

Teaching Materials Support

Details and Setup of the Quarto site for teaching statistics

Matthew Ivory

2024-04-02

Table of contents

1 Teaching Materials Support	3
1.1 Executive Summary	3
1.2 About	3
1.3 Other key things	4
2 GitHub integration and workflow	5
2.1 Collaborators	5
2.2 Pushing changes to the site	7
2.2.1 via terminal	7
2.2.2 via RStudio	8
3 Creating and Managing Worksheets	9
4 Features and Content	10
4.1 Images	10
4.2 Callouts	10
4.3 Includes	11
4.4 Embedding Panopto	11
5 The previous content repository	13
6 Technical	14
6.1 Folders with underscores	14
6.2 Search Bar Placeholder	14
6.3 Freeze - collaboration caching	14

1 Teaching Materials Support

1.1 Executive Summary

- The lu-psy-r.github.io site hosts the teaching materials used in conjunction with the R server and enables a more interactive/integrated experience for students between R and statistics.
- The site pages are created using Quarto and are published to GitHub pages to allow for collaborative working and easy publication processes.
- Each year (1XX, 2XX, and 4XX) are stored in separate repositories which ensures content is siloed for confidence that the site will not go down altogether post-edits.
- Content is simply rendered and then pushed to a GitHub repository, so no significant new processes are required to post content.
- Core content required for site operation is stored separately (either in git-level folders or in repo-level folders) that allows for greater focus on the teaching content and not site management.
- Content can be created in advance and kept hidden from students until needed.
- Lecture recordings, Quarto presentations, and fill-in-the-blanks sections can be used to increase interactivity and engagement.

1.2 About

The R teaching materials are hosted via lu-psy-r.github.io. The pages are written in Quarto, which are then rendered and then hosted in a GitHub repository. Each “year” (first, second, and MSc) are separate repositories, which is intended to minimise the potential for “cross-contamination” and breaking of the website. As much content as possible has been abstracted away from these repositories (or hidden at a minimum) to try and make the contributor experience straightforward.

Each repository is similar in structure but equally, slightly different. Each contains some basic items:

- a `/_core` folder. Please don't touch this, this contains files that cannot be abstracted out of the repository - without this, the site will break.
- a `_quarto.yml` file. This controls a bunch of site functionality, with much of being consistent across each project. This controls the sidebar look as something unique in each folder. Depending on how each repository is being managed, the sidebars may need tweaking slightly. For two examples, year 1 is a straightforward setup, and so it just auto lists the contents of PSYC121 and PSYC122. MSc is more complicated, and as PSYC402 has lots of documents, to keep it cleaner, the two parts are split into separate folders and so the sidebar is tweaked so it can auto-show all the documents.
- a `404.qmd` file. This is what shows when someone goes to a nonexistent webpage, don't touch this.
- a `docs` folder - this is **very** important. This is created when the project is rendered, and contains the webpages for the site. You should not need to manually touch this.
- an `Images` folder - this holds necessary icons
- the `Includes` folder contains items that can be universally used (see below at includes)
- `index.qmd` - this can be edited. Under the Welcome banner, there is some text that should be altered to reflect something relevant about the year's content.
- Finally there should be two `PSYCxxx` folders. These contain the `.qmd` files that become the worksheets for students. This is where most of the changes are made on a weekly basis. More on these shortly.

Of importance is that the main materials, the workshop materials are held in the folder and named after the weeks - now these can be named anything, Week1 through 10, week11 through 20, whatever really but best to try and keep them sequential and clear of the order. These files are rendered and turned into html (web pages) and so these are where you create your teaching materials. For how to structure or what features to include (like callouts), refer to the Quarto docs: <https://quarto.org/docs/guide/>. But to emphasise, **these .qmd files are what create the website, this is what the students will interact with**, everything else is supplementary to these.

1.3 Other key things

As I said, there is abstracted content, and that you are unlikely to have access to, but these help maintain consistency across the board. These are in separate repositories (e.g. **essentials**). There isn't very much you need to know about this, just that it exists and any changes will need to go through someone with access to the repo (primarily whoever has control of the lu-psy-r account).

