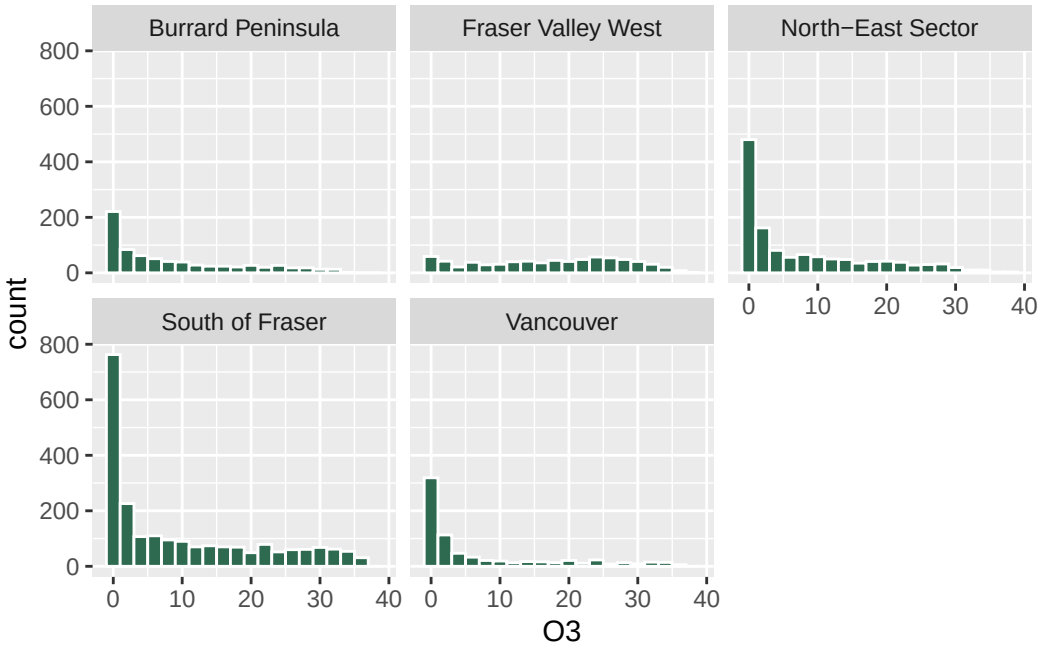

Figure 9.10: Hourly ozone (O_3, ppb) distribution, one panel per region.

9.11 Polish — labels, scales, theme

```
ggplot(bc, aes(x = Fiscal_Year, y = Reforestation_ha)) +
  geom_col(fill = "#2D6A4F") +
  scale_y_continuous(labels = label_comma()) +
  scale_x_continuous(breaks = seq(1990, 2025, by = 5)) +
  labs(
    title    = "BC Reforestation Area, 1987-2023",
    subtitle = "Annual area reforested (planted or naturally regenerated)",
    x        = "Fiscal year",
    y        = "Reforested area (hectares)",
    caption  = "Source: Environmental Reporting BC (Open Government Licence - BC)"
  ) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(colour = "grey40"))
```

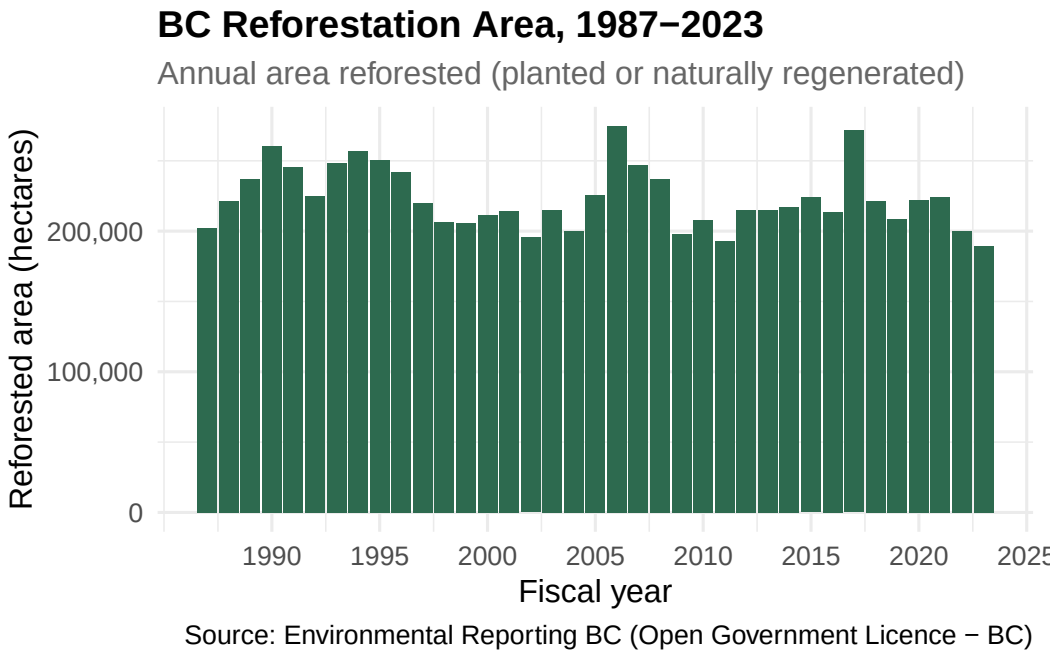

Figure 9.11: BC reforestation area (ha), 1987–2023 (polished chart).

