Using Quarto and RStudio to build Data fellowship module materials

MACEPA Data Fellowship - Training Materials

Table of contents

Starting Point

These resources are intended to help us build and maintain materials and modules for the Data Fellowship training materials. These can also serve as a useful resource when wanting to use Quarto and GitHub pages for other aspects of our work. Therefore these resources will continue to grow and evolve with us - send suggestions for content or improvements through the issues tab on the GitHub Repo.

System Requirements

Before you can start using Quarto to build out webpages or reports you need to ensure the following are installed on your computer

- R - https://www.r-project.org/
- Rstudio - https://posit.co/download/rstudio-desktop/
- Quarto - https://quarto.org/docs/download/
- Git - https://git-scm.com/downloads

Ensure you have a GitHub **profile** and are a member of **PATH-Global-Health organization**

If you haven't installe these packages before ensure the `usethis` `gh` `renv` and `quarto` packages are installed by typing this in the RStudio console:

```
install.packages(c("usethis", "gh", "renv", "quarto", "gitcreds"))
```

Configure Git with RStudio

After installing Git and setting up a GitHub account, the next step is to configure Git in RStudio. If you've linked your GitHub account to R already then skip ahead to **?@sec-stage**

1. Open RStudio.
2. Configure Git in RStudio:

- Go to **Tools** > **Global Options**.
- In the left-hand sidebar, click **Git/SVN**.
- Make sure the path to the Git executable is correct (this should automatically detect where Git was installed). For Windows, it might look something like `C:/Program Files/Git/bin/git.exe`.
- Click **Ok**

3. We want Git to know who we are so it can associate changes with you. Enter the following code in your console and replace the user name and email to those linked to your github account

```
usethis::use_git_config(user.name="Jane Doe", user.email="jane@example.org")
```

Linking to GitHub

When its time to send our files to GitHub, we need GitHub to know who we are and that we have permission to write to our repositories. We can establish this authorisation through either a **personal access token** or a **SSH key**.

We can generate a PAT through GitHub directly through https://github.com/settings/tokens and click "Generate token". Look over the scopes; I highly recommend selecting `repo`, `user`, and `workflow`. Copy the generated PAT to your clipboard. Or leave that browser window open and available for a little while, so you can come back to copy the PAT.

Another option is to create this directly in R using:

```
usethis::create_github_token()
```

Recommended scopes will be pre-selected if you used `create_github_token()`.

Now in R call, `gitcreds::gitcreds_set()` to get a prompt where you can paste your PAT:

```
> gitcreds::gitcreds_set()

? Enter password or token: ghp_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-> Adding new credentials...
-> Removing credentials from cache...
-> Done.
```

You should be able to work with GitHub now, i.e. push and pull!

What stage are you building from?

Before getting started we need to understand from what point we are building resources from - the following starting points cover the most common starting points we tend to find ourselves in

Starting Point	Description
Building from scratch	Start here if you want to learn about setting up a completly new resource and with tips and steps about using GitHub for the first time too
Building from an existing GitHub Repository	Start here if you are collaborating on resources and want to add additional materials to an already existing module

1 Building From Scratch

These resources are intented to be used if you are building module resrouces from scratch and need to set up a new Quarto Project, GitHub Repository and Associated GitHub Pages to host the materials.

See the table of contents to the right of this page for a summary of the steps involved in this process. Ensure you have followed the System requirements and have set up your GitHub profile and linked it in RStudio as detailed in the Start Point section.

1.1 Step 1: Set up new Quarto Book Project in RStudio

We suggest using the Quarto Book format for hosting training materials - there are several other Quarto output styles that you can work with but this what has worked well for this style of content delivery.

1. Open **RStudio**.
2. Go to the menu: `File` → `New Project...` → `New Directory` → `Quarto Book`.
3. **Name the project** and select the location where you want to save the project files on your computer. (I like to use a folder called github and all my repos exist in separate folders but pick what works for you!)
4. Ensure the checkbox for `Create a Git repository` is selected (this initializes Git locally in the project).
5. And also tick the checkbox for `use renv with this project`. This will create a local environment for the project and store all your package dependencies in an renv.lock file, this is an important aspect for hosting our site on gh-pages later on.
6. Click **Create Project**.

