Dissemination using Quarto and Github Pages

James Bartlett

2025-03-28

Table of contents

Overview

This book outlines how you can combine Quarto - a new version of R Markdown - with Github Pages to create an online profile and disseminate your work. You will learn about writing online books, sharing reproducible presentations, and creating websites and blogs.

There are alternative options to all of these outputs, but once you are comfortable with the workflow of Quarto and github, you will have a flexible toolkit to manage and share all your outputs in one place. Once you render your book/presentation/website and push any changes to github, you get a URL to share with anyone.

For early career researchers, a website or blog is important for people finding out about you and your work. Before you have the opportunity to publish papers, you will probably have more conference presentations. The ability to share your presentation with a URL will provide more reach than your title and abstract in a programme. For more experienced academics, you might run courses where you want to develop your own materials for your students or share your materials with your peers. An online book is great for impact and sharing your work as wide as possible.

The PDF version of this book was last updated on **28th March 2025**.

Intended Learning Outcomes (ILOs)

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

- Use the basic functions of git and github to commit changes and host your materials using github pages.
- Create an online book using Quarto or the PsyTeachR `booktem` R package in RStudio.
- Create reproducible presentations using Quarto to combine text, images, and code.
- Create a simple website and/or blog using Quarto to share your profile and outputs.
- 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 focus on creating in the workshop.

1.1 The workflow of Quarto and github pages

For an overview to show why you need the following the pieces of software, you will follow this workflow to create and edit your materials. Figure 1.1 shows the Quarto/github pages workflow you will follow.

1.1.1 Creating your output

1. Create a new project to produce the skeleton of your book/presentation/website.
2. Edit the basic details of your book, like the title, author(s), and description.
3. Create a new github repository for your output.
4. Publish the initial repository so the code is available on github.
5. Activate github pages to render the book online via a URL.

1.1.2 Editing your output

1. Open the output .Rproj file to start working on the materials in RStudio.
2. Edit or add chapter files in RStudio, specifying their order in the `_quarto.yml` file.
3. Render individual chapter/section 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.

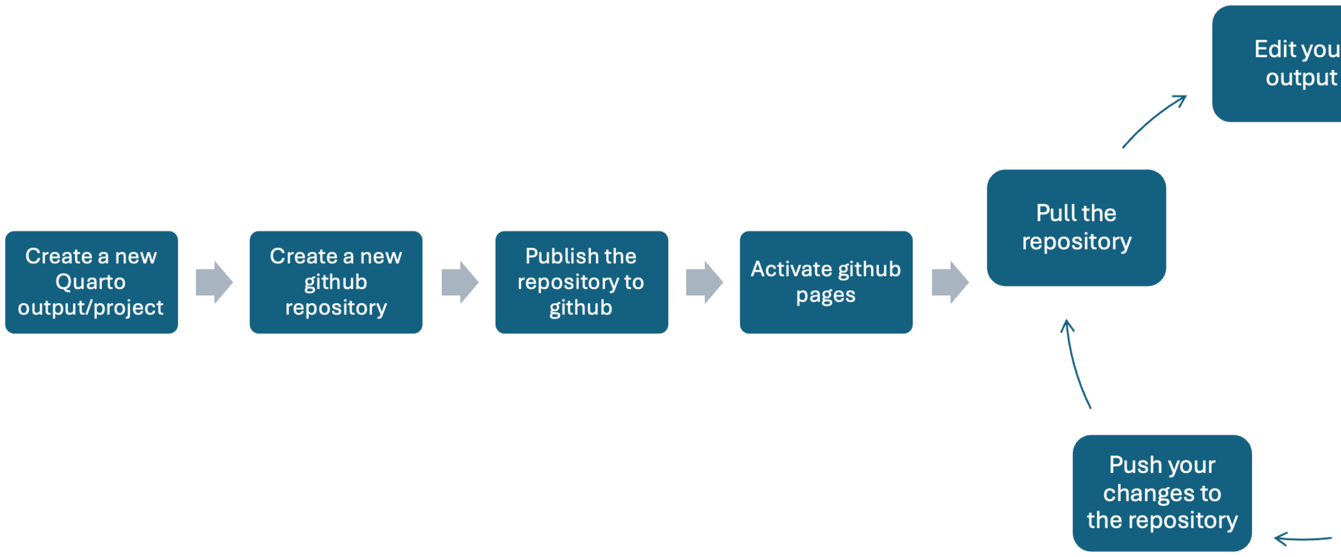
Create a new Quarto output/project
Create a new github repository
Publish the repository to github
Activate github pages
Pull the repository
Edit you output
Push your changes to the repository

Figure 1.1: A flowchart showing the creation and editing process behind a Quarto output and sharing it using github pages.

