PsyTeachR Book Template Workshop

James Bartlett

2024-07-19

Table of contents

Workshop Overview **3**

1 Preparation before the workshop **4**

1.1 Creating an online book workflow 4

1.1.1 Creating the book 4

1.1.2 Editing the book 4

1.2 Download R and RStudio 5

1.2.1 Installing Base R 5

1.2.2 Installing RStudio 5

1.2.3 Install the `booktem` R package 6

1.3 Download git and github desktop 6

1.4 Finished 7

2 Workshop plan **8**

2.1 Example book 8

2.2 Creating the book template 8

2.3 Tour of the Quarto book template 9

2.4 Creating a github repository 12

2.5 Navigating github and github pages 12

2.5.1 Publish to github pages 13

2.6 Commiting and pushing changes 14

2.7 Start working on your own book! 15

3 Worked examples for Quarto and booktem features **16**

3.1 Quarto guide 16

3.2 Worked examples 16

3.3 Useful features 16

3.3.1 Markdown 16

3.3.2 Code chunks 21

3.3.3 Callout blocks 22

3.3.4 References 24

3.3.5 `webexercises` interactive questions 25

3.3.6 Embedding files to download 27

3.3.7 Adding a glossary 27

References **29**

Workshop Overview

In this informal workshop, I will walk through the steps to use the new PsyTeachR book template `booktem` to create online books. This uses Quarto - a new version of R Markdown - to combine code with text. You can create a very simple Quarto book within R Studio as standard (I wrote this workshop guide using it) but the PsyTeachR template has more complete settings, placeholders, and conventions for our typical needs.

If you are unfamiliar with the PsyTeachR suite of books, you can see the collection here: https://psyteachr.github.io/.

Some of these examples use the old R Markdown version, so you can see an example of the new Quarto template in the Applied Data Skills book developed by Dr Emily Nordmann and Prof Lisa DeBruine (who also develops the PsyTeachR book template).

When you are writing your book independently, it's useful to look at the code behind examples to see how they create certain features. For example, Dr Wil Toivo and I are rewriting the Fundamentals of Quantitative Analysis book and you can see our github repository for inspiration.

All of these books are designed to teach data skills as they demonstrate and render code well, but you can use them for any materials using Markdown features. For example, I host the course information for RM1 using the book template.

Once you knit them and push any changes to github, you get a url to link to them like any website. This is great for sharing with students as it's free and simple to access, and it's great for you and demonstrating impact as you can share your work with others for #impact.

Intended Learning Outcomes (ILOs)

By the end of the workshop, you will be able to:

- Create an online book using the `booktem` R package in RStudio.
- Use the basic functions of git/github to commit changes and host the book using github pages.
- Communicate your ideas to your target audience using features such as Markdown formatting, code chunks, referencing, glossaries, and interactive questions.

1 Preparation before the workshop

Before we can get to the fun stuff, you will need to download a few pieces of software to your computer to focus on creating in the workshop. If you have any problems, just message me on Teams.

1.1 Creating an online book workflow

For an overview to show why you need the following the pieces of software, you will follow this workflow to create and edit the book.

1.1.1 Creating the book

1. Install the `booktem` R package.
2. In RStudio, use the `create_book()` function to create a new project in your working directory containing the initial template files.
3. Edit the basic details of your book, like the title, author(s), and description.
4. Create a new github repository for your book folder.
5. Publish the initial repository so the code is available on github.
6. Activate github pages to render the book online via a URL.

1.1.2 Editing the book

1. Open the book .Rproj file to start working on the book in RStudio.
2. Edit or add chapter files in RStudio, specifying their order in the `_quarto.yml` file.
3. Render individual chapter files as you work on them. Render `index.qmd` when you want to update all the chapters, or render them one by one to ensure all new changes are available.
4. Add commits at key points using git/github to mark milestones with useful commit messages.

5. When you want the book updating, push the changes to be available on github and your book URL after a short delay.

1.2 Download R and RStudio

If you are new to R/RStudio, you will need to install both pieces of software which is *normally* pretty straightforward. You might find this YouTube video here useful.

If you are a more experienced R/RStudio user, just make sure you update your version of R/RStudio as the `booktem` package will only work for recent versions as Posit are rapidly developing Quarto. I am currently using R version 4.4.1 (2024-06-14) and RStudio 2023.6.2.561, but the more recent the better.