RStudio will open up your new project. The Quarto Book project structure will include:

- **_quarto.yml:** A configuration file where you can define the book's structure, theme, output format, and other settings.
- **index.qmd:** The landing page or introduction to the book. This file can serve as the cover page with a description of your project.
- **qmd files for chapters:** A few example .qmd files will be provided as placeholders for different chapters, which you can edit or replace with your content.

- An **renv** folder: This directory contains your project-specific package library, and an `renv.lock` file will be created to lock down package versions and dependencies, ensuring the project remains reproducible and code can be executed on publishing to our online portal.

1.2 Step 2: Stage initial commit to GitHub

Before linking this project to GitHub, we need to make sure the initial project files are committed to the local Git repository.

- Head to the `Terminal` tab next to the `console`.
- In the terminal, check which files are ready to be staged using:

```
git status
```

- This will show the files that have been modified or are new and need to be added to the repository. It will also tell us which branch we are working on in brackets. If this is the `master` branch lets change it to be called `main`.

```
git branch -m master main
```

- To add all files to the staging area (the files you want to include in your commit - here this will just be our default Quarto Book Project files which is okay, run:

```
git add .
```

- The `.` adds all the files in the current directory
- After staging the files, you'll need to commit them. The commit message should describe what changes or additions you're committing.To commit the changes, use: The `-m` flag allows you to add a message in quotes (`" "`) describing the commit.

```
git commit -m "Initial commit for Quarto website"
```

1.3 Step 3 Push the local project to GitHub using `usethis::use_github()`

We now want to link our local repository to GitHub and specially we want it to be part of the PATH-Global-Health GitHub organisation. We can use the following code to do this, run this in your console:

```
usethis::use_github(
organisation = "PATH-Global-Health",
visibility = "public"
)
```

This command will:

- Create a new GitHub repository.
- Link your local project to this repository.
- Push the project files to GitHub.

This should then open up the repository automatically in you browser.

1.4 Step 4: Setting up gh-pages

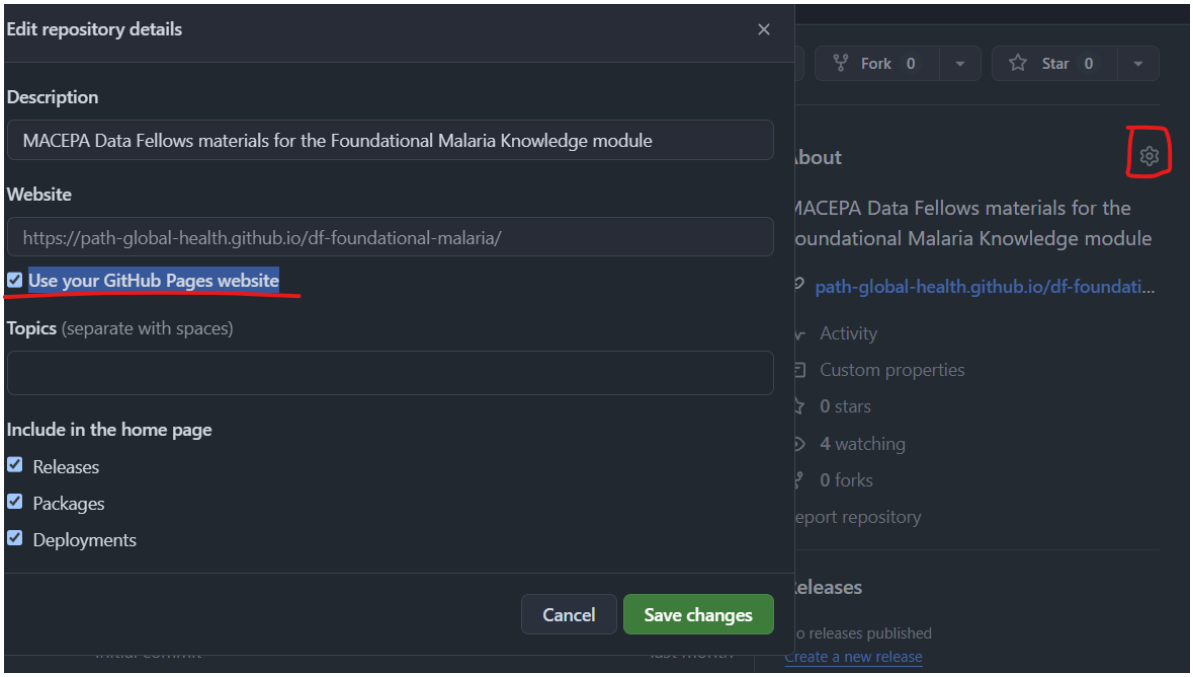
Once our repository is on GitHub, we can configure the GitHub Pages site - which is where our module resources will be hosted. Use the `usethis::use_github_pages()` function to set the publishing branch for GitHub Pages.