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 useful or the RSetGo guide we prepared for students in the School of Psychology and Neuroscience.

If you are a more experienced R/RStudio user, just make sure you update your version of RStudio as Posit are rapidly developing Quarto. I am currently using R version 4.4.1 (2024-06-14) and RStudio 2024.09.1+394, but the more recent the better.

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.

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 (optional) Install the `booktem` R package

For one version of a Quarto book, we have a specialised PsyTeachR book template for the School of Psychology and Neuroscience which you can use, but it is contained within a package hosted on Prof. Lisa DeBruine'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.

Warning

Unfortunately, this course is only available to students and staff within the University of Glasgow. If you are not, please use the github guidance above or another resources for learning about git and github.

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 materials 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 your outputs in the workshop or follow along to the rest of the book here!

Part I

Writing Books

2 Quarto books

In this chapter, you will learn how to create and edit the standard version of a Quarto book. This differs to the specific PsyTeachR template we cover in the next chapter. If you do not want additional features built in like interactive exercises and a glossary function, you may want a more streamlined version. The editing and rendering process is exactly the same, it is just the creation process and additional features which differ.

Intended learning outcomes

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

- Create and edit a standard Quarto book using RStudio.
- Publish your book using github pages.

2.1 Example book and Quarto documentation

For an example of the standard Quarto version, you can explore this very book on Dissemination using Quarto and Github Pages! If there is a feature you like, you can see the source code on github to adapt to your needs.

Your main source of information for this output is the Creating a Book section of the Quarto guide.

2.2 Creating a Quarto book

If you have not followed the preparation instructions yet, you need R/RStudio and git/github desktop installed on your computer. I will be demonstrating how to use github desktop rather than the command line, as git could easily be it's own workshop.

The first step is thinking about where you want your book folder stored on your computer for 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).

⚠ 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 decided where your book will live, open RStudio and click `File > New Project... > New Directory > Quarto book`. This will be the process we follow for all the Quarto outputs and you see there are options for a range of documents like websites and blogs.

You will see a new window asking for you to specify:

- "Directory name:" - the name of the folder it will create, so keep it short and without spaces.
- "Create a project in the subdirectory of:" - click browse to navigate to the folder you want your book to live in. Creating the book will create a new folder within this directory, so you do not need to create a folder for the book yourself.

Click "Create project" and after a couple of seconds, it will open a new project window with the template in.

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, author(s), and the licence. In Quarto books, this is also how you control the order of chapters.

ℹ 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 in the workshop key features such as: project, book, bibliography, csl, and format.

For presenting your book using github pages, edit the `project` from the standard first two lines:

```
project:
  type: book
```

To add a new third line:

```
project:
  type: book
  output-dir: docs
```

Be careful, indentation is important in .yml files. We do this as by default, it will create a folder called `_book` where the rendered .html files will live. For github pages though, it looks for a folder called `docs`, so this will streamline things later.

❗ Add a license for your work

One important consideration is telling people how they can use your work. In the .yml, you can specify a license (see the License section of the documentation) and make this super clear in the footer of your output. I like to specify this using the following settings under `website`:

```
book:
  license: "CC BY-SA"
  page-footer: CC-BY-SA-4.0 (2025) James Bartlett
```

I usually apply a CC-BY-SA license from Creative Commons as one of the most permissive licenses for reusing work, but they must provide attribution. You can look at the Creative Commons site for different license options.

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. Chapter files

By default, you get an example index.qmd and a few other example chapters like intro.qmd, summary.qmd, and references.qmd. The example chapters demonstrate some features of Quarto 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

Depending on what you want to include in your book and how complicated it becomes, you might want to separate chapters into different sections. The indentation can be frustrating, but you can add parts to the `book:` section of the .yml like:

```
book:
  chapters:
    - index.qmd
    - workshop_prep.qmd
    - part: "Writing Books"
      chapters:
        - quarto_books.qmd
        - psyteachr_template.qmd
```

4. references.bib

In the .yml, you can specify a BibTeX file (.bib) to add formatted citations and references. The template comes with an example for how a citation and its reference will look. A .bib file stored information for referencing like authors, the journal, the DOI etc. The Quarto file will take this information and present it as a citation. The Quarto features chapter explains how to use a .bib file to format citations and specify a specific citation style.

5. 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.

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 specify that folder when you created a project sub directory 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. When you are used to the process, you can work out which workflow suits you better.

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. It makes sense to call this the same as your book folder wherever possible.
- 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.

Click on Settings for the next 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. That is not something we are covering in this workshop or materials though.