! Important

From what I can tell by testing it out, it does not work with the online server version of RStudio. I could not get it to download the `booktem` package and I am not sure how well it works with github, so you will need a local installation.

1.2.1 Installing Base R

Install base R. Choose the download link for your operating system (Linux, Mac OS X, or Windows).

If you have a Mac, install the latest release from the newest `R-x.x.x.pkg` link (or a legacy version if you have an older operating system). After you install R, you should also install XQuartz to be able to use some visualisation packages.

If you are installing the Windows version, choose the "base" subdirectory and click on the download link at the top of the page. After you install R, you should also install RTools and use the "recommended" version highlighted near the top of the list.

If you are using Linux, choose your specific operating system and follow the installation instructions. If you use Linux, you probably do not need help from me for installing software and I will probably not know how to help you as I have never used Linux.

1.2.2 Installing RStudio

Go to rstudio.com and download the RStudio Desktop version for your operating system. It should recognise your operating system and allow you to download via the blue Download button, but you can look for previous versions if you need one.

1.2.3 Install the `booktem` R package

The PsyTeachR book template is contained within a package hosted on Lisa's github called `booktem`. To install the package, run the following code in the Console of RStudio:

```r
devtools::install_github("debruine/booktem")
```

If you are new to R/RStudio, you probably have no user packages installed, so you will get a prompt to allow `booktem` to install the other R packages it depends on to work. This might take a few minutes, so go and enjoy yourself a hot drink.

Caution

One of these messages might say something like "do you want to compile packages where there is a binary version" and give you several options to select. You will only be able to select 1 if you have Rtools downloaded on a Windows computer as you need developer tools to do this. Macs should not need any additional software to compile binary packages. These binary versions are normally a little more recent, so its useful to install them if possible.

If you are a more experienced R/RStudio user, you might be prompted to update your R packages that `booktem` depends on. Obviously use your judgement if you are in a place to update your packages, but the book template might not work with older packages.

1.3 Download git and github desktop

Potentially the most unfamiliar element of this process will be working with git and github. If you have not used it before, git is a version control system which tracks file changes on your computer (like OneDrive but for code). github is an online system which uses git to host those changes and make your code available online. There is a graphical user interface called github desktop which I use and will demonstrate. If you want to use the command line version of git/github, you probably do not need me to show you how.

There is an installation guide on github, but we have access to a fantastic resource developed by our colleagues in Mathematics and Statistics. They developed an accessible online course on Moodle introducing staff and students to version control using git and github. You will need the enrollment code **git_psych_24** to access the course. If you use it, please consider completing their feedback form on the Moodle page so they can improve the resource in future.

There are 7 units in total which do not take very long, but for the purposes of this workshop, I would consider 1 and 2 as **essential** for downloading git/github desktop and using it as a single user to track changes. Reaching this point will be super helpful for following along in the workshop as the terms will be more familiar to you like repositories, commits, and pushing changes.

If you have time, completing unit 3 will give you everything you need if you are only using it on your own. If you plan on writing the book in a team, units 4 and 5 cover being a group project user, then 6 and 7 for advanced features.

Apply for education membership

The standard version of github should meet all your needs, but by working at a university you are eligible to apply for an education membership. If you are interested you can find out more on the github education site for teachers. For example, with an education profile, you can get unlimited private repositories.

1.4 Finished

You are ready to make a book in the workshop! See you all on the 1st of August.

2 Workshop plan

To follow along in the workshop if you fall behind, we will complete the following steps.

2.1 Example book

For an example of the new book template, we will explore the new Fundamentals of Quantitative Analysis book. I have edited the first two chapters to use new Quarto features, so they can provide inspiration for what you can do with the book template.

2.2 Creating the book template

If you have not followed the preparation instructions yet, you need R/RStudio installed to your computer, the `booktem` package installed from Lisa's github, and git/github desktop installed on your computer. I will be demonstrating how to use github desktop rather than the command line.

The first step is thinking about where you want your book folder stored on your computer, where all the files will live. I have a folder within `Documents` called `git_repos` where I store all my git repositories away from OneDrive (see below). You do not need to create a folder for the book itself as the function will do that for you, but you need somewhere for that folder to live.