-
-
-
-
-

9.12 Saving a plot

```
my_plot <- ggplot(bc, aes(x = Fiscal_Year, y = Reforestation_ha)) +
  geom_col(fill = "#2D6A4F") +
  labs(title = "BC Reforestation Area")

ggsave(here("outputs", "bc_reforestation.png"),
       plot   = my_plot,
       width  = 7,
       height = 5,
       dpi    = 300)
```

9.13 Seven geoms in one place

9.14 A worked composite — O_3 by station, faceted by region

```
airdata_region <- airdata_clean |>
  mutate(date_parsed = mdy(Date)) |>
  left_join(station_regions, by = "location") |>
  filter(!is.na(region), !is.na(date_parsed))

ggplot(airdata_region,
       aes(x = date_parsed, y = O3, colour = location)) +
  geom_line(linewidth = 0.7, alpha = 0.85) +
  facet_wrap(~ region, ncol = 1, scales = "free_y") +
  scale_x_date(date_breaks = "1 week", date_labels = "%b %d") +
  labs(
    title    = "Hourly O3, January 2000",
    subtitle = "Metro Vancouver air-quality monitoring network",
    x        = NULL,
    y        = "O3 (ppb)",
    colour   = "Station",
    caption  = "Source: Environmental Reporting BC"
  ) +
  theme_minimal(base_size = 10) +
  theme(legend.position = "right",
        axis.text = element_text(size = 7),
        plot.title = element_text(face = "bold"))
```

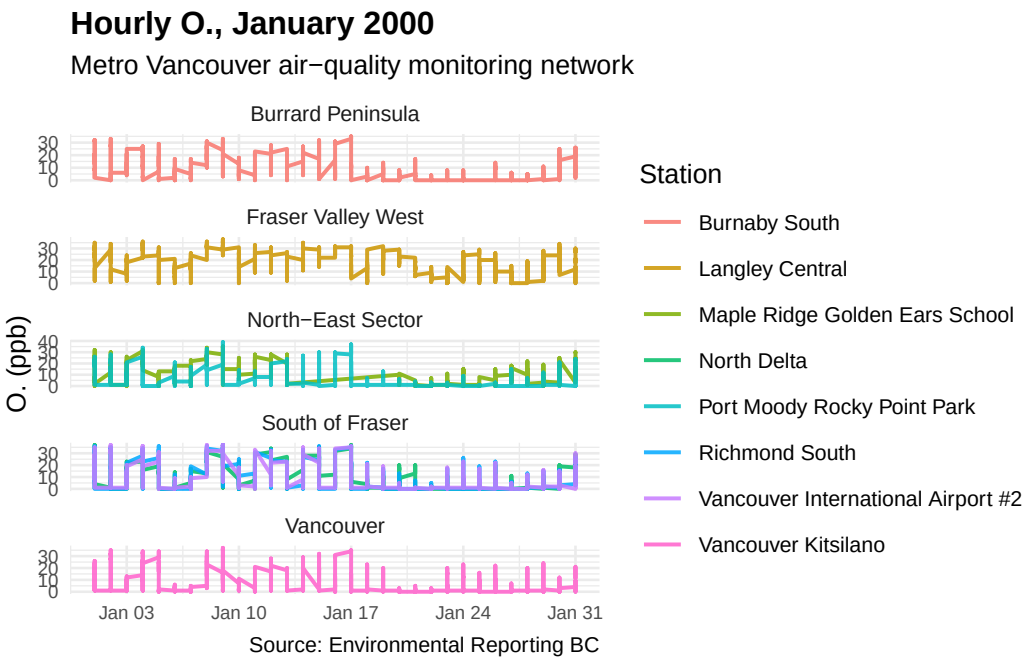

Figure 9.12: Hourly ozone (O_3, ppb) over January 2000, one panel per region.

9.15 Active learning

9.15.1 Activity 1 — Bar chart

9.15.2 Activity 2 — Three-line chart

9.15.3 Activity 3 — Histogram with sensible bins

3

9.15.4 Activity 4 — Boxplot by region

3

9.15.5 Activity 5 — Facet a time series

3

9.15.6 Activity 6 — Polish and save

9.16 Practice Demo Lab

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

1.
2.
3.
4.
5.
6.

