Evanbook

A Template for quarto

Evan Zhou

May 4, 2025

Table of contents

Index

Evanbook

A modern, modular, and professional Quarto Book template

Created by Evan Zhou ·
Powered by Quarto

Start Reading →

Preface

> **Note**
>
> Welcome to **Evanbook**, a modern and modular Quarto Book template designed to support your academic, technical, and documentation writing workflows.

Whether you're creating:

- a package manual,
- a scientific project report, or
- an educational book,

Evanbook provides a clean and extensible starting point with opinionated structure and styling.

Why this template?

This book template was born out of the need for: - **structured content organization** - **customizable and elegant visual design** - **R + Python + Markdown compatibility** - **multi-chapter publishing and web deployment**

Who should use Evanbook?

- **Researchers** writing technical documentation
- **Instructors** creating course material
- **Developers** building package or API documentation
- **Writers** seeking a professional digital publishing format

---

Structure of this book

Each part of this book demonstrates how to structure a multi-part Quarto project:

1. **Getting Started** — project setup and tool basics
2. **Workflow Modules** — reusable writing components
3. **Advanced Topics** — deployment, extensions, customization
4. **Appendices** — reproducibility, references, tooling

Feel free to customize this page with your own motivations and audience-specific notes.

Part I

Getting Started

1 Installation and Setup

Before using this book template, ensure your environment is properly configured.

1.1 1.1 Install Quarto

You can install Quarto from the official website:

- https://quarto.org/docs/get-started

```
# Windows/macOS/Linux
# Download from the website and follow the installer instructions
```

Check version:

```
quarto --version
```

1.2 1.2 Install R and RStudio

If you're using R:

- Install **R** from: https://cran.r-project.org
- Install **RStudio IDE** from: https://posit.co/download/rstudio-desktop

1.3 1.3 Install Python (optional)

If your book uses Python code blocks, install Python (3.8):

```
# Install via Anaconda or python.org
python --version
```

Optional: set up a virtual environment using `venv`, `conda`, or `virtualenv`.

1.4 1.4 Recommended Tools

Tool	Purpose
Quarto	Rendering and publishing
R / Python	Language support
VS Code	Lightweight editing
Git + GitHub	Version control, deployment

1.5 1.5 Initialize a New Project

To create a new project based on this template:

```
quarto use template evanbio/evanbook
cd evanbook
quarto preview
```

2 Quarto Basics

This chapter introduces the key components of a Quarto document and how to work with them interactively.

2.1 2.1 Anatomy of a `.qmd` file

Each Quarto document starts with a **YAML header** (metadata block), followed by **markdown text** and **code chunks**.

```
---
title: "Example Document"
author: "Evan"
format: html
---

# Heading 1

Some text.
```

```
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
```

```
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

You can use `r`, `python`, `bash`, and more as code chunk engines.

2.2 2.2 Markdown Elements

Quarto supports standard Markdown syntax:

- **Bold**, *italic*, `code`
- Lists: `- item`, `1. item`
- Links: `[text](url)`
- Images: `![](image.png)`
- Tables:

Name	Value
A	100
B	200

2.3 2.3 Code Chunks

Code chunks are delimited by triple backticks and `{lang}`:

```r
# R chunk
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
```

```
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

Use `echo`, `eval`, `warning`, `message` chunk options to control output.

2.4 2.4 Rendering Documents

To preview your book:

```
quarto preview
```

To render all chapters:

```
quarto render
```

You can also render individual `.qmd` files if needed.

2.5 2.5 Directory Structure

Typical Quarto book structure:

```
evanbook/
  _quarto.yml
  styles.css
  index.qmd
  01_installation.qmd
  02_quarto-basics.qmd
  ...
```

Part II

Workflow Modules

3 Data Cleaning Essentials

Data cleaning is one of the most critical steps in any data workflow.
This chapter outlines reusable strategies for cleaning and preparing your data for analysis.