2 GitHub integration and workflow

Some of us are familiar with GitHub and version control, at least for personal projects, but perhaps less so for collaborating together. This is possibly the biggest “change” in the way that materials need to be managed. In theory, because only one person is likely to be working in a single module folder at a time, and each person working on unique documents, there should be minimal conflicts in files and so should be smooth.

2.1 Collaborators

To preemptively address what is potentially a concern, “what happens if someone breaks the website for everyone else?”. Valid question, but during the process detailed below and shown in the image below, if there is something wrong or breaking in any of the created content, then this fails on your local machine and by following the procedure detailed below, then damaging content shouldn’t have an opportunity to reach the server. If you received an error like below and proceeded to follow the instructions anyway, anything render up to and not including the breaking page will be added to the website. Any pages including and after will not be altered, so the broken page will need to be resolved. The errors will point you to the code chunks that are problematic. Note that this only captures truly broken content, incorrect hyperlink destinations won’t be considered as a break. In short, it should be hard to break the website. Even if you did, you can roll back on GitHub to a last working version.

First off, on GitHub, everyone will need to be added as a collaborator to ensure that they can directly push to the project and not require approval. This is done by going to the GitHub project > Settings > Collaborators. Then each GitHub account needs adding as a collaborator. More information can be found here: <https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-personal-account-settings/permission-levels-for-a-personal-account-repository#collaborator-access-for-a-repository-owned-by-a-personal-account>.

When making changes to content, it is best practice to first perform a pull request to get the latest version of the project (to reflect changes others may have made) by `git pull https://github.com/lu-psy-r/<enter-relevant-repo-here>.git` into an appropriate place on your local machine. You may be asked for your github user and password.

The you can make the changes you want to make. Add new files, alter your .qmd worksheets, whatever you want.

```
Statistics_for_PSYC — -zsh — 80x24

|.....| 54%
ordinary text without R code

|.....| 56%
label: write-csv (with options)
List of 1
$ eval: logi FALSE

|.....| 57%
ordinary text without R code

|.....| 59%
label: read-csv
Quitting from lines 732-739 (Week17.qmd)
Error: 'long.all.noNAs.csv' does not exist in current working directory ('/Users
/ivorym/Documents/PhD/Workbench/Statistics_for_PSYC/PSYC402').
In addition: Warning message:

In do_once((if (is_R_CMD_check()) stop else warning)("The function xfun::isFALSE
() will be deprecated in the future. Please ", :
  The function xfun::isFALSE() will be deprecated in the future. Please consider
  using base::isFALSE(x) or identical(x, FALSE) instead.
Execution halted
ivorym@M2 Statistics_for_PSYC %
```

Figure 2.1: Failed render in terminal

Note

Remember, students will only see the .qmd files as webpages, so anything else is effectively hidden!

2.2 Pushing changes to the site

There are two ways of doing this, through the terminal/command line, or through the RStudio interface. I describe both, but I would always recommend the terminal approach as it is more transparent and possible issues are clearer to see. I also have never really used the RStudio method, but I know people in the department use it - so I describe it vaguely.

2.2.1 via terminal

Once you've made the changes and you want to make these live changes to the site, the best thing to do is to click the render button in RStudio which gives you a preview of the materials. You can check you're happy with the format and presentation [here](#).

When you're happy with the preview, navigate to the directory in terminal, then run **quarto render**. This will create the webpages needed, and store them in the folder called "docs" (which is what the students actually see. Quarto render changes the qmd files to html and stores them here, and these are then displayed on the site). If the render process finds an issue, perhaps an error in the qmd files, it will abort and tell you where the issue is. Fix these and then re-render.

Warning

Sometimes the render fails and gives a weird complex looking error. If it doesn't point to a directly obvious issue, re-rendering usually fixes this

Once the render completes successfully, then you need to run the commands

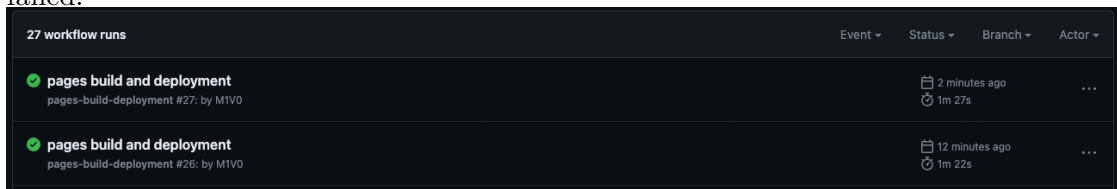
- `git add --all` - to add new files created
- `git commit -am "enter message"` - to commit your changes and to add a message to the changes, perhaps something like "added new week 3 content"
- and finally, `git push` to make the changes live. Website changes should be relatively quick, but can take five minutes or so sometimes. Depends on the changes really.