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, keep 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. Likewise, editing the .qmd files and committing the changes to github will not change anything without first rendering your chapter/book. You edit the .qmd files but rendering creates the .html files that display as a website. 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 final chapter on Useful Quarto and booktem features for inspiration / example code for the kind of features you can use.

3 PsyTeachR book template

In this chapter, you will see how to use the PsyTeachR book template developed by Prof. Lisa DeBruine. This template is what we use for all the data skills resources in the School of Psychology and Neuroscience. This has some nice features compared to the standard Quarto book template, such as supporting webexercises for interactive questions and including a glossary.

Intended learning outcomes

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

- Create and edit a Quarto book using the PsyTeachR template.
- Publish your book using github pages.

3.1 Example books and Quarto documentation

For a selection of examples of the PsyTeachR book template, you can explore:

- The Fundamentals of Quantitative Analysis book designed for MSc Psychology conversion students. If there is a feature you like, you can see the source code on github to adapt to your needs.
- The Data Skills for Reproducible Research book designed for MSc Research Methods of Psychological Science and MSc Brain Sciences students. If there is a feature you like, you can see the source code on github to adapt to your needs.

You can see the whole PsyTeachR suite of resources on our website.

Your main source of information for this output is the Creating a Book section of the Quarto guide.

3.2 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, as git could easily be it's own workshop.

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!

3.3 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 Quarto books, 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.

> **Add a license for your work**
>
> One important consideration is telling people how they can use your work. In the .yml, you can specify a license (see the License section of the documentation) and make this super clear in the footer of your output. I like to specify this using the following settings under `website`:
>
> ```yaml
> book:
>   license: "CC BY-SA"
>   page-footer: CC-BY-SA-4.0 (2025) James Bartlett
> ```
>
> I usually apply a CC-BY-SA license from Creative Commons as one of the most permissive licenses for reusing work, but they must provide attribution. You can look at the Creative Commons site for different license options.

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

```r
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.

3.4 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.

3.5 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.

3.5.1 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.

3.6 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.

3.7 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 extra panache!

Part II

Sharing Presentations

4 Reproducible presentations

In this chapter, you will learn how to create presentations that you can share with a URL. Your presentation might be purely conceptual where you use words and images, or you might be presenting your research and you want to share reproducible analyses and figures. This is great for exposure as it's better than your name and abstract in a programme, and it's much easier to share a URL link to your presentation which people can repost compared to a PDF.

4.1 Example presentations and Quarto documentation

For an example of creating Quarto presentations, you can explore a papaja demonstration I co-presented for a previous data upskilling workshop. If there is a feature you like, you can see the source code on github to adapt to your needs.

Prof. Lisa DeBruine has a great website where she shares all her talks and you can see the underlying code in her github repository. Each folder contains all the files for one talk, so you can see the variation in how to achieve different features. **Note**: Lisa also adds some custom css features (styling for html pages) which we will not really be getting into in this workshop.

Your main source of information for this output is the Revealjs section of the Quarto guide.

4.2 Creating a Quarto presentation

The first step is thinking about where you want your presentation 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).

⚠ 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 presentation, 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.

For presentations, there are a couple of options for your starting point. Compared to books and websites in other chapters, there is not a standalone option for a presentation project to create a folder with the files inside. So, you have one of two options:

1. You just create a presentation by clicking `File > New File > Quarto Presentation...`. Enter a title, author, and keep the default Reveal JS option. Click Create and you will have a blank .Qmd called Untitled1.qmd. You can then create a new folder for the presentation to live in as you save Untitled1.qmd with a proper file name.
2. You first create an R Project by clicking `File > New Project` to create a new folder/directory. This creates the folder and an R Project (.RProj) file to help organise things, then you complete step 1 to add the Quarto presentation.

If you are not including any data, it will not make much difference which way you create the presentation. However, if you want to include an element of data analysis, it can be useful to use an R Project to help organise your working directory.

Once you have created your presentation .qmd file, you can Render it and see a bland looking .html page with your title and author. Congratulations, you have a presentation skeleton to work with!

4.3 Editing your Quarto presentation

Weirdly, the default presentation file does not actually specify it should be a presentation. So, your initial render was a very bland looking webpage. Edit the YAML section at the start of the file from the default title and author to add a few lines about the formatting:

```yaml
---
title: "Example Presentation"
author: "James Bartlett"
format:
  revealjs:
      embed-resources: true
---
```

If you click Render again, *now* it will look and behave as a presentation.

The `embed-resources` part makes the .html files slightly larger and takes longer to render but it means you can open them independently. Without this, there will be a folder in your directory called "yourtitle_files" which it needs to add the formatting. If you just shared the .html file without the files folder, it would look weird. If you set `embed-resources` to true, all the formatting will be contained within the .html file and you can share it as a single file.

4.3.1 Adding sections and slides

The YAML section will control the title page but to add individual slides, you use level 1 and level 2 headers.

A level 1 header (one hash #) will create a section and a level 2 header (two hashes ##) will create a regular title. You can just use level 2 headers to create basic slides with a title, but periodic level 1 headers can be useful for organisation to create sections which are easier to navigate in the contents menu.