3.1 3.1 Inspecting Data

Use tools like `summary()`, `str()`, or `skimr::skim()` to understand the structure and quality of your dataset.

```
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

3.2 3.2 Handling Missing Values

Common approaches include:

- Removing rows with `NA`
- Imputing values
- Using domain knowledge to infer missing information

```
# Drop rows with missing values
clean_data <- na.omit(raw_data)
```

3.3 3.3 Removing Duplicates

```
clean_data <- distinct(raw_data)
```

Use `duplicated()`, `distinct()` (dplyr), or `janitor::get_dupes()`.

3.4 3.4 Standardizing Column Names

```
library(janitor)
clean_data <- raw_data |> janitor::clean_names()
```

3.5 3.5 String Cleaning

Use `stringr::str_trim()`, `str_to_lower()`, etc., to normalize text.

```
library(stringr)
df$name <- str_to_title(str_trim(df$name))
```

4 Data Visualization Basics

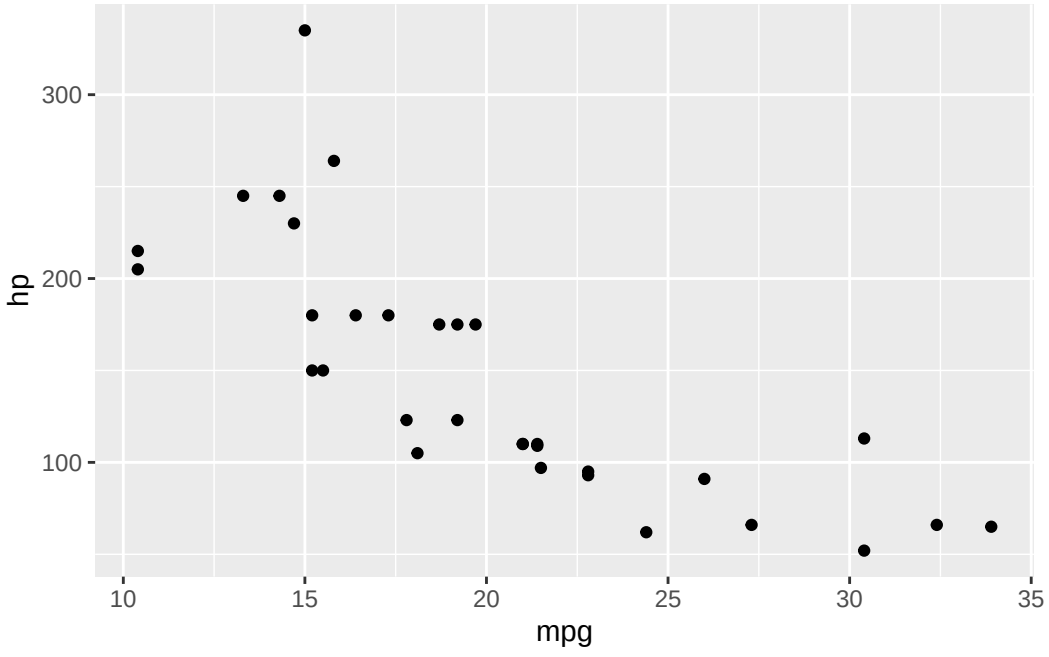
This chapter introduces the basics of data visualization using **ggplot2**.

> *Visualizations turn raw data into insights.*

4.1 4.1 The Grammar of Graphics

The core idea of `ggplot2` is to build plots layer by layer:

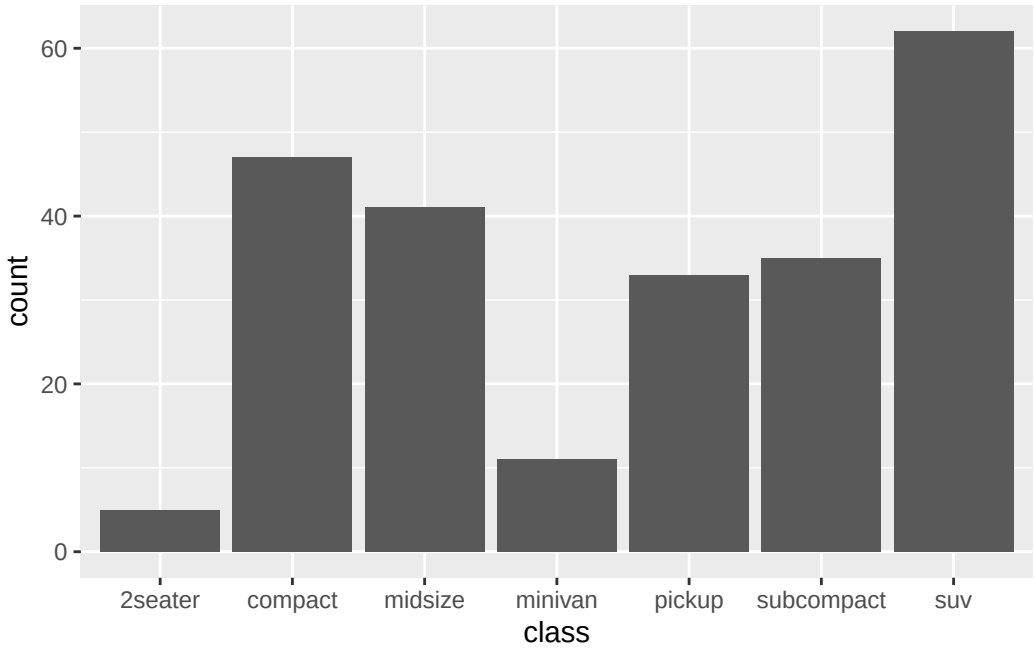
```
library(ggplot2)