2.2.2 via RStudio

If memory serves, then if you have the repo git enabled (which it should be), then you'd see a `git` pane up by files/packages, click commit/push offers you a chance to add a message and then pushing actually renders the project simultaneously. I think that's it, but there may be a step or two I have missed.

Note

You can actually see the website change progress if you go to the GitHub repo and look at the actions tab. Obviously the repo may change, but use the postfix `/actions` which shows you upload progress. As the teaching materials grow more and more complex, changes take longer. For example, the initial uploads took ~35s and now take three times as long. If it is taken longer than normal, check the actions page to see if anything has failed.



The screenshot shows a table of GitHub Actions workflow runs. The table has columns for 'Event', 'Status', 'Branch', and 'Actor'. There are two rows of workflow runs, both titled 'pages build and deployment' and 'pages-build-deployment #27: by M1V0'. The first row shows a status of '2 minutes ago' and '1m 27s'. The second row shows a status of '12 minutes ago' and '1m 22s'.

Event	Status	Branch	Actor
pages build and deployment	2 minutes ago		...
pages-build-deployment #27: by M1V0	1m 27s		
pages build and deployment	12 minutes ago		...
pages-build-deployment #26: by M1V0	1m 22s		

And that's it. Because the preview and render actually test the content, it is difficult to upload anything that'll break the website.

Now there are potentially for two sticking points really.

1. During the render process, errors can appear that are unexpected. Broadly speaking, the render process will try and run anything that is an R file (`qmd`, `Rmd`, `R`) and run the code. If you've included dummy code in your `qmd` that isn't going to run successfully (intended bugs) then it will fail. You can fix this a couple of ways, the easiest being a chunk setting in the `qmd` file that specifies not to eval that chunk. More detail on this later.
2. During the push, failures here should be minimal and would likely come from merge conflicts, where multiple people have made changes to the same file and Git doesn't know which to use. This is handled in Git.

3 Creating and Managing Worksheets

This section runs through the general setup of a single week's qmd file. For a better understanding, take a look at the currently existing content.

Each module folder is set up however you wish as module coordinators, there are some basic rules to follow though for consistency. As the first year content has been up for a year, I can recommend looking at the content here as John and Tom have successfully integrated Panopto, slides, and content really nicely in PSYC121. Margriet and Rob have done an equally great job of doing something similar in PSYC122.

The index.qmd holds course-level content. The yaml contents for this is in the form:

```
title: Learning to use Quarto subtitle: "Course coordinators: Matthew Ivory" email-  
obfuscation: javascript order: 1
```

The first two lines are obvious, the third should stop your emails being scraped and used for spam (as best as one can) and the order option determines the order that this page is shown in the side menu.

The content for a worksheet is something like:

```
title: 1. Introduction to PSYC121 subtitle: Written by John Towse & Tom Beesley order: 2
```

Where the title starts with a number, to make it more obvious for the order and week, and not the order number is incrementally one above (because index.qmd is 1, so week1 is 2...).

Beyond the YAML, everything else is pretty much up to you and your module design. But as a general rule, the content has been moves to include a header for lectures, anything pre-lab, and then the actual worksheet. In other modules, it is arranged differently.

4 Features and Content

Here I will describe and include options in quarto to make more engaging content. A more complete list is seen here: <https://quarto.org/docs/authoring/markdown-basics.html>

4.1 Images

A pretty obvious one, but images are included by using the `! [Caption] (/Images/filepath.png)` and alt-text etc. can be added.