⚠ Do not create github repositories within OneDrive

We have not reached the github step yet, but as you think about where you want your folder for the book, please **do not** use a folder within OneDrive. Sometimes it works, but most of the time it causes chaos as OneDrive is trying to track changes, github is trying to track changes, which ends in them fighting over file permissions.

Once you have a folder your book can live in, open RStudio and set your working directory to this folder, for example from the top menu `Session >> set working directory >> Choose directory` and navigate to this folder.

Once RStudio knows where you want your working directory, you can create the book using the following code in the console and editing accordingly before you run the code. Do not worry though, you can edit all of these later, but this will create the initial version.

```
# We first need to load the booktem library, assuming its installed properly
library(booktem)

create_book(path = "your_book_file_name", # If you set your working directory, you should no
            title = "My book title", # The main title of your book
            authors = list( # You need a new line for any additional author, or delete the au
              c("Author 1 first name", "Author 1 last name", "Author 1 ORCiD"),
              c("Author 2 first name", "Author 2 last name", "Author 2 ORCiD"))
            )
```

Once you press run, you should get a bunch of progress messages from `Setting up project....` Once it's finished, your book will open in a new session and you will see the rendered version appear in your default internet browser.

Congratulations, you have a book skeleton to work with!

2.3 Tour of the Quarto book template

1. `_quarto.yml`

We will start by exploring the `_quarto.yml` file where you can edit all the details for your book, like the title, description, author(s), and the licence. In the new Quarto book template, this is also how you control the order of chapters.

> What is a .yml file?
>
> YAML / .yml are configuration files for programs which must follow specific formatting conventions.

Within the .yml file, I will highlight key features in: project, book, bibliography, csl, and format.

2. index.qmd

Index will be the opening page for the link to your book, so this will typically include an overview of what your book contains, who to contact for problems/questions etc.

3. `R/` folder

The `R/` folder is where you can save bits of R code that your book relies on. There is some code in here that Lisa has worked on to help with certain functionality, like how the glossary looks.

4. `include/` folder

The `include/` folder is a similar idea to `R/`. It has a bunch of files the book uses for formatting and any chapters would be able to access stuff here. For example, a .bib file for your references and a .csl file for the style of your references.

5. `docs/` folder

The `docs/` folder is where the rendered .html versions of your book will update to. The process behind creating books in Quarto is writing them in Markdown, then Markdown is converted to .html. When we add the book to github pages, this is the folder you point it to as the source for how it appears as a webpage.

6. Licence

By default, `booktem` gives the books a CC-BY-SA-4.0 licence. This is a Creative Commons licence which states people can adapt your materials but they must provide you with credit. You can learn more about different types of Creative Commons licences online. If you want to state a different licence depending on your materials, you can update the text in the Licence file and in the .yml.

7. Chapter files

By default, you get an example index.qmd and one chapter .qmd for an example. The example chapter has a bunch of advice on what to edit and features like I will demonstrate, but you can delete this text or create your own files to start writing chapters.

You will need one level 1 header (`# Chapter title`) to start the file, and that will be the name of your chapter at the top of the page and in the table of contents.

Warning

Make sure you only use one level 1 header per chapter. If you try and add multiple within one file, it will think they are separate chapters and try and split them when it renders, making it look weird.

Once you start adding multiple chapters, remember to update the .yml file for the order you want them in your book.

Tip

If you are converting a previous book to the new template, there is a handy little function **rmd2qmd**. This copies .Rmd files and renames them to .qmd. The function looks like this

```
rmd2qmd(from_path = "",  # file path where your .Rmd files are
        to_path = "") # file path where you want your new .qmd files to go
```

where you specify a file path to access the old .Rmd files and a file path to where you want the new .qmd files saving. Keep your working directory in mind as you will probably be starting from your book folder at this point. I usually save my old .Rmd files in a folder within the new book directory, then save the new .qmd files to the new book main directory.

8. Rendering files

Once you have finished editing or you want to check how it looks, you need to click Render for Quarto to process the code and turn it into something pretty. Once you click Render once, you can open it up in the browser and keep checking as you make edits. When you want it to rerender, click Reload the page and it will show your new edits.