ggplot(mtcars, aes(x = mpg, y = hp)) +
  geom_point()
```



4.2 4.2 Common Plot Types

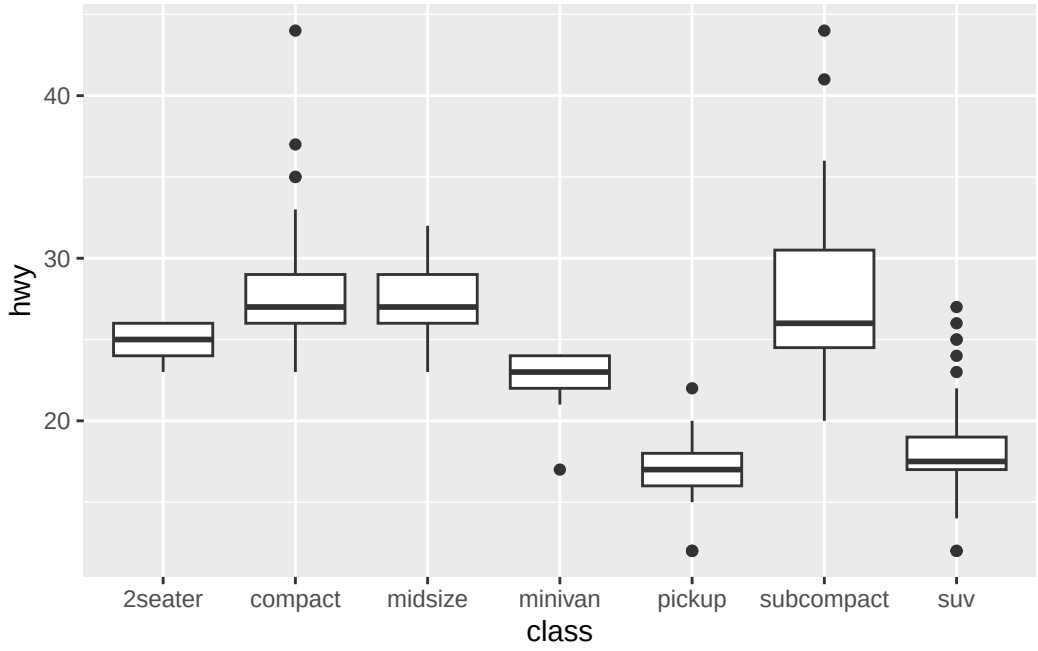
4.2.1 Bar Chart

