**first-timer**

**1. How to collaborate across timezones**

There is a 9-hour time difference between Maëlle and me for most of the year. It helped immensely to have a weekly 30-minute meeting (8:30am for Stef and 5:30pm for Maëlle) with agenda and notes in a shared google doc. We would talk through our approaches and priorities and Maëlle would coach me in new-to-me tools.

At the end of a day, I would messaged to say *“I’m finished for the day, can you please review and merge my pull requests?"* or *“Please review the structure but not the text yet”*. Next morning, Maëlle would have done that plus her own work so I could update my local copy of our bookdown book and open new pull requests for new pieces of work.

**2. How to collaborate in RStudio & GitHub**

* For setting up RStudio and knowing how to open a project from .Rproj I loved R-Ladies Sydney’s

**3. How to usethis**

I also followed the rest of the setup instructions somewhat religiously (e.g. “Prepare your system to build packages from source”), because I wasn’t confident enough to make a judgement on what I would and would not need.

Practically, I kept the usethis [pull request helpers](https://usethis.r-lib.org/articles/articles/pr-functions.html) article open in my browser and forced myself to use them.  
At the start of the day, I would open our bookdown RStudio project by clicking my local ropensci-blog-guidance.Rproj file, pull to update my local master (because Maëlle would have reviewed, edited and merged my pull requests while I slept), create a new branch with pr\_init(branch = "branchname"), make edits and commits, and pr\_push() to push my local changes to GitHub master.  
usethis automatically opened a browser to the GitHub web interface prompting me to open a pull request.

There’s no magic conclusion here.  
This feels a bit better than working on pull requests on the command line.  
I expect by forcing myself to usethis I’ll discover some magic soon enough.

**4. How to fill up a bunch of Rmd files to make a book**

To answer the question, “how do I do this thing?” I often compared the  
GitHub file structure and contents of completed books with their corresponding live pages. This, for me, had the biggest payoff in learning how to bookdown.

Things I learned:

* One book chapter is made from one Rmd file.
* Chapters are woven together in the \_bookdown.yml file that references those Rmd files
* Create links inside the book with {#anchor}. In the authorcontent.Rmd file, the heading # Content Guidelines {#content} means that a markdown format [link to that chapter](#content) anywhere in the book will link to the Content Guidelines chapter.
* Appendices A to H (*too many; we know*) are created from a single appendix.Rmd file made up of groupings of a heading, some text, and sometimes a code chunk that points to a file, like a template or checklist that populates that appendix.

I often felt quite euphoric about the things I was able to figure out comparing GitHub file structures to their books.  
Consistently, within days I would take this hard-earned knowledge for granted and feel inadequate in the face of my next technical challenge…until I felt the satisfaction of owning that next one too.

**5. How to … everything**

What doesn’t appear in this list of tips are all the things I’ve already forgotten that I had trouble with and learned to overcome by reading error messages and searching and poking at my setup until things worked.  
Looking back in Slack conversations with Maëlle I see that setting up usethis to use my GitHub credentials and serving the bookdown preview using the RStudio Addin were tricky.  
But errors had a lot to do with making sure the packages I was using were up to date .This humbling bookdown experience required me to figure out a whole new workflow and up my skills.

**Maëlle (more experienced bookdowner)**

**1. How to start a bookdown project**

Both of the bookdown projects using [Sean Kross’ excellent primer](https://github.com/seankross/bookdown-start),  
but whilst looking for a reference to show Stef[3](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#fn:3),

**2. How to get the copy-paste button for code chunks**

In the blog guidance, if you hover around the top-right corner of [e.g. the Markdown post template](https://blogguide.ropensci.org/templatemd.html) you get a copy-paste button.  
For this to work, the chunk needs to have some language information i.e.

```

code

```

will not get a copy-paste button, but

```yaml

code

```

will!  
I’m glad I know that now.  
Chunks with language info are prettier anyway since they get adapted code highlighting.

**3. How to define functions and chunk options for all chapters**

knitr::opts\_chunk$set(

cache = TRUE,

echo = FALSE

)

library("magrittr")

show\_template <- function(filename,

lang = "markdown",

details = FALSE,

yaml\_only = FALSE,

...) {

lines <- suppressWarnings(

if(grepl("roweb2", filename)) {

readLines(filename)

} else {

readLines(

file.path("templates", filename)

)

}

)

if (yaml\_only) {

lines <- bookdown:::fetch\_yaml(lines)

}

lines %>%

glue::glue\_collapse(sep = "\n") -> template

if (details) {

toshow <- details::details(template, summary = filename,

lang = lang,

...)

} else {

toshow <- glue::glue("````{lang}\n{template}\n````")

}

return(toshow)

}

show\_checklist <- function(filenames) {

filenames <- file.path("checklists", filenames)

purrr::map(filenames,

readLines) %>%

unlist() %>%

gluedown::md\_task() %>%

glue::glue\_collapse("\n") -> x

glue::glue("````markdown\n{x}\n````") %>%

knitr::asis\_output()

}

Our bookdown project uses DESCRIPTION to track dependencies, I suppose I could use the package infrastructure more and define the helper functions as functions *of a package*, but the approach above is pleasant too.