Tip

The single best part of Quarto and the new book template is you can keep rendering and checking what your work looks like in the flesh. Previously, you had to render the whole book to check how it rendered, but now you can keep updating the browser to see what your changes look like.

Caution

If you introduce an error, you will get an error and red box on the screen to highlight Quarto cannot render the book. If you look in the Background Jobs tab in the console below, you should get an error message for the source of the problem if you are unsure what you did wrong.
After an error, you will need to press Render again rather than just refreshing the browser.

These are the key components to make your book. Until this point, these edits all exist on your own computer. But now it is time to track your code using github and make your book accessible online via github pages.

2.4 Creating a github repository

Once you have a working barebones version of your book ready to go, it's time to associate your book folder with a github repository and start some version control. If you want another resource, you can see the github documentation online.

In future, you could actually start with this part. You can create a new folder, create a repository using this new folder, and then use the `create_book()` function to add the book file to. However, we started by creating the book first, so we need to create a repository for an existing folder without a git component.

In the github desktop application, click `add >> Create a New Repository` and complete the details.

⚠ Seriously, do not create github repositories within OneDrive

As a reminder, please **do not** use a folder within OneDrive for your github repository.

- Name: This will be the name of your repository on github, so call it something short but sensible.
- Local path: Click Choose… and navigate to your book folder. You want the path to be the main folder your book lives in.

The other fields you can edit later, so click Create Repository when you are ready.

Your newly minted repository should be showing as the Current Repository. This exists on your computer, but it is still not available online. You need to click Publish repository, and that will push all of your files to github and be available online.

2.5 Navigating github and github pages

Now your files are available online, navigate to your github account and find your new repository.

I will provide a little overview in the workshop of key things to look out for and what each tab contains.

If you are only interested in using github to work on a book, the key tabs are Code and Settings. In Code, you will see all your files you published. These will all be the same as what you created on your computer. This is the idea behind version control and storing all your code/files like OneDrive.

In Settings, this is where you can edit things about your repository. If you plan on working on your book in a team, you can add collaborators by adding their github profile. They will then receive an email saying you have invited them to collaborate on their github repository. After they accept, they can pull the repository and start working on it too.

Warning

Working collaboratively is one of the main motivations behind git and github, but it can be tricky if you are unfamiliar with more advanced features like merging and clashes. Before you start editing the same repository with someone, I *heavily* recommend completing units 4 and 5 of the version control Moodle course provided by Maths and Stats.

Within Settings is the key section we need to get your book available online.

2.5.1 Publish to github pages

In Settings, navigate to Pages within Code and automation. Under Build and deployment, this will be set to none by default. You must click the drop down, choose Main and select /docs. You will remember /docs is where we store all the html versions of the book files, so you are pointing Github pages here as the source for your book.

When you press Save, this will start building your book. It will not be available immediately and will take a few minutes. When it is ready, at the top of the page, it will say "Your site is live at..." with your new URL you can click on and open. In the Actions tab, it will also show as a green tick when it has finished building.

Congratulations! After seeing your rendered book for the first time, this is the second most satisfying part as you can see everything is working.

Add shortcuts to your book

Once the site is live, I recommend adding the link to two places. First, save it as a browser shortcut so you can quickly access it outside of github. Second, return to the Code tab. Click the cog icon next to Above on the right side and tick to add your website from github pages. This will show the URL to your book on the Code tab for easy access from here.

Customise the URL

By default, the URL to your book will be your github username + .github.io + your repository name. If you want to get fancy, you can add a custom domain from within Settings and Pages if you have bought one.

Warning

If you try and push a change that contains an error and your book does not render, you will get a red cross in Actions saying your book did not build and you will receive an email warning you about it too. Just go back to the book files in RStudio and fix any errors you are getting before you push the updates again.

2.6 Commiting and pushing changes

At this point, you have everything you need for the book workflow. As you work on your book, you will go through the workflow of:

1. Open .RProj and edit your book in RStudio, either by editing your past progress or adding new files.
2. When you hit a milestone you want to record, in github desktop tick all file changes or specific files, and add a commit message (and longer description if necessary).
3. If you are continuing to work on the book, keeping editing and committing.
4. When you are ready to push changes to github and github pages, push your commits.