```
ggplot(mpg, aes(class)) +
  geom_bar()
```



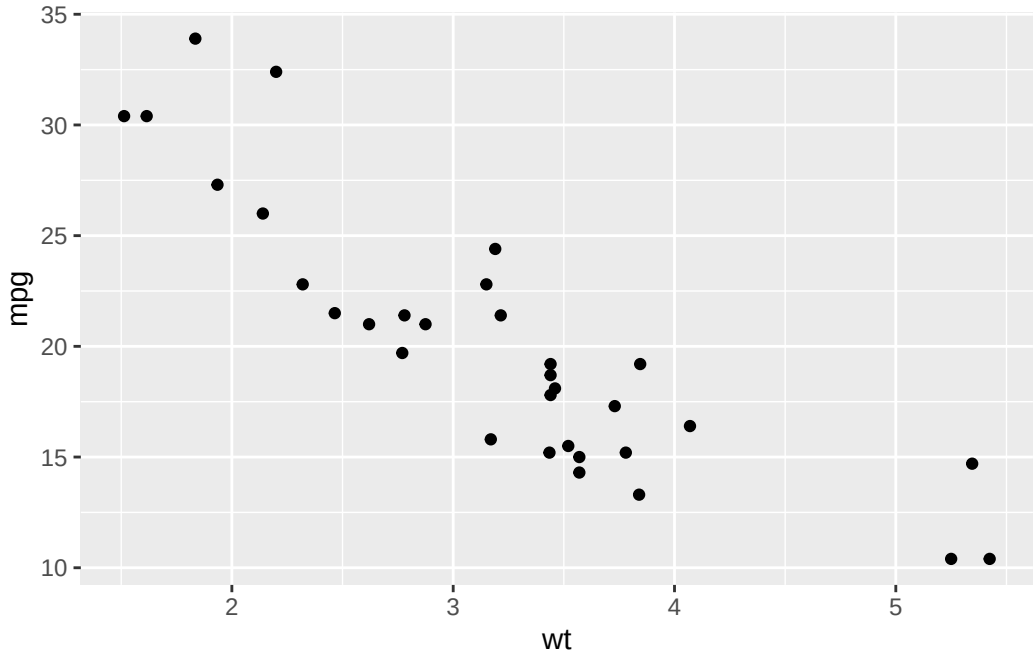
4.2.2 Boxplot

```
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot()
```



4.2.3 Scatter Plot

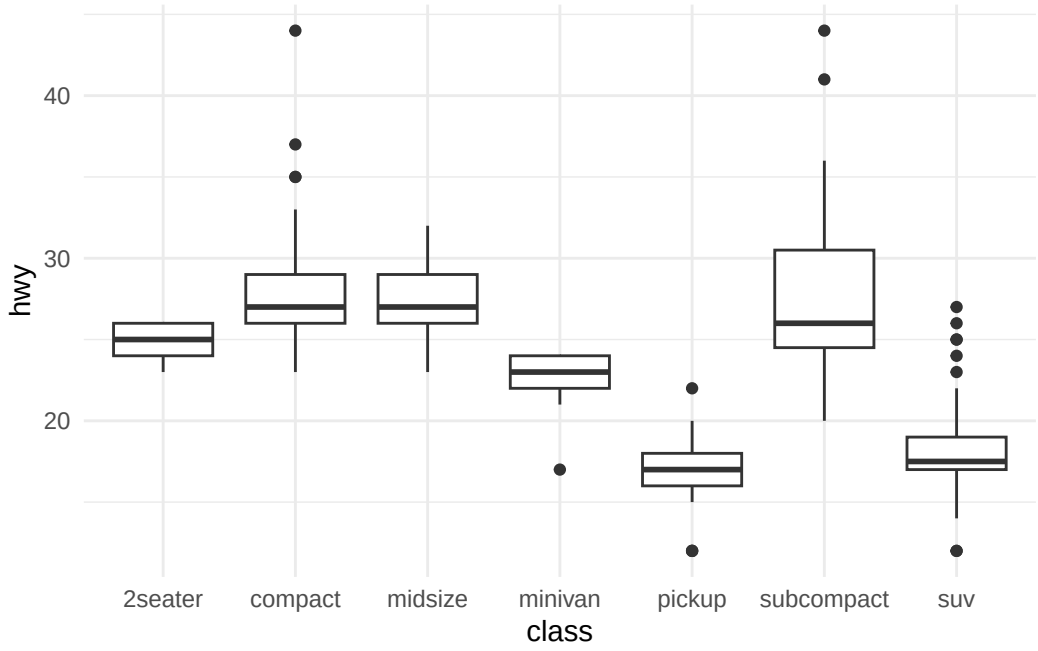
```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
```



4.3 4.3 Themes and Aesthetics

Use `theme_minimal()`, `theme_classic()` and control colors, sizes, and fonts.