Lab: Build a chart that tells a story

Work in groups of 3 or 4.
Task 1 — Pick a question. Each group picks one forestry question their chart will answer, e.g.:

- *Did BC harvest area decline after 2010?*
- *Which region had the most extreme O_3 hours in January 2000?*
- *How does the relationship between harvest and reforestation change over decades?*

Task 2 — Choose the right geom. Discuss: scatter, line, bar, histogram, or boxplot? Pick the one that makes the answer visible.
Task 3 — Build the plot. One member shares their screen and the group types the code together. Iterate on bin widths, scales, colours.
Task 4 — Polish. Add `labs()`, pick a theme, format the axes with `scale_*_continuous(labels = label_comma())` or `scale_x_date()`.
Task 5 — Save. Before saving, run this quick **plot checklist**:

- Is the **geom** right for the question (bar / line / point / box)?
- Are **x** and **y** the variables you intended?
- Do the **axis labels include units**?
- Are the **title / subtitle** informative, and is a **source caption** present?
- Will it save into **`outputs/`** (not your Desktop)?

Then save the chart to `outputs/` as PNG with `ggsave()`.
Task 6 — Caption it. Write a one-sentence caption that states the **headline finding** in plain language. A useful template:

> *"This chart shows [variable] by [group / time]. The main pattern is [finding], measured in [unit]."*

Task 7 — Submit. Each group submits the PNG, the code, and the caption.

Reference solution — download

Open the Chapter 9 reference solution (rendered HTML) — the completed, rendered solution for this demo lab. Open it **after** your own attempt and use it to check your work. *This is the **public practice** solution. The **graded lab on Canvas** has its own private answer key — do not copy this into a Canvas submission.*

9.17 Exercises

1.

2.

3. 3

4. 3

5. 3

6.

7.

8. 3

9.18 Optional, advanced

9.18.1 Plot themes

9.18.2 Colour palettes

-
-
-

9.18.3 `patchwork` for multi-panel layouts

```
library(patchwork)
p1 / p2          # stack p1 on top of p2
p1 | p2          # p1 and p2 side by side
(p1 | p2) / p3   # two-column top row, single-row bottom
```

9.18.4 Interactive plots

9.19 A glimpse ahead: the Integrated Case Study

-
-
-
-

[3]

Optional: Spatial Data Analysis in R

9.20 Self-assessment quiz

9.21 AI as a debugging companion

Useful prompts

- *"My ggplot only draws one line, but my long-format tibble has three series. What did I miss in aes()?"* (Almost always: forgot to add `group = series` or `colour = series`.)
- *"How do I make the x-axis show dates like 'Jan 15' instead of '2000-01-15' in ggplot?"* (Answer: `scale_x_date(date_labels = "%b %d")`.)
- *"My chart's y-axis says 8.26e+06 instead of 8,264,063. How do I fix it?"* (Answer: `scale_y_continuous(labels = label_comma())`.)

9.22 Reading

- https://r4ds.hadley.nz/
- https://ggplot2-book.org/
- https://rstudio.github.io/cheatsheets/data-visualization.pdf

Instructor notes

Pacing and emphasis

- Spend most time on **`geom_col`, `geom_line`, `geom_histogram`, `geom_boxplot`, and `facet_wrap`** — these five cover 90% of routine forestry plotting.
- The **long-format-for-plotting** habit is the chapter's most important reusable insight. When a student's chart looks wrong, ask first: "is your data long or wide?"
- **`ggsave` matters.** Students will hand in screenshots instead of PNG files if you do not insist on `ggsave`.

9.22.1 Materials provided

File	Purpose
`frst232_ch09_learner.qmd`	Quarto starter with TODO chunks.
`frst232_ch09_solutions.qmd`	Completed solutions. **Do not distribute.**
`quiz_ch09.html`	Standalone interactive quiz.

9.22.2 Common stumbling points

- **`+` instead of `|>` in pipelines.** Walk through: `|>` for data transformation, `+` for ggplot layers.
- **`colour` outside `aes()`** instead of inside (or vice versa). Inside aes = data-driven; outside aes = constant.
- **Wide data plotted with `geom_line`** giving one line per observation instead of one per series. Fix: `pivot_longer` first.
- **Forgetting `group = ...`** in line charts with categorical

    x. Symptom: zig-zag lines.

Part V

Case Study & Communication

10 Integrated Case Study: Choosing the Right Tools

> **Data for this chapter**
>
> This chapter uses the **BC silviculture** workbook (`bc_disturbance_reforestation.xlsx`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

-
-
-
-
-
-
-

Expected output for this chapter

What you will hand in

A short **case-study report** (rendered from a `.qmd`) that:

1. states a **question** in plain language;
2. shows your **chosen workflow** with a one-line justification for each step ("I used `group_by()` because …");
3. includes **one clear figure**;
4. shows your **plausibility checks** (row counts, missing-value checks, summary statistics);
5. ends with a **two- to three-sentence interpretation** a non-technical forestry reader could follow.

10.1 Why this chapter is different

10.2 The decision framework — think before you code

1.
2.
3.

4.

5.

6.