Before we have time to start working on your newly minted books, I will end on a couple of warnings and tips.

Caution

Remember just committing changes will do nothing to your github repository and book link. You need to push those committed changes for them to be available on github and used to build the book in github pages. It takes a few minutes to rebuild, so do not worry if you do not immediately see the changes.

Reverting changes

One of the main features I use way too infrequently is reverting changes when something goes wrong. The idea behind version control is you save your work at specific milestones, where you can add commit messages that describe key changes you make. If you make a change that breaks something since your last commit, you can revert the changes to a previous version. To do this, go to github desktop, click History, and you will see all your commit history. Identify the last commit you want to revert to, right click on it, and select Revert Changes in Commit.

2.7 Start working on your own book!

Start working on your own book in the remaining time we have together.

See the next chapter on Quarto features and book conventions for inspiration / example code for the kind of features you can use.

Do not forget to make your own hex sticker for for extra panache!

3 Worked examples for Quarto and booktem features

As you work on your own book, you might find the following features useful for inspiration.

3.1 Quarto guide

To check different features of Quarto, you can see an extensive user guide online.

3.2 Worked examples

If you look through these books for inspiration, you can check the code to see how they do it.

Dr Wil Toivo and I are rewriting the Fundamentals of Quantitative Analysis book and you can see our github repository.

You can also see the Applied Data Skills book developed by Dr Emily Nordmann and Prof Lisa DeBruine, along with their github repository.

3.3 Useful features

3.3.1 Markdown

Quarto still uses Markdown for formatting, so you can see this part of the Quarto guide for Markdown Basics.

3.3.1.1 Headers

The book template table of contents goes by default to level 3 headers (but you can go to level 6 headers if you *really* want), so keep in mind what will be a logical nesting of headers, sub-headers etc. You add a hash for each level of header:

```
# Level 1 header

## Level 2 header

### Level 3 header
```

3.3.1.2 Text formatting

You make text *italics* by surrounding it with one star (`*italics*`), or **bold** by surrounding it with two stars (`**bold**`).

If you want to give text code formatting, you can add back ticks around it.

```
`example code`
```

produces:

`example code`.

If you want to add bullet points, you can use `-` or `*`:

```
- Bullet point 1

- Bullet point 2

- Bullet point 3
```

- Bullet point 1
- Bullet point 2
- Bullet point 3

The same applies to numbered lists:

```
1. List 1

2. List 2

3. List 3
```

1. List 1
2. List 2
3. List 3

Or even sub-lists with a little indent:

```
1. List 1

    1. Sublist 1

    2. Sublist 2

2. List 2
```

1. List 1
    1. Sublist 1
    2. Sublist 2
2. List 2

3.3.1.3 Links

You can add hyperlinks with the form:

```
[hyperlinks](https://quarto.org/docs/authoring/markdown-basics.html#links-images)
```

where the writing in square brackets is the text, and the link goes in the round brackets. By default, these link within the current page which I find infuriating. So, you can add a little html code to open a new tab with the link:

```
[hyperlinks](https://quarto.org/docs/authoring/markdown-basics.html#links-images){target="_bl
```

3.3.1.4 Internal hyperlinks

If you want to reference a chapter or section of your book, you can create internal hyperlinks through curly brackets and a hash. For example, if I wanted to point you back to the Workshop Plan chapter, you first need to add a tag on the chapter heading:

```
# Workshop plan {#workshop_plan}
```

You can then use the tag to create a link with a similar format to URL hyperlinks:

```
...back to the [Workshop Plan](#workshop_plan) chapter
```

3.3.1.5 Images

There are different ways to add images, where the Markdown version uses a similar format to links which I will demonstrate through this duck:

```
![This is a duck.](images/Duck.png)
```