```
# This is a section

## This is a slide

Add your content here.
```

Try this

Try and switch between combinations of level 1 and level 2 headers to see how it looks when you render your presentation.

Alternatively, if you want to create a slide without a title, you can use three dashes to start a new slide:

```
# This is a section

## This is a slide

Add your content here.

---

This is a slide without a title.
```

4.3.2 Adding content

Once you have added your slides, you can add your content to them. This can include simple text, bullet points, code, and images. For the following demonstrations, if you want to play around with text to see how it looks, try using the *Lorem ipsum* typesetting passages:

> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis

nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3.2.1 Smaller text

It will take some practice and tweaking to learn how much you can fit on slides before you need to create a new slide. It's probably a good hint to question how much content you are including if there is too much to fit, but you can specify smaller text with a tag on the header.

```
## This is a slide {.smaller}

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
```

Try this

Try and compare the full typesetting extract with and without the `{.smaller}` tag.

4.3.2.2 Multiple columns of content

Like the smaller text tag, there is a class of tag called `.columns` which you can use to specify columns and their width on the slide. For example, if we wanted two equal columns both containing text, you could add:

```
:::: {.columns}

::: {.column width="50%"}
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
:::

::: {.column width="50%"}
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
:::

::::
```

Within each column, you can include any regular content like text, bullet points, or images.

4.3.2.3 Individual images

If you do not want to run code to create an image or visualisation for your presentation, you can save an image to your

If you want a test image, you can download this lovely picture of a duck. Save the image in your presentation folder (or a subfolder called images if there are lots you want to keep tidy) and add one of the following two options:

```
![](Duck.png)
```

or

~~~
```{r}
knitr::include_graphics("Duck.png")
```
~~~

You can see a longer overview of image options in the Quarto features chapter, but knitr is a package RStudio uses for dynamic documents like RMarkdown and Quarto. It includes a handy function called `include_graphics()` which takes a file path as it's first argument.

If there are several images you want to include on a slide and you want to carefully size or position them, there are some advanced features in the reveal documentation.

4.3.2.4 Code and output

As we are using Quarto within RStudio, you can add any code and output that would comfortable fit on a slide. This is a great feature if you are teaching about coding as you do not need to worry about adding screenshots of code and output. If you are presenting results from your research, it also means you can include reproducible analyses or visualisation.

By default, you will only see the output of code if you try and render a slide with this:

~~~
```{r}
summary(mtcars)
```
~~~

But you can set echo=TRUE within the code chunk after the `r` to also show the underlying code if that is relevant to what you are presenting.

4.3.3 Changing the theme

The slides use a default theme without any further edits, but Quarto presentations come with a few built-in themes you can use. They are all pretty minimalist but you can explore which suit your needs best.

To edit the theme, you need to specify the theme in the YAML section at the start of the file, but be careful of indentation:

```
---
title: "Example Presentation"
author: "James Bartlett"
format:
  revealjs:
    embed-resources: true
    theme: dark
---
```

> **Try this**
>
> There are 11 built-in themes, so try and change it to one of the other themes such as "simple" or "moon" to see what they look like.

> **How can I make more specific edits?**
>
> The one downside to using Quarto presentations is that it is more difficult to edit basic features like the font than if you were using MS PowerPoint. If you do want a specific look to your presentation, you will need to play around with adding a .css file for styling. You can learn more about this on the Reveal Themes page.

These are the key components to make your presentation. 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 presentation accessible online via github pages.

4.4 Creating a github repository

Once you have a working barebones version of your presentation ready to go, it's time to associate your presentation 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 add your new presentation. However, we started by creating the presentation 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 presentation folder. You want the path to be the main folder your presentation 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.

4.5 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 share your presentation, 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 and it is the key section we need to get your presentation online.

4.5.1 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 `/(root)`. This is slightly different to the book and website process in the other chapters. We do not need to point GitHub Pages to the `docs/` folder as we do not have one for this output.

Instead, the rendered .html file is in the main directory. This means `/(root)` will be looking for something within your top-level (or root) directory.

When you press Save, this will start building your presentation. 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! Your presentation is now available to view and share online!

Add shortcuts to your presentation

Once the site is live, I recommend adding the URL to the about page. 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 presentation on the Code tab for easy access from here.

Why is it showing a different page as my site?

As the directory is set to the root directory, Github Pages will look for the first .html file in your directory. If that is not your presentation, it will show a different file. In some of my repositories, I have a rendered README file (explaining information about the repository and usage instructions) which shows as the "site". However, you can still get a link to your presentation by adding the full extension such as https://bartlettje.github.io/papaja_demo/papaja_slides.html.

4.6 Commiting and pushing changes

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

1. Open the RProj file (if you are using a project file) and edit your presentation in RStudio.
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 presentation, 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 presentation, I will end on a couple of warnings and tips.

Caution

Remember just committing changes will do nothing to your github repository and presentation link. You need to push those committed changes for them to be available on github and used to build the presentation 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.

4.7 Start working on your own presentation!

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

See the final chapter on Quarto features for inspiration / example code for the kind of features you can use.

Part III

Creating Websites

5 Websites and blogs

In this chapter, you will learn how to create professional websites and/or blogs using Quarto. For academics, an online presence is critical but often overlooked. From early career researchers to experienced academics, you will benefit from people being able find out more about you, such as your publications, talks, and contact information. You might think the technical barrier is too high to maintain your own website, but there are plenty of tools out there to make it easy. The combination of Quarto and Github Pages is a free example of this which you can maintain yourself alongside the other outputs I covered in this book. This chapter rolls websites and blogs into one as the general principles are the same, where blogs are just a specialised form of website to present individual posts on the index page.

5.1 Example websites and Quarto documentation

For an example of a Quarto website, you can explore the UK Conference on Teaching Statistics website which I helped to create and maintain. If there is a feature you like, you can see the source code on github to adapt to your needs.

For other examples, Solomon Kurz - a psychology researcher who writes about programming / stats - posted on BlueSky about academic websites. The thread has over 40 replies, so you can explore how other people have used Quarto or similar tools:

Who has an academic website they're proud of?

I'm especially interested in websites by grad students, and in websites built with Quarto. I'll be offering a local workshop on building a website with Quarto, and I'm keen to give my audience examples. #rstats

— Solomon Kurz ((**solomonkurz.bsky.social?**)) 19 March 2025 at 14:27

Your main source of information for this output is the Creating a Website and Creating a Blog sections of the Quarto guide.

5.2 Creating a Quarto website or blog

If you have not followed the preparation instructions yet, you need R/RStudio and git/github desktop installed on your computer. I will be demonstrating how to use github desktop rather than the command line, as git could easily be it's own workshop.

The first step is thinking about where you want your website folder stored on your computer for 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).

⚠ 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 website, 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 decided where your website will live, open RStudio and click `File > New Project... > New Directory > Quarto Website` for a website, or `File > New Project... > New Directory > Quarto Blog` for a blog.

Regardless of whether you want to create a website or a blog, you will see a new window asking for you to specify:

- "Directory name:" - the name of the folder it will create, so keep it short and without spaces.
- "Create a project in the subdirectory of:" - click browse to navigate to the folder you want your website/blog to live in. Creating the website/blog will create a new folder within this directory, so you do not need to create a folder yourself.

Click "Create project" and after a couple of seconds, it will open a new project window with the template in. Congratulations, you have a website/blog skeleton to work with!

Although there is a lot of overlap between the two outputs, I have split the tour into a website section and a blog section as there are a couple of key distinctions to highlight. So, you can navigate to the one most relevant to you at this point.

5.3 Tour of the Quarto website template

1. `_quarto.yml`

We will start by exploring the `_quarto.yml` file where you can edit all the details for your website, like the title, licence, theme, and any styling. In Quarto websites, this is also how you control the number and order of pages in the navigation bar.

> 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 in the workshop key features such as: title, navbar, and format.

> Adding new pages to your website
>
> By default, your website will have the main index and an about page in the navigation bar. To add more pages, simply create new .qmd files in the directory (including at least a title in the YAML) and specify where you want them in the .yml file.

For presenting your website using github pages, edit the `project` from the standard first two lines:

```yaml
project:
  type: website
```

To add a new third line:

```yaml
project:
  type: website
  output-dir: docs
```

Be careful, indentation is important in .yml files. We do this as by default, it will create a folder called `_site` where the rendered .html files will live. For Github Pages though, it looks for a folder called `docs`, so this will streamline things later.

> Adding Google analytics
>
> If you want to track how many people view your website and access other kinds of analytics, you can add a Google analytics key into the .yml options.
> You would add it into the `website:` section like:
>
> ```yaml
> website:
>   google-analytics: "UA-XXXXXXXX"
> ```

There are additional options like turning on a cookies warning if you are using this feature to tell people you are tracking user activity on your website, so I recommend reading the Quarto documentation for these features.

Add a license for your work

One important consideration is telling people how they can use your work. In the .yml, you can specify a license (see the License section of the documentation) and make this super clear in the footer of your output. I like to specify this using the following settings under `website`:

```yaml
website:
  license: "CC BY-SA"
  page-footer: CC-BY-SA-4.0 (2025) James Bartlett
```

I usually apply a CC-BY-SA license from Creative Commons as one of the most permissive licenses for reusing work, but they must provide attribution. You can look at the Creative Commons site for different license options.

2. index.qmd

The index file will be the opening page to your website, so imagine it as a kind of home page. You can edit the title which may or may not be the same as your overall website title. Everything below the YAML section will be rendered as a regular .qmd file, so think about what you want to present to people.

3. About.qmd

By default, you get an about page which is meant to describe you or your website. You might include information like your social media profiles and how best to contact you.

Add about page formatting

As you will see in the blog section, the about page in a Quarto blog can include several default templates. Weirdly, this is not activated in the website template, but you can use the same code in the YAML section.
The standard about page will look like:

```yaml
---
title: "About"
---
```

But you can specify `about` as a section to add this formatting and preset link options such as github:

```
---
title: "About"
about:
  template: jolla
  links:
    - icon: github
      text: Github
      href: https://github.com
---
```

4. Custom css styling

The only other file remaining in the template is styles.css. In this workshop, we are not exploring css (cascading style sheets), but this is essentially how you can customise the look of html pages. If you are not totally happy with the preset themes, you can explore how to edit the css styling on the CSS Styles Quarto documentation.

5. Rendering files

Once you have finished editing each page and 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, you can open it up in the browser and keep checking as you make edits.

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.

These are the key components to make your website. 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 website accessible online via github pages. Skip ahead to the creating a github repository section if you do not want to read about blogs.

5.4 Tour of the Quarto blog template

1. `_quarto.yml`

We will start by exploring the `_quarto.yml` file where you can edit all the details for your blog, like the title, author(s), and the licence. In Quarto websites, this is also how you control the number and order of pages in the navigation bar.

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 in the workshop key features such as: title, navbar, and format.

Adding new pages to your website

By default, your website will have the main index and an about page in the navigation bar. To add more pages, simply create new .qmd files in the directory (including at least a title in the YAML) and specify where you want them in the .yml file.

For presenting your blog using github pages, edit the `project` from the standard first two lines:

```
project:
  type: website
```

To add a new third line:

```
project:
  type: website
  output-dir: docs
```

Be careful, indentation is important in .yml files. We do this as by default, it will create a folder called `_site` where the rendered .html files will live. For github pages though, it looks for a folder called `docs`, so this will streamline things later.

Adding Google analytics

If you want to track how many people view your website and access other kinds of analytics, you can add a Google analytics key into the .yml options.
You would add it into the `website:` section like:

```
website:
  google-analytics: "UA-XXXXXXXX"
```

There are additional options like turning on a cookies warning if you are using this feature to tell people you are tracking user activity on your website, so I recommend reading the Quarto documentation for these features.

❗ Add a license for your work

One important consideration is telling people how they can use your work. In the .yml, you can specify a license (see the License section of the documentation) and make this super clear in the footer of your output. I like to specify this using the following settings under `website`:

```
website:
  license: "CC BY-SA"
  page-footer: CC-BY-SA-4.0 (2025) James Bartlett
```

I usually apply a CC-BY-SA license from Creative Commons as one of the most permissive licenses for reusing work, but they must provide attribution. You can look at the Creative Commons site for different license options.

2. index.qmd

For a blog, there is not too much to edit here. An index file will be the opening page to your blog, but it's main function here is to display a preview of all your blog posts. Below the YAML which controls the title and listings, you can add text, code, and/or images like a normal .qmd file, but following this will be a list of blog posts in descending date order.

3. About.qmd

By default, you get an about page which is meant to describe you and provide key contact information, such as your social media profiles.

About pages have a special template with a few presets which you can learn about in the Quarto documentation. The default will be "jolla", but you can change it to another template that suits your needs.

Otherwise, anything below the initial YAML section will be formatted like a regular .qmd file for text, code, images etc.

4. The `posts` folder

By default, you get two example posts to show how they should be structured. In index.qmd, part of the YAML specified `contents: posts`. This means it is using the contents of the `posts` folder to generate the blog posts.

If you click on the `posts` folder, you will see each post lives in its own folder. To create a new post, you would create a new folder with a short but informative name. Within that folder, there is another index.qmd file and any supporting files such as images. Each post needs its own index file and the YAML at the top controls the title, author, date, and any categories. Editing these will customise the look of the post in the preview on your home page and the top of your blog post when you open it.

5. Custom css styling

The only other file remaining in the template is styles.css. In this workshop, we are not exploring css (cascading style sheets), but this is essentially how you can customise the look of html pages. If you are not totally happy with the preset themes, you can explore how to edit the css styling on the CSS Styles Quarto documentation.

6. Rendering files

Once you have finished editing each page or post and 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, you can open it up in the browser and keep checking as you make edits.

> 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.

These are the key components to make your blog. 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 blog accessible online via github pages.

5.5 Creating a github repository

Once you have a working barebones version of your website ready to go, it's time to associate your website 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 create a new R Project for your output within this folder. However, we started by creating the website 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.

Your github username as your website URL

Unless you want to buy a custom URL, the typical format for the link will be yourusername + github.io/ + your repository name. For a website or blog, you can take advantage of a feature where every profile receives one site through Github Pages. You can create a repository called yourusername + github.io (e.g., bartlettje.github.io) and this will be the URL of your site, rather than another repo appended to the end. You only get one per github account though, so it's best to use it on a website and blog where you're likely to only have one.

- 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 website folder. You want the path to be the main folder your website 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.

5.6 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 website, 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 website 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 website available online.

5.6.1 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 website files, so you are pointing Github Pages here as the source for your website.

When you press Save, this will start building your website. 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 website 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 website on the Code tab for easy access from here.

Customise the URL

By default, the URL to your website 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. See the github documentation for further details.

Warning

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

5.7 Commiting and pushing changes

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

1. Open .RProj and edit your website 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 website, keep 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 website, I will end on a couple of warnings and tips.

Caution

Remember just committing changes will do nothing to your github repository and website link. You need to push those committed changes for them to be available on github and used to build the website in github pages. Likewise, you can edit the code in your .qmd file but not render those edits into the formatted html files. 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.

5.8 Start working on your own website!

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

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

Part IV

General Quarto Resources

6 Useful Quarto and booktem features

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

6.1 Quarto guide

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

6.2 Markdown

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

6.2.1 Headers

The table of contents in Quarto outputs 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
```

6.2.2 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

6.2.3 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="_b
```

6.2.4 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 preparation chapter, you first need to add a tag on the chapter heading:

```
# Preparation before the workshop {#workshop_prep}
```

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

```
...back to the [workshop preparation](#workshop_prep) chapter
```

6.2.5 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)
```

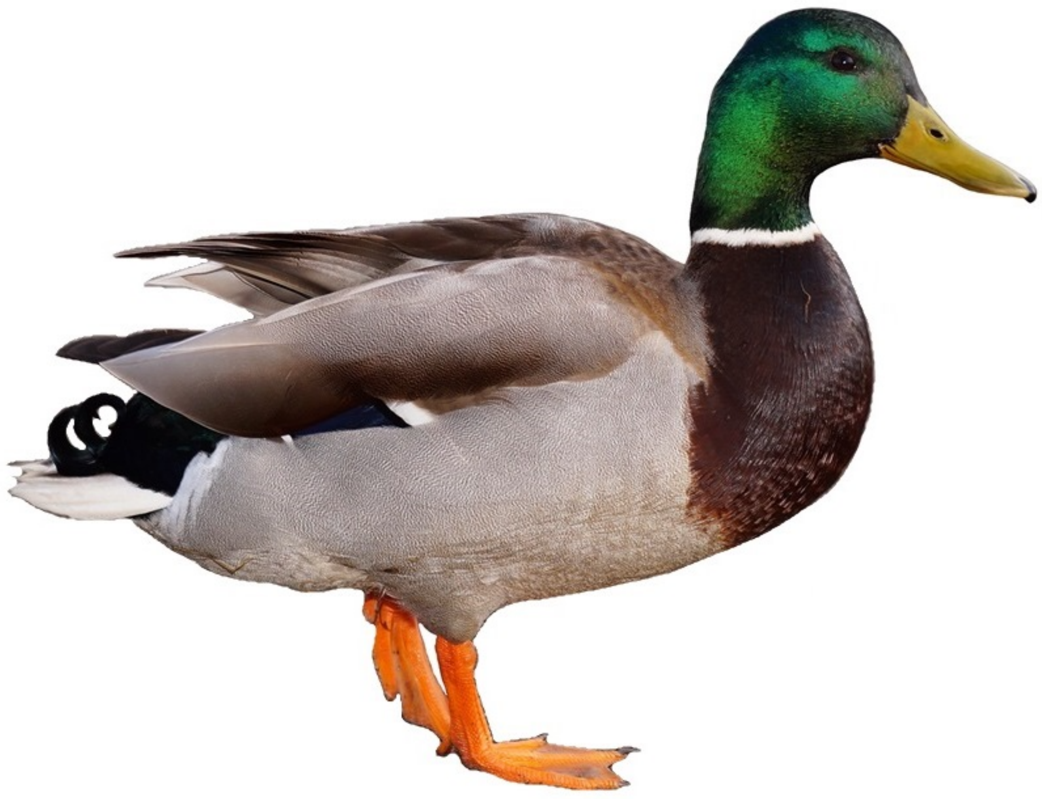

Figure 6.1: This is a duck. If Helena is reading this, yes it has a creative commons licence.