10.3 A worked example — the process end to end

-
-
-
-

-
-

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")

by_decade <- bc |>
  mutate(decade = floor(Fiscal_Year / 10) * 10) |>
  group_by(decade) |>
  summarise(
    n_years            = n(),
    mean_reforested_ha = round(mean(Reforestation_ha, na.rm = TRUE)),
    mean_harvested_ha  = round(mean(Harvested_ha,     na.rm = TRUE)),
    .groups = "drop"
  ) |>
  arrange(desc(mean_reforested_ha))

by_decade |> kable()
```

```
by_decade |>
  ggplot(aes(x = factor(decade), y = mean_reforested_ha)) +
  geom_col(fill = "#2D6A4F") +
  scale_y_continuous(labels = scales::label_comma()) +
  labs(x = "Decade", y = "Mean reforested area (ha/yr)",
       title = "Average annual reforestation by decade, BC",
       caption = "Source: Environmental Reporting BC") +
  theme_minimal()
```

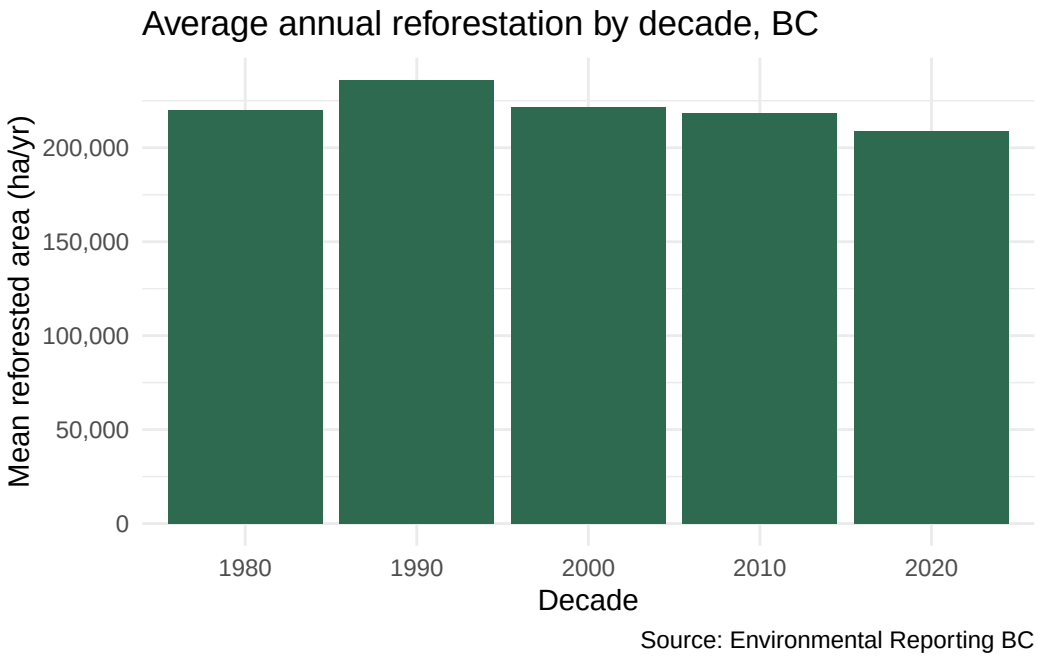

Figure 10.1: Mean annual reforested area (ha) by decade, British Columbia.

10.4 Choose the function — activities

1.
2.
3.
4.
5.

-

-

10.5 Integrated case-study questions — decide the tools

-
-
-
-
-
-

10.6 Verify before you trust it

10.7 AI as a verification partner — not the analyst

-
-
-

10.8 Self-check quiz

1.

2.

3.

10.9 Done

11 Group Presentations and Portfolio Outputs

Data for this chapter

This chapter's fresh dataset is the **BC silviculture** workbook (`bc_disturbance_reforestation.xlsx`) — click to download it into your `data/` folder (source and details on the Datasets Used in This Book page). But your **main inputs here are your own outputs from earlier chapters** — the figures, cleaned tables, and rendered reports (and optional maps, only if your group used the spatial extension) that you will assemble into a presentation and a portfolio.

Chapter goals

-
-
-
-
-
-
-
-

Expected output for this chapter

What you will hand in

1. A **rendered group presentation** in revealjs HTML format, covering one analysis from the term (10–15 slides for a ~12-minute talk).
2. A **personal portfolio** — at least three of your previous chapter deliverables (rendered HTML reports) collected into a single linked landing page.
3. A **citation file** (`CITATION.md`) listing the data sources, R packages, and any external code used.

4. A **licence statement** (CC BY 4.0 or equivalent) at the bottom of your portfolio landing page.
5. The corresponding **`.qmd` source files** for both the presentation and the portfolio.

11.1 Where we are

1.
2.

11.2 Quarto outputs — three formats from one file

11.3 PowerPoint-to-Quarto bridge

11.4 Your first revealjs file

```
---
title: "BC Reforestation Trends, 1987-2023"
author: "Your name"
date: today
format: revealjs
---
```

11.4.1 Slide separators — two patterns

```
## Slide 1 title

content

## Slide 2 title

more content
```

```
A slide with no title.