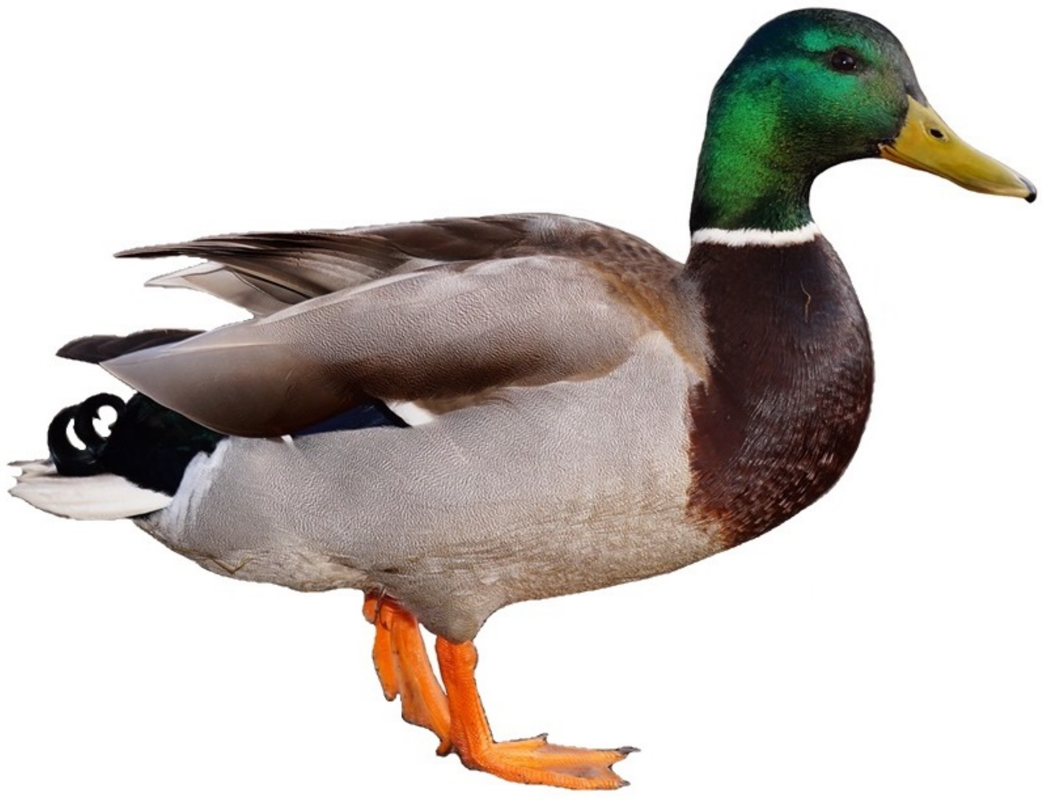

Figure 3.1: This is a duck. If Helena is reading this, yes it has a creative commons licence.

There are some cool new Quarto features making it easy to combine multiple images. For example, we can add two ducks and specify we want them in two columns.

```
::: {#fig-duck layout-ncol=2}
![](images/images/Duck.png)
![](images/images/Duck.png)
Duck 1 (left) and Duck 2 (right).
:::
```

You can also reference figures to add little hyperlinks and automatically number them. The title after the hash must start with `fig` to be registered as a figure, and you can also add `tbl` to number tables separately.

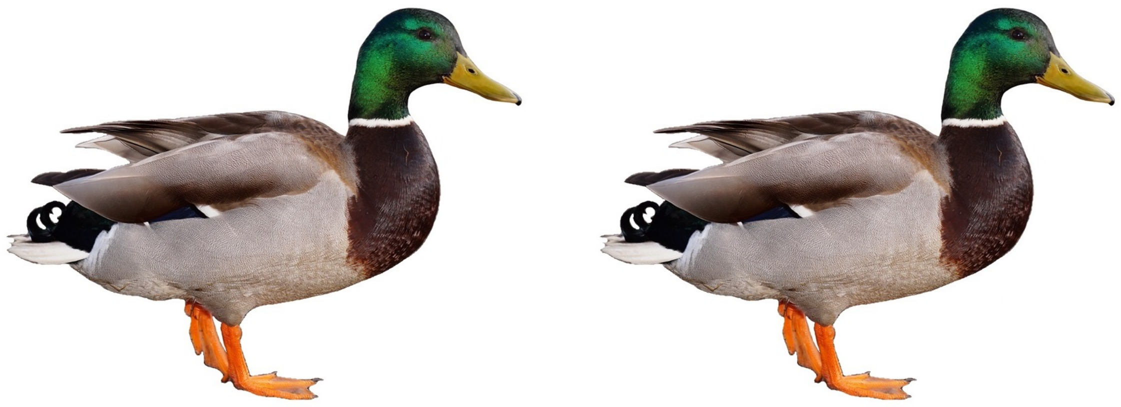

Figure 3.2: Duck 1 (left) and Duck 2 (right).