More information here: <https://quarto.org/docs/authoring/markdown-basics.html#links-images>

4.2 Callouts

Callouts are a great way to call attention to important information, but can also be used for creating collapsed answers. This first callout is a note and the code for it is below.

i Note

Here's a note for you to remember, like "Use the VPN!"

```
::: {.callout-tip}
Here's a tip for you to remember, like "Use the VPN!"
:::
```

This next callout is a collapsed tip which hides a simple answer for walkthroughs or testing knowledge. Maybe ask whether they can identify the correct p-value in a model output? Use it for answers, or perhaps hints that they can use if they need it but don't want a prompt straight away.

Answer

It doesn't work properly in a pdf, but on a website - hoo boy! It's hidden and students need to click on it to see it.

```
::: {.callout-tip icon="false" collapse="true"}  
## Answer  
What did you really expect here? Use your imagination!  
:::
```

4.3 Includes

Do you have a section of content that is repeated in each lecture, perhaps a how to access the RServer? With includes (<https://quarto.org/docs/authoring/includes.html>), you can create a separate document with whatever content you want, and then this can be simply inserted using the command:

```
{{< include _content.qmd >}}
```

It is worth reading the documentation for this on the Quarto site to understand how it works. A guide for accessing the server has been created and can be added into your worksheets with the line:

```
{{< include /Includes_login.qmd >}}
```

4.4 Embedding Panopto

This is a mildly more technical implementation, and works by leveraging the fact that markdown is primarily used for html content.

We can create an html chunk in the qmd file and then insert the embedded content for panopto lectures. What's good is that because panopto is behind cosign, only students can access the videos, so despite the worksheets being public, the lectures are still only accessible to the university.

So first create a chunk and where it shows the `r` in the backticks, replace this with `=html`, so the parentheses looks like `{=html}`. Then go to panopto, find the video and then click the settings cog. Then click share and then embed.

Overview

Share

Outputs

Quiz Results

Streams



Clips

Search


Captions

Manage


Log


 **People and groups**
0 added | 1 inherited from  **My Folder**


Add people and groups


 **Matt Ivory**
Moodlelvorym | m.ivory@lancaster.ac.uk


Creator

 **Who can access this video**
Anyone at your org can find and access [Change](#)

 Link

 Embed

 Facebook

 Twitter

☐ Autoplay

☒ Enable 'Watch in Panopto'

☒ Show Title

☐ Show Logo

☐ Show Captions

Aspect Ratio
16:9

Width (px)
720

Height (px)
405

☐ Start at 0:00

Interactivity
All

<iframe src="https://dtu-panopto.lancs.ac.uk/Panopto/Pages/Embed.aspx?i

Copy Embed Code

Click “Copy Embed Code”. Paste this into the chunk and that should be it, no other edits needed. An in-page lecture video.

5 The previous content repository

This section describes the process of adding previous year's content to the previous-content repository. It assumes you are the collaborator (which is likely the role of whoever is in charge of stats teaching).

- In the repo: create a new academic year folder (e.g. 2324 or 2425). The current existing folder is 2324 so make sure folders are consistently named.
- Go to the different active/present repositories and copy-paste each of the PSYCxxx folders into the new previous-content folder.
- open up `previous-content/_quarto.yml` and add a new section under the sidebar called 2023-24 (relevant to the year being added), add the section/contents under here to reflect the structure in the folders. This can either be taken from the `quarto.yml` files in the repositories, or copy-pasted from previous year content in the previous-content repo.

6 Technical

Here contains a bit more of the technical implementation of the site.

6.1 Folders with underscores

Folders or files that begin with an underscore are not rendered by quarto and serve to ensure certain content doesn't break the site or is visible. For example, in some of the `/module/data` folders, there are `.zip` files and then the uncompressed folder which has an underscore at the front so that Quarto doesn't try and read these and fail.

6.2 Search Bar Placeholder

This is not managed via the Quarto method of the `search-text-placeholder` argument because despite their documentation, I couldn't get it to work. So this is implemented via a workaround with a script included in the local `_quarto.yml` file for each repo that contains a script argument in the html header. This changes the placeholder. What a pain.

6.3 Freeze - collaboration caching

This was enabled in version 1, but it actually doesn't work as ideally as we want. Older week's content that isn't updated weekly doesn't updated the sidebar menus. SO this is removed, all this means is that if you are in a later module or a later part of a module, the render may fail if you don't have the installed packages as others on the year modules. Simply install these and re-render.