---

Another slide with no title.
```

Heading hierarchy in revealjs

- **`# Section Title`** (h1) — creates a *section divider* slide.
- **`## Slide title`** (h2) — creates a *content slide*.
- **`### Subheading`** (h3) — used within a slide.

A talk with three sections, each containing three slides, has `#` × 3 and `##` × 9.

11.5 The one-idea-per-slide rule

-
-
-
-

11.6 Embedding R output in slides

```
## BC reforestation trend


::: {.cell}
::: {.cell-output-display}
![](12-presentations-portfolio_files/figure-pdf/unnamed-chunk-1-1.pdf){fig-alt='Bar chart of
:::
:::
```

11.7 Speaker notes

```
## BC reforestation trend

[chart]

::: {.notes}
Talking points: emphasise the post-2010 increase, mention the
2018 dip caused by silviculture-budget cuts, prompt audience
to predict 2024.
:::
```

11.8 Incremental reveals

```
## Three findings

::: {.incremental}
- Reforestation has risen ~50% since 2010.
- Natural disturbance peaked in 2017 wildfires.
- The ratio reforestation : disturbance crossed 1 in 2019.
:::
```

11.9 Two-column layout

```
## Static vs interactive charts

::: {.columns}
::: {.column width="50%"}
**Static** - `ggplot2`

- One PNG
- Fixed view
- Goes in printed reports
:::

::: {.column width="50%"}
**Interactive** - `plotly`

- HTML widget
- Hover, zoom, pan
- Goes in HTML deliverables
:::
:::
```

11.10 A complete revealjs `.qmd` skeleton

```
---
title: "BC Air Quality, January 2000"
subtitle: "An FRST 232 group presentation"
author: "Members A, B, C, D"
date: 2027-04-10
format:
  revealjs:
    theme: simple
    incremental: false
    transition: slide
    slide-number: true
    chalkboard: false
    embed-resources: true
```

```
    code-overflow: wrap
---

# Background

## The question

What did January 2000 PM2.5 look like across Metro Vancouver?

::: {.notes}
Set up the question - the file we cleaned in Ch 6 is hourly PM2.5
across 16 stations.
:::

## The data

::: {.incremental}
- 11,904 rows × 27 columns raw
- 16 stations across Metro Vancouver
- Source: Environmental Reporting BC
- Licence: Open Government Licence - BC
:::

# Cleaning

## The pipeline

[paste the Ch 6 dplyr pipeline as a code chunk with `echo: true`]

# Findings

## Per-station means

[paste the bar chart of mean PM2.5 by station]

## Regional differences

[paste the box plot from Ch 9]

## On a map (optional)

[If your group used the optional spatial extension, paste your leaflet map here.]

# Conclusion

## What we found
```

```
One sentence. Big and centered.

## Thank you

Questions?
```

11.11 Building a portfolio

11.11.1 Simple folder structure

11.11.2 Minimal `_quarto.yml`

```yaml
project:
  type: website
  output-dir: docs

website:
  title: "FRST 232 - Forestry Data Analysis"
  navbar:
    left:
      - href: index.qmd
        text: Home
      - href: chapters/ch05.qmd
        text: Ch 5 - Inspection
      - href: chapters/ch06.qmd
        text: Ch 6 - Cleaning
      - href: chapters/ch07.qmd
        text: Ch 7 - Summarising
      - href: chapters/ch09.qmd
        text: Ch 9 - Charts
```

```
      - href: chapters/ch10.qmd
        text: Ch 10 - Case study

format:
  html:
    theme: cosmo
    toc: true
```

11.11.3 A minimal `index.qmd`

```
---
title: "FRST 232 Portfolio"
---

# About this site

A collection of forestry data analyses produced for FRST 232
(UBC, Fall 2026). Each linked report covers a different
methodological topic, using openly licensed BC and Ontario
forestry data.

## Highlights

- [Chapter 6 - Cleaning the BC air-quality file](chapters/ch06.html)
- [Chapter 9 - Visualising reforestation trends](chapters/ch09.html)
- [Integrated Case Study - choosing the right tools](chapters/10-case-study.html)

## Author

Your name · [LinkedIn / email]

## Licence

All content © 2026 Your Name, released under
[CC BY 4.0](https://creativecommons.org/licenses/by/4.0/).
Code is released under the MIT licence.

## Data sources

- BC silviculture: Environmental Reporting BC
- BC air quality: Environmental Reporting BC
- Ontario forest stats: Government of Ontario
```

11.12 Citation — citing data, code, and packages

1.
2.
3.

11.12.1 A `CITATION.md` in your project root

```
# Citations

## Data

- BC Government, Environmental Reporting BC. *Trends in
  Silviculture in British Columbia*. Open Government Licence -
  BC. <https://catalogue.data.gov.bc.ca/dataset/indicator-summary-data-trends-in-silvicultur
  Accessed 2026-09-15.

- Government of Ontario. *Forest Resources of Ontario 2021*.
  Open Government Licence - Ontario.
  Accessed 2026-09-22.