Figure 3.2 showed two ducks side by side. You do not even need to type Figure: `@fig-duck`.

You can also use `knitr` to add figures, and using code chunks has some new handy features using tags. They work in a similar way to code options, but make it easier to add longer captions etc, as shown in Figure 3.3.

~~~
```{r}
#| label: fig-img-duck
#| fig.cap: "This is a longer caption about our beloved duck."
#| fig-alt: "You can also add alt text to images."

knitr::include_graphics("images/Duck.png")
```
~~~

3.3.2 Code chunks

If you are making a book to show code, there are a couple of features that might be useful.

Adding code chunks will by default show both the code and output:

~~~
```{r}
rnorm(n = 5, mean = 10, sd = 2)
```
~~~

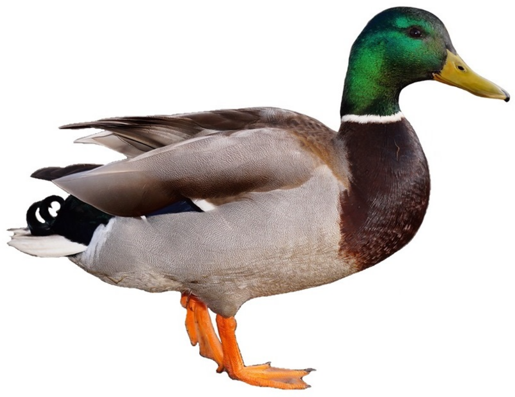

Figure 3.3: This is a longer caption about our beloved duck.

```
rnorm(n = 5, mean = 10, sd = 2)
```

```
[1]  7.420800  9.105154 10.412665  8.993307  5.472392
```

There are several features you can edit by adding different options. For example, if you do not want to show the code, you can set `echo=FALSE` after the r `{r echo=FALSE}`:

```
[1] 11.051400  8.809330  8.599247 10.375750  9.672140
```

If you want to demonstrate code but not execute it - such as to demonstrate inaccurate code, you can set `eval=FALSE` after the r `{r eval=FALSE}`:

```
rnorm(n = 5, mean = 10, sd = 2)
```

3.3.3 Callout blocks

My personal favourite features, you can highlight content with callout blocks. These range from notes that people might find interesting, to warnings that something could go mortally wrong.

```
::: {.callout-note}
These are notes.
:::
```

ℹ Note

These are notes.

You can change the title by using hashes within the callout. They count as real headers in the Quarto outline. So, if you use one hash, it looks like a level one header which deeply disturbs me, so I like to use four hashes to make more sense in the chapter structure.

```
::: {.callout-note}
#### Look at my interesting title
These are notes.
:::
```

ℹ Look at my interesting title

These are notes.

You can also make the box collapse by default, which can be handy to hide solutions or obscure tangents.

```
::: {.callout-note collapse=true}
#### Please look at me
Secret secret notes.
:::
```

ℹ Please look at me

Secret secret notes.

Other types of callout blocks include:

- Warning

```
::: {.callout-warning}
These are warnings.
:::
```

⚠ Warning

These are warnings.

- Important

```
::: {.callout-important}
This is something important.
:::
```

Important

This is something important.

- Tip

```
::: {.callout-tip}
Here is a handy tip.
:::
```

Tip

Here is a handy tip.

- And, a caution

```
::: {.callout-caution}
Here is a caution about something.
:::
```

Caution

Here is a caution about something.

3.3.4 References

If you want to add proper references instead of just hyperlinks, you need a .bib file from a reference manager.

The .bib file should be in the include folder (unless you point it somewhere else) and you can specify it in the `_quarto.yml` file through the bibliography entry:

```
bibliography: include/references.bib
```

How do I download and edit a .bib file?