```
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot() +
  theme_minimal()
```



4.4 4.4 Exporting Figures

Save your plot with:

```
ggsave("my-plot.png", width = 6, height = 4, dpi = 300)
```

5 Reporting and Output Formats

Quarto supports multiple output formats with consistent source content.
This chapter explains how to control the report format and export behavior.

5.1 5.1 Rendering to HTML

The default output format is HTML:

```
format:
  html:
    toc: true
    number-sections: true
```

Render the report:

```
quarto render
```

Or preview:

```
quarto preview
```

5.2 5.2 Rendering to PDF

You can also output to PDF using LaTeX:

```
format:
  pdf:
    documentclass: article
    toc: true
```

> Ensure LaTeX is installed. Use TinyTeX via `tinytex::install_tinytex()` in R.

5.3 5.3 Rendering to Word or Markdown

```yaml
format:
  docx: default
```

Useful for collaborators who use Microsoft Word.

5.4 5.4 Multi-format Output

You can specify multiple formats in one YAML:

```yaml
format:
  html: default
  pdf: default
  docx: default
```

Use the dropdown in the top right corner of each page (HTML only).

5.5 5.5 Controlling Code Display

Use chunk options like:

```
summary(df)
```

To hide code, suppress warnings, or customize output.

5.6 5.6 Styling Reports

Link a custom stylesheet in `_quarto.yml`:

```yaml
format:
  html:
    css: styles.css
```

Part III

Advanced Topics

6 Interactive Content in Quarto

Quarto supports various forms of interactive content to enrich your document and engage your audience.

6.1 6.1 Using htmlwidgets

You can embed interactive widgets from R directly:

```
library(plotly)
plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length, color = ~Species, type = 'scatter
```

Other useful widgets:

- `DT::datatable()`
- `leaflet::leaflet()`

These are rendered as interactive HTML widgets in the output.

6.2 6.2 Embedding Shiny Apps

You can embed Shiny apps inside a Quarto document:

```
selectInput("var", "Variable", choices = names(mtcars))
renderPlot({
  hist(mtcars[[input$var]])
})
```

> Requires `runtime: shiny` in the YAML header.

6.3 6.3 Observable JS Blocks

Use Observable blocks to embed JavaScript-powered interactivity:

```
Plot.plot({
  marks: [
    Plot.dotY([1, 2, 4, 8, 16])
  ]
})
```

> Quarto automatically includes Observable runtime when needed.

6.4 6.4 Code Folding and Reveal

Enable collapsible code blocks:

```
format:
  html:
    code-fold: true
```

Use `revealjs` format for slideshow-like interaction:

```
format:
  html:
    code-fold: true
```

6.5 6.5 User Input + Reactive Output (Shiny)

Use input/output bindings to create small tools or exploratory panels.