## R packages

- Wickham H et al. (2019). Welcome to the tidyverse. *Journal of
  Open Source Software*, 4(43), 1686.
- Pebesma E (2018). Simple Features for R: Standardized Support
  for Spatial Vector Data. *The R Journal* 10(1), 439-446.
- Cheng J, Karambelkar B, Xie Y (2024). leaflet: Create
  Interactive Web Maps with the JavaScript "Leaflet" Library.
```

11.13 Licensing — choose one openly

11.14 Publishing — three free hosts

11.15 Active learning

11.15.1 Activity 1 — Convert a chapter to slides

11.15.2 Activity 2 — One idea per slide

11.15.3 Activity 3 — Two-column layout

11.15.4 Activity 4 — Citation

11.15.5 Activity 5 — Apply a licence

11.15.6 Activity 6 — Publish

11.16 Practice Demo Lab — In-class presentations

> **!** Practice demo only
>
> This is a **guided practice lab in the OER book**, designed to help you learn the workflow before completing the official lab assignment on **Canvas**. The Canvas lab is the graded assignment. Use this activity to practice, compare your work with the reference solution, and learn how to write an AI verification prompt.

1.
2.
3.
4.
5.

6.

Lab: The synthesis presentations

The lab session for this chapter is given over to **in-class group presentations**. Each group presents one analysis from the term, using a revealjs deck they have built in advance.

Format

- 4 groups × 12 minutes each (10 min talk + 2 min Q&A).
- Each group covers one chapter's headline analysis (e.g., Ch 6 cleaning, Ch 7 summarising, Ch 8 joining, or Ch 9 charts).
- Each group member speaks for at least 2 minutes.

Required slide structure

1. **Title slide** — title, group members, date.
2. **The question** — what forestry question are we answering?
3. **The data** — source, licence, row/column counts.
4. **The method** — show the actual R code (10 lines).
5. **The result** — one chart, table, or map.
6. **Interpretation** — one sentence headline + one paragraph nuance.
7. **Limitations** — what does this analysis NOT show?
8. **Acknowledgements** — citations of data sources and packages.

Peer evaluation

Every learner submits a peer-eval form for every other group, scoring:

- Clarity of the question (0–3)
- Quality of the slides (0–3)
- Depth of interpretation (0–3)
- Acknowledgement of limitations (0–3)

Submission

- The rendered revealjs HTML.
- The source `.qmd`.
- The peer-eval forms.

11.17 Exercises

1.

2.

3.

4.

5.

6.

7.

8.

11.18 Optional, advanced

11.18.1 Slide themes and templates

```
format:
  revealjs:
    theme: [default, dark, sky, beige, simple, serif, blood, night, moon, solarized]
```

11.18.2 Custom CSS

```
format:
  revealjs:
```

```
    theme: simple
    css: my-custom.css
```

```
.reveal h1, .reveal h2, .reveal h3 {
  color: #1B4332;
}
.reveal blockquote {
  border-left: 4px solid #2D6A4F;
  padding-left: 1em;
}
```

11.18.3 Print to PDF during the talk

11.18.4 Multi-author Quarto projects

11.19 A glimpse ahead: Wrap-up and final exam preparation

-
-
-
-

11.20 Self-assessment quiz

11.21 AI as a debugging companion

Useful prompts

- *"My revealjs slide has too much text. Help me split it into three shorter slides."* (Paste the slide content.)
- *"My ggplot chart looks fine in HTML but the labels are tiny on the projector. What do I change?"* (Answer: increase `base_size` in `theme_*(base_size = 18)`.)
- *"Write a CITATION.md entry for the readxl R package."*

11.22 Reading

- https://quarto.org/docs/presentations/revealjs/
- https://quarto.org/docs/websites/
-
- https://creativecommons.org/choose/
- https://www2.gov.bc.ca/gov/content/data/policy-standards/open-data/open-government-licence-bc

Instructor notes

Lab logistics

- The lab session for this chapter is in-class group presentations.
- 4 groups × ~12 minutes each fits in a 60–75 minute lab.
- Bring the projector hook-up that matches the venue; have a PDF backup of each group's slides in case revealjs fails on the projector machine.
- Encourage groups to use **presenter mode (S)** during the talk — speaker notes are vastly more useful than memorising bullet points.

11.22.1 Grading rubric

Criterion	Weight	What to look for
Title slide + agenda	10 %	All members named; clear topic
The question	10 %	Stated as one sentence, not a paragraph
The data	10 %	Source, licence, row counts
The method	15 %	Real R code, not pseudocode
The result	25 %	One clear chart, table, or map
Interpretation	15 %	Headline + nuance

Limitations	10 %	What the analysis does NOT show
Acknowledgements	5 %	Data citations + R-package citations

11.22.2 Materials provided

File	Purpose
`frst232_ch12_learner.qmd`	Quarto starter — exercises walking through revealjs and portfolio assembly.
`frst232_ch12_solutions.qmd`	Completed solutions. **Do not distribute.**
`frst232_ch12_slides_template.qmd`	A ready-to-customize revealjs presentation skeleton.
`quiz_ch12.html`	Standalone interactive quiz.

11.22.3 Common stumbling points

- **Chart fonts too small on projector.** Insist on `theme_*(base_size = 18)` in slide chunks.
- **No limitations slide.** Push back hard — limitations are the single best signal of professional maturity in an analytical presentation.
- **Wall-of-text slides.** Apply the 50-word rule live; refactor in front of the class.
- **Missing data citations.** Every dataset used must be credited on the acknowledgement slide.
- **`format: revealjs` not rendering.** Some Posit Cloud setups need `install.packages("rmarkdown")` even though Quarto is bundled. The error message is descriptive.

12 Wrap-up and Final-Exam Preparation

Data for this chapter

The end-to-end case study uses the **BC silviculture** workbook (`bc_disturbance_reforestation.xlsx`). Click the file name to download it directly, then save it in your `data/` folder. For the source and details, see the Datasets Used in This Book page.

Chapter goals

-
-
-
-

Expected output for this chapter

What you will produce

A complete **end-to-end case-study report**: take one of the course datasets, start from the raw file, document your cleaning, summarize by group, produce **one figure** (and an **optional map** only if you used the spatial extension), and render it as a single Quarto HTML report. This is both your final review and the best possible exam practice.

12.1 Where we are

12.2 End-to-end case study: BC silviculture

12.2.1 1. Import

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"),
                 sheet = "Data")
dim(bc)
```