I use Zotero as a reference manager and its super easy to download a .bib file for a project you are working on. Downloading one entry is a little annoying as you need to export it as BibTex and copy from the file it produces, but if you create a folder to store everything for the book, you can just export the folder each time you add new entries (`right click >> export collection >> BibTex format and OK`).
If you do have the single entry, you can open the .bib file within RStudio and copy the entry in. It will look something like this:

```
@article{bartlett_power_2022,
    title = {Power to the {People}: {A} {Beginner}'s {Tutorial} to {Power} {Analysis} using
    volume = {6}
    ...}
```

which stores all the information for the `.csl` to pull out and cite/reference as needed.

To cite, you need the code at the start of the bib entry. For example, `@bartlett_power_2022` produces Bartlett & Charles (2022) and the full reference will be added to the references chapter.

Depending on the citation style you want, there are different codes, such as adding it in parentheses `[@bartlett_power_2022]` (Bartlett & Charles, 2022). For a full list of options, you can check out the Quarto citation guide.

By default, the book template has APA style for referencing, but if you need a different referencing style, you can add and specify a different .csl file within `_quarto.yml`.

```
csl: include/apa.csl
```

How do I specify a .csl file?

`.csl` stands for citation style language and you can download one from the Zotero style repository. For example, you could search for vancouver, click on the link, and it will download a new .csl file you can add to your repository within `include/`.

3.3.5 `webexercises` interactive questions

The book template automatically includes the `webexercises` package which can add interactive questions for self-tests. This is great for students checking their understanding through MCQs or adding easy to check answers like numbers.

3.3.5.1 MCQs

You can add questions through inline code, or by first specifying them in an R code block if it makes it easier to edit longer text.

For example, this workshop is:

- (A) Life changing
- (B) Boring
- (C) Mediocre
- (D) OK

```
`r longmcq(c(answer = "Life changing", "Boring", "Mediocre", "OK"))`
```

3.3.5.2 Single answer

You can ask simple single answers that are easy to evaluate:

- On a scale of 1 (very dissatisfied) - 7 (very satisfied), how pleased are you with this workshop? __

```
`r fitb(7)`
```

3.3.5.3 True or false

If you want an even simpler response, you can ask true or false.

- After the workshop, I am going to make my own book: TRUE / FALSE

```
`r torf(TRUE)`
```

3.3.6 Embedding files to download

Using a similar format to creating hyperlinks, you can embed files for people to download and use in the chapter. This can be really useful for student activities as you can give them a data set to follow along to your tutorials with.

First, you need to add a file within your book directory. If you have loads of data or files across your book, you might want a separate folder (I call mine `data` or `supporting`), but I have put a simple .csv in the `include/` folder.

If you click on .csv file, it will download to your browser or people might need to right click and "save link as". It follows the same format as hyperlinks:

```
click on [.csv file](include/test_data.csv)
```

3.3.7 Adding a glossary

Another cool feature that Lisa created is adding a glossary of terms. `glossary` is its own R package, but by default `booktem` includes it. You have two options, you can either add your own definitions as you go along, or if you are teaching data skills, you can use the PsyTeachR glossary.

! Important

You still need to add definitions for anything that is not included in the PsyTeachR glossary. If you try and render and the item does not exist, you will get an error and you will need to manually add the definition within the inline code.

There are two main components to creating a glossary. First, you need to add glossary items as you work through your chapter using inline code. For example, I might want to define what a glossaryAn alphabetical list of words with explanations. is:

```
`r glossary("glossary", def = "An alphabetical list of words with explanations.")`
```

If you hover or click on the text, you will see the definition appear. There are different settings for this, so make sure you check the glossary documentation.

At the end of each chapter, you can then include a glossary table which shows all the words you used in the chapter. Just make sure you add `echo=FALSE` to the code chunk, so that the function does not appear.

~~~
```{r}
glossary_table()
```
~~~

This produces:

term	definition
glossary	An alphabetical list of words with explanations.

The behaviour of glossary table and whether you use all your own definitions or point to the PsyTeachR glossary is controlled by some R code in the `booktem` files. It will be easier to point out in the workshop, but you are looking for `R/my_setup.R`.

You can add definitions as you go along with inline code, or you can create and edit a .yml file for your terms if you would prefer to edit that way. See the `glossary` documentation for more information.

References

Bartlett, J., & Charles, S. (2022). Power to the People: A Beginner's Tutorial to Power Analysis using jamovi. *Meta-Psychology*, *6*. https://doi.org/10.15626/MP.2021.3078