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.

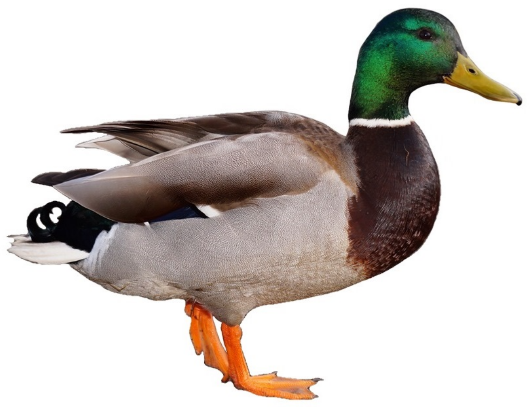
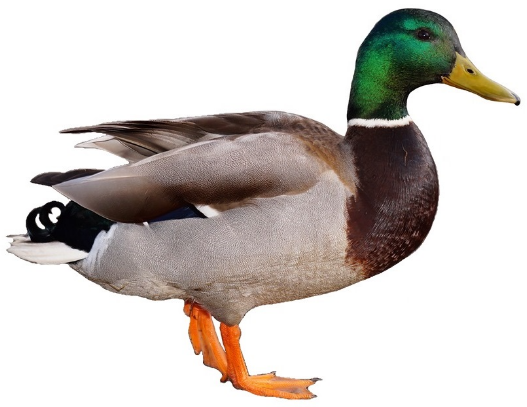

Figure 6.2: Duck 1 (left) and Duck 2 (right).

Figure 6.2 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 6.3.

~~~
```{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")
```
~~~

6.3 Code chunks

If you are making a book or website 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)
```
~~~

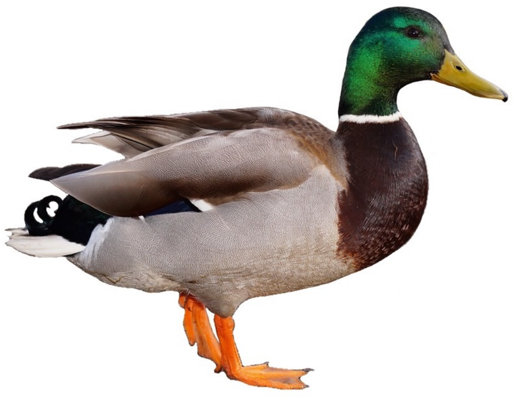

Figure 6.3: This is a longer caption about our beloved duck.

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

```
[1] 11.533641 10.418023  7.382991 13.788234  9.983890
```

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] 11.471680 12.992746 11.478046  9.402364 10.033443
```

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}`:

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

6.4 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.

6.5 Embedding posts or videos

When you write in Markdown language, it renders to a .html page. This means you can add html code directly into your .qmd files and it will render in the final .html page. If you want to embed posts from social media, you can copy the embed post code from something like BlueSky (click the three dots … below a post and select embed post):

```
<blockquote class="bluesky-embed" data-bluesky-uri="at://did:plc:lrgnqp3xjombwh5sairhzou3/ap

This is our data skills book for MSc conversion students, so they must rapidly get up to spe
```

Which will produce the following:

A little delayed but Wil Toivo and I finally have a complete rewrite of our Fundamentals of Quantitative Analysis book psyteachr.github.io/quant-fun-v3/

This is our data skills book for MSc conversion students, so they must rapidly get up to speed and complete chapters 1-11 in semester 1.[image or embed]

— James Bartlett ((**bartlettje.bsky.social?**)) 11 November 2024 at 09:55

This also works for embedding something like a YouTube video where you can click `Share > Embed` from under a video, and paste the code into your .qmd file.

6.6 Equations using LaTeX

For books, one useful feature might be the support of LaTeX equations if mathematics or statistics is a key component of your material. For example, you can add simple or complex equations within two dollar symbols:

```
$$
E = MC^2
$$
```

And it will produce the following:

$$E = MC^2$$

6.7 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/`.

6.8 `webexercises` interactive questions

The PsyTeachR `booktem` 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.

6.8.1 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"))`
```

6.8.2 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)`
```

6.8.3 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)`
```

6.9 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)
```

6.10 Adding a glossary

Another cool feature that Lisa DeBruine 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