12.2.2 2. Inspect

```
glimpse(bc)
```

```
colSums(is.na(bc))   # missing values per column
```

12.2.3 3. Clean and derive

```
bc_clean <- bc |>
  mutate(
    decade          = (Fiscal_Year %/% 10) * 10,
    reforest_ratio  = Reforestation_ha / Harvested_ha
  )
bc_clean |> select(Fiscal_Year, decade, Harvested_ha,
                   Reforestation_ha, reforest_ratio) |> head()
```

12.2.4 4. Summarize by group

```
by_decade <- bc_clean |>
  group_by(decade) |>
  summarise(
    n_years           = n(),
    mean_harvest_ha   = round(mean(Harvested_ha, na.rm = TRUE)),
    mean_reforest_ha  = round(mean(Reforestation_ha, na.rm = TRUE)),
    mean_ratio        = round(mean(reforest_ratio, na.rm = TRUE), 2),
    .groups = "drop"
  )
by_decade |> kable()
```

12.2.5 5. Visualize — one figure

```
bc_long <- bc_clean |>
  select(Fiscal_Year, Harvested_ha, Reforestation_ha) |>
  pivot_longer(-Fiscal_Year, names_to = "series", values_to = "area_ha")

ggplot(bc_long, aes(Fiscal_Year, area_ha, colour = series, linetype = series)) +
  geom_line(linewidth = 1) +
  scale_colour_manual(values = c("#B7791F", "#2D6A4F"),
                      labels = c("Harvested", "Reforested")) +
  scale_linetype_manual(values = c("solid", "dashed"),
                        labels = c("Harvested", "Reforested")) +
  labs(title = "BC harvest and reforestation, 1987-2023",
       x = "Fiscal year", y = "Area (hectares)",
       colour = NULL, linetype = NULL,
       caption = "Source: Environmental Reporting BC") +
  theme_minimal()
```

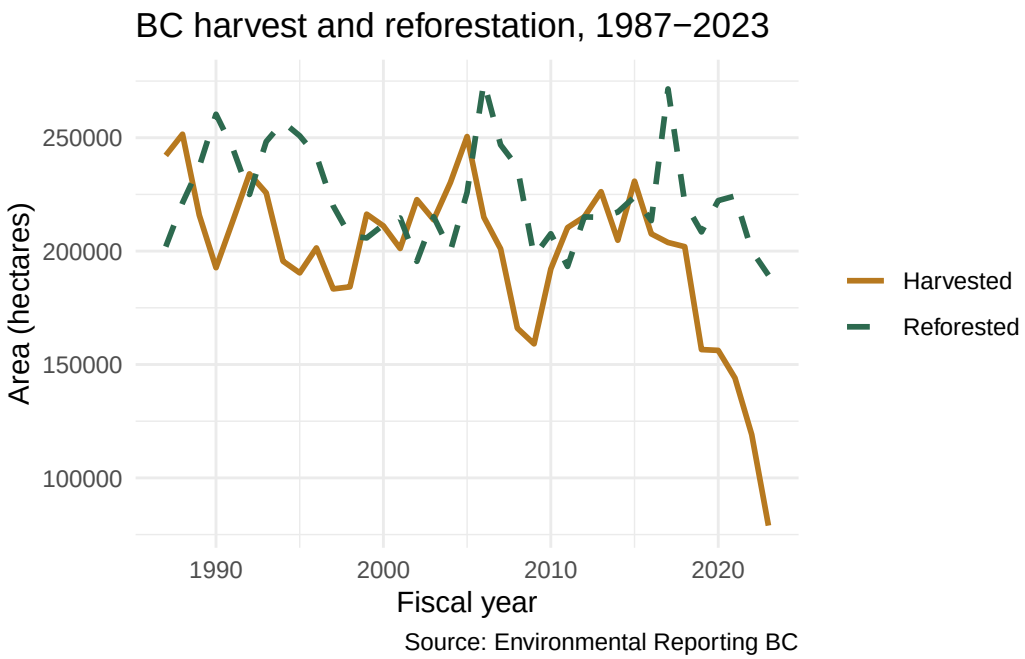

Figure 12.1: BC harvested and reforested area (ha) by fiscal year, 1987–2023.