To publish from the `gh-pages` branch, run:

```
r
usethis::use_github_pages(branch = "gh-pages")
```

If we head to our GitHub repository online we want to add some details to the repo page:

Head back to the `<> Code` tab and in the `About` section on the top right open the settings wheel - Under `Website` check the box next to: [x] "Use your GitHub Pages website" as shown in the image below.



In addition we can add a short description in this section as in the above image e.g. "MACEPA Data Fellows materials for the [insert module title]".

1.5 Step 5: Automate Deployment with GitHub Actions

This is something I've found works best for me and my workstyle when creating these modules. Instead of ever rendering my work locally and then publishing this to GitHub I include a GitHub Action command so that when I commit and push changes to the repository GitHub will automatically render the new outputs to the `gh-pages` site.

Manually building and deploying our project every time we make a change can be time-consuming and prone to error. So by configuring GitHub Actions, we can automate the entire publishing process. Whenever we push changes to the repository (e.g., updated content, code adjustments), GitHub Actions will automatically trigger the workflow to build and deploy our site. Which saves us time and reduces manual effort. This also helps ensure that everyone is working on the most recent version of the materials, with automatic deployment occurring in the background.

1.5.1 Set up

More detials on setting up GitHub actions can be found here: https://quarto.org/docs/publishing/github-pages.html.

1. In your Quarto project directory, create a folder called `.github/worflows`
2. Inside `.github/workflows/`, create a file called `quarto-publish.yml` - You do this from within RStudio by heading to the `files` pane and into the `workflows folder` → `new blank file` → `Text file` and this opens up in R studio and then save this as `quarto-publish.yml`
3. Add the following content to the `quarto-publish.yml`

```yaml
on:
  workflow_dispatch:
  push:
    branches: main

name: Quarto Publish

jobs:
  build-deploy:
    runs-on: ubuntu-latest
    permissions:
      contents: write
    steps:
      - name: Check out repository
        uses: actions/checkout@v4

      - name: Set up Quarto
        uses: quarto-dev/quarto-actions/setup@v2

      - name: Install R
        uses: r-lib/actions/setup-r@v2
        with:
          r-version: '4.2.0'

      - name: Install R Dependencies
        uses: r-lib/actions/setup-renv@v2
        with:
          cache-version: 1

      - name: Install TinyTeX
        run: |
          Rscript -e 'tinytex::install_tinytex(force = TRUE)'
          echo "PATH=$HOME/.TinyTeX/bin/x86_64-linux:$PATH" >> $GITHUB_ENV

      - name: Ensure TinyTeX Path
```

```
        run: echo "$HOME/.TinyTeX/bin/x86_64-linux" >> $GITHUB_PATH

      - name: Render and Publish
        uses: quarto-dev/quarto-actions/publish@v2
        with:
          target: gh-pages
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

4. call `renv::snapshot()`, `snapshot()` updates the lockfile with metadata about the currently-used packages in the project library.

This `quarto-publish.yml` I have worked on standardising across the projects I've developed for the Data Fellows so far - should you have any issues please reach out to Hayley to help troubleshoot!

Other options for publishing content can be found here: https://quarto.org/docs/publishing/

1.6 Step 6 Push updates to GitHub

We can now push the all of the following changes to GitHub and test if the publishing action has worked - don't worry that we haven't changed any content yet we will get there!

Switch to the `Terminal` pane and run the following:

```
git add .
git commit -m "deploying and testing github actions and publishing"
git push origin main
```

2 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```