2 Review

Appendix A — File Organization & Paths

1. Always Use an R project
    - start every assignment in .Rproj folder
    - sets your project's home base so all file paths work automatically
2. Use Relative Path Files
    - relative path = location within your project folder
    - absolute path = your computer's file system
3. Organize Files Into Standard Folders
    - clean organization, reproducible, works on any computer, easy for grading
4. 4. Keep Raw Data Raw
    - never edit or overwrite the original dataset
    - if you need cleaned data, write a cleaning script in R
5. Scripts Should Be Modular
    - your analysis script should not contain everything
    - put repeated or long chunks of code into separate R files and use source("R/clean-data.R")
6. All Code Should Run From the Project Root
    - anything created/saved should use project-relative paths
7. Quarto Files Must Knit on Another Computer
    - the grader should be able to click Render and your document runs perfectly
8. Good Practices
    - no spaces or weird symbols in filenames
    - use meaningful names for files
    - keep your project clean, delete clutter

Appendix B — Git and GitHub

1. What Git Is
    - Git = a version control system
    - it helps you track changes, experiment safely, and go back to previous versions of your work
    - no more losing code
    - easy to collaborate
    - easy to show your progress and professionalism
2. What GitHub Is
    - GitHub = an online platform where your Git repositories live.
    - think of it as Google Drive for code, but with version history, collaboration tools, automatic backups, public or private sharing
3. Repositories

- a repo is a project folder managed by Git
- a repo contains your code, your data, your history, your branches
- each assignment/project should have its own repo on GitHub

4. The 3-Step Workflow You Use Constantly
    - stage, commit, push
5. Cloning vs. Pulling
    - clone = download a GitHub repo to your computer once
    - pull = get new changes from GitHub before you start working
    - prevents conflict
6. Why Commit Messages Matter
    - help you remember what you did
    - show progress to instructors
    - make debugging easier
7. Never Use GitHub Like Google Drive
    - don't manually upload files in the browser
8. Only Sync the Files That Should Be Version Controlled
    - do not put huge datasets, images, or temporary files in Git
9. Keeping Your Repo Clean
    - use .gitignore to hide unwanted files
    - don't track outputs you can regenerate
    - always pull before pushing
    - commit frequently (every logical change)

Appendix C — Keyboard Shortcuts

1. RStudio Shortcuts Save Time
    - using keyboard shortcuts makes coding faster, cleaner, and more efficient
2. Most Important Shortcuts
    - run code
    - comment/uncomment code
    - insert a pipe
3. Navigating in RStudio
    - move between tabs
    - search within a file
    - search across the entire project
    - indent/align code
4. R Markdown/Quarto Shortcuts
    - insert code chunk
5. Console Tricks
    - clear console
    - interrupt a long/buggy process

    - recall previous commands
6. General Coding Efficiency Tips
    - keep your hands on the keyboard
    - use shortcuts instead of menus
    - keep code chunks small and readable
    - comment frequently
    - use the pipe shortcut to avoid typing %>%