12.2.6 6. Render and explain

12.3 Final-exam preparation: the debugging ladder

1.
2.
3.
4.
5.
6.

Using AI during study (not during a closed exam)

When studying, AI is excellent for *explaining* an error or *generating practice variants*. Paste the **exact** error, your R and package versions, and the code you tried. Then verify the explanation against `?function_name` and a small test — the same discipline the rest of the book taught. In a closed-book exam, the ladder above is what you carry in your head.

12.4 Active learning — exam practice

12.4.1 Activity 1 — Full pipeline, new dataset

12.4.2 Activity 2 — Break and fix

12.4.3 Activity 3 — Reproduce from clean

12.5 Where to go next

-
-
-
-

- https://r4ds.hadley.nz
- https://r.geocompx.org
-
- https://www.bcfnfc.ca/

AI as a debugging companion

Instructor notes

Pacing, exam logistics, and accommodations

- **Pacing.** One session of review, one of Q&A / sample-exam walkthrough, one wrap-up. No graded lab this week.
- **Final exam** covers import, clean, summarize, reshape, join, and visualize with R, tidyverse, and ggplot2 — weighted 30% per the syllabus. (Spatial mapping with `sf` is an optional extension and is not required on the exam.)
- **Accommodations.** Confirm Centre for Accessibility arrangements *before* the exam (extended time, alternative formats, rest breaks). Where the timed format raises equity concerns, consider a take-home practical for the whole cohort, not just students with formal accommodations — see the *Accessibility statement* in the front matter.
- **Common stumbling points.** Forgotten `library()` calls; data paths that work locally but not from a clean project; reading `NA` as zero.

Working with AI on this chapter

AI prompt to check your work

Fill in the brackets and paste this into your AI assistant. It should *explain and check*, not hand you a final assignment answer:

> *"I am working on FRST 232, this chapter. My dataset is `[file name]` with columns `[columns]`. My task is `[what you are trying to do]`. I tried `[paste your code or steps]` and got `[paste the exact output or error]`. Explain what this does line by line, tell me whether it answers the question, and list the checks I should run to verify it. Do not give me the final answer."*

-
-
-
-

Compare your solution with the reference

When a reference solution is available, do **not** just copy it — use it to check your own work:

- **Expected result** — does your output match the target shape and values?
- **Required components** — did you include every step the task asked for?
- **If your result differs** — say *where* it matched, *where* it differed, *what* caused the difference, *how* you fixed it, and *what* you learned.

AI replication challenge

Write a prompt that would guide an AI *toward* the reference solution without handing over the final answer. Include the dataset, the columns, the task goal, the output format you need, and the verification checks. Then paste (or summarise) the AI's response and answer: did its approach match the book's? did it miss a verification step? did it invent a column, function, or value? what did your group change after checking it?

Optional: Spatial Data Analysis in R

Spatial data analysis in R — additional support

Continue to the optional spatial-data resource:

Spatial Data Analysis in R — additional support

Direct link (works in print too): **https://subornaa.github.io/spatial-analysis-r/**

References

Key Terms and Coding Vocabulary

> **Tip**
>
> Use the table of contents (or your browser's find, `Ctrl`/`Cmd` + `F`) to jump to a term. Every term heading is a link target.

Excel and data basics

Data dictionary

Cleaning log

Cell reference (relative vs absolute)

Structured table

PivotTable

Missing value

3

Open Government Licence (OGL)

R basics

Object

```
bc <- read_excel(here("data", "bc_disturbance_reforestation.xlsx"))
```

Vector

```
ha <- c(220000, 180000, 250000)   # a numeric vector
```

Function

Argument

Package

```
install.packages("readxl")   # once
library(readxl)              # every session
```

Working directory

```
here("data", "airdata3.csv")   # a portable path
```

Data frame

Tibble

Tibble

Pipe

```
bc |> filter(Fiscal_Year >= 2000) |> summarise(mean(Reforestation_ha))
```

Tidyverse vocabulary

Filter

```
airdata |> filter(!is.na(O3))     # keep rows where O3 was measured
```

Select

```
bc |> select(Fiscal_Year, Reforestation_ha)
```

Mutate

```
bc |> mutate(decade = floor(Fiscal_Year / 10) * 10)
```

Summarize

```
airdata |> group_by(location) |> summarise(mean_o3 = mean(O3, na.rm = TRUE))
```

group_by

Join

```
airdata |> left_join(station_regions, by = "location")
```

Key

Pivot

Quarto and reproducible reporting

Quarto

Chunk

Render

Reproducible workflow

Visualization

Visualization

Geom

Aesthetic mapping

```
ggplot(bc, aes(x = Fiscal_Year, y = Reforestation_ha))
```

Facet

AI-assisted debugging

Prompt

Hallucination

Verification

Spatial terms (optional extension)

CRS (coordinate reference system)