```
sliderInput("bins", "Number of Bins", min = 1, max = 50, value = 30)
renderPlot({
  hist(faithful$waiting, breaks = input$bins)
})
```

7 Extending Quarto

Quarto is designed to be extensible. This chapter introduces how to install, use, and create extensions such as themes, filters, and shortcodes.

7.1 7.1 What is a Quarto Extension?

An extension is a reusable bundle that can include:

- Themes (CSS, SCSS)
- Lua filters
- Shortcodes
- Project templates

They can be published to GitHub and installed via `quarto add`.

7.2 7.2 Installing Extensions

To install an extension:

```
quarto add quarto-ext/fontawesome
```

This will download the extension and add it to your project's `_extensions/` folder.

Other examples:

```
quarto add quarto-ext/clean
quarto add quarto-ext/authoring
```

List installed extensions:

```
quarto list
```

7.3 7.3 Using Extensions

Example: using the Font Awesome icon extension:

This is a book icon

Themes:

```
format:
  html:
    theme: [cosmo, clean]
```

7.4 7.4 Creating Your Own Extension

You can create a new extension skeleton with:

```
quarto create extension my-ext --type=filter
```

Quarto supports extension types:

- `format` (custom format template)
- `filter` (Lua filter logic)
- `shortcode` (templating)
- `theme` (CSS)

  For full reference: https://quarto.org/docs/extensions/

7.5 7.5 Building a Quarto Template

If you'd like to turn Evanbook into a reusable template:

```
quarto use template evanbio/evanbook
```

Or share a structure via GitHub with a `template.qmd`, `_quarto.yml`, and `README.md`.

8 Deployment and Publishing

Once your Quarto book is complete, you can publish it as a website for others to read and reuse.

8.1 8.1 Rendering the Site

Build your book locally with:

```
quarto render
```

The output will be saved in the `output-dir`, typically `_book/`.

Preview locally:

```
quarto preview
```

8.2 8.2 Publishing to GitHub Pages

8.2.1 Step 1: Push your book to GitHub

Make sure your `_book/` folder is **not ignored** if you're pushing directly.

```
git add .
git commit -m "deploy: render book"
git push origin main
```

8.2.2 Step 2: Deploy via `gh-pages` branch

You can use the official GitHub Actions:

```
quarto publish gh-pages
```

Automatically creates and deploys to the `gh-pages` branch.

8.3 8.3 Publishing via Netlify

8.3.1 Step 1: Link GitHub repo in Netlify

8.3.2 Step 2: Set build command

```
quarto render
```

Step 3: Set publish directory:

```
_book

Push to GitHub → Netlify auto-builds → auto-publishes.
```

8.4 8.4 Publishing via Vercel (Optional)

- Upload static `_book/` to `/public`
- Or use Vercel CLI

Note: Vercel is better for dynamic content (like APIs); Netlify is preferred for books.

Part IV

Appendices

9 Reproducibility and Project Organization

Ensuring reproducibility is a cornerstone of good scientific and analytical practice.

9.1 A1.1 Organizing Your Project

- Use a consistent folder structure: `data/`, `scripts/`, `output/`, `figures/`
- Track raw vs. processed data separately
- Always include a `README.md`

9.2 A1.2 Managing Dependencies

9.2.1 R:

Use `{renv}` to lock your R environment:

```r
install.packages("renv")
renv::init()
```

9.2.2 Python:

Use `virtualenv`, `venv`, or `conda` for isolation.

9.3 A1.3 Quarto Tips

- Use `freeze: auto` in `_quarto.yml` to cache outputs
- Always set `execute:` chunk-level control
- Consider storing all outputs in `/output/` and images in `/figures/`

10 Using Git and GitHub

Version control is essential for collaborative work and tracking progress.

10.1 A2.1 Basic Git Workflow

```
git init
git add .
git commit -m "Initial commit"
git push origin main
```

10.2 A2.2 Recommended .gitignore

```
_book/
*.html
.Rhistory
.Rproj.user/
.DS_Store
```

10.3 A2.3 GitHub Integration

- Create repo at https://github.com
- Use quarto publish gh-pages to deploy
- Use README.md + LICENSE + description for open projects

10.4 A2.4 Git in RStudio

- Use built-in Git tab
- Or install usethis:

```
usethis::use_git()
usethis::use_github()
```

References