**4. How to have alternative text but no captions for figures**

In R Markdown, the same chunk options fig.cap controls the caption and alternative text of images.

.caption {

display: none;

}

**5. How to deploy a preview of the book for pull requests**

Here’s what we now have

* after each commit to the master branch, the book is built and deployed to the gh-pages branch that points to the blog guide official URL.

on:

push:

branches:

master

name: Render-Book-from-master

jobs:

bookdown:

name: Render-Book

runs-on: macOS-latest

steps:

- uses: actions/checkout@v1

- uses: r-lib/actions/setup-r@v1

- uses: r-lib/actions/setup-pandoc@v1

- name: Query dependencies

run:

Rscript -e "install.packages('remotes')" -e "saveRDS(remotes::dev\_package\_deps(dependencies = TRUE), 'depends.Rds', version = 2)"

- name: Cache R packages

uses: actions/cache@v1

with:

path: ${{ env.R\_LIBS\_USER }}

key: ${{ runner.os }}-r-${{ hashFiles('depends.Rds') }}

restore-keys: ${{ runner.os }}-r-

- name: Install dependencies

run:

Rscript -e "library(remotes)" -e "deps <- readRDS('depends.Rds')" -e "deps[['installed']] <- vapply(deps[['package']], remotes:::local\_sha, character(1))" -e "update(deps)"

- name: Render Book

run: Rscript -e 'bookdown::render\_book("index.Rmd")'

- name: Commit results

if: github.repository == 'ropensci-org/blog-guidance'

run: |

cp ghpagescname docs/CNAME

cp -r favicon/ docs/

cd docs

git init

git add .

git commit -m 'update book'

git push

on: pull\_request

name: PR-workflow

jobs:

bookdown:

name: Render Book

runs-on: macOS-latest

steps:

- name: Is this a fork

run: |

fork=$(jq --raw-output .pull\_request.head.repo.fork "${GITHUB\_EVENT\_PATH}");echo "::set-env name=fork::$fork"

- uses: actions/checkout@v1

- uses: r-lib/actions/setup-r@v1

- uses: r-lib/actions/setup-pandoc@v1

- name: Query dependencies

run:

Rscript -e "install.packages('remotes')" -e "saveRDS(remotes::dev\_package\_deps(dependencies = TRUE), 'depends.Rds', version = 2)"

- name: Cache R packages

uses: actions/cache@v1

with:

path: ${{ env.R\_LIBS\_USER }}

key: ${{ runner.os }}-r-${{ hashFiles('depends.Rds') }}

restore-keys: ${{ runner.os }}-r-

- name: Install dependencies

run:

Rscript -e "library(remotes)" -e "deps <- readRDS('depends.Rds')" -e "deps[['installed']] <- vapply(deps[['package']], remotes:::local\_sha, character(1))" -e "update(deps)"

- name: Render Book

run: Rscript -e 'bookdown::render\_book("index.Rmd")'

- uses: actions/setup-node@v1

if: env.fork == 'false'

with:

node-version: "12.x"

- name: Install Netlify CLI

if: env.fork == 'false'

run: npm install netlify-cli -g

- name: Deploy to Netlify (test)

if: env.fork == 'false'

run: DEPLOY\_URL=$(netlify deploy --site ${{ secrets.NETLIFY\_SITE\_ID }} --auth ${{ secrets.NETLIFY\_AUTH\_TOKEN }} --dir=docs --json | jq '.deploy\_url' --raw-output);echo "::set-env name=DEPLOY\_URL::$DEPLOY\_URL"

- name: Create check

if: env.fork == 'false'

run: |

curl --request POST \

--url https://[api.github.com/repos/${{](http://api.github.com/repos/$%7B%7B) github.repository }}/check-runs \

--header 'authorization: Bearer ${{ secrets.GITHUB\_TOKEN }}' \

--header 'Accept: application/vnd.github.antiope-preview+json' \

--header 'content-type: application/json' \

--data '{

"name": "Preview Book",

"head\_sha": "${{ github.event.pull\_request.head.sha }}",

"conclusion": "success",

"output": {

"title": "Preview link",

"summary": "[Preview link](${{ env.DEPLOY\_URL }}) :rocket:"

}

}'

Highlights from the pull request workflow above:

* To deploy to Netlify *and* get the preview URL, the workflow doesn’t use Netlify’s GitHub Actions but instead installs Netlify CLI, extracts the URL using jq[4](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#fn:4) and sets it as an environment variable that can be used by next steps.
* run: DEPLOY\_URL=$(netlify deploy --site ${{ secrets.NETLIFY\_SITE\_ID }} --auth ${{ secrets.NETLIFY\_AUTH\_TOKEN }} --dir=docs --json | jq '.deploy\_url' --raw-output);echo "::set-env name=DEPLOY\_URL::$DEPLOY\_URL"

After the successful deployment of a preview, in the pull request checks, one check called “Preview book” appears.

* We want to skip all deploy steps for pull requests *from forks*, since forks don’t have access to the Netlify secrets[5](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#fn:5) saved in our repo settings. To find out whether the pull request is from a fork, - name: Is this a fork

run: |

fork=$(jq --raw-output .pull\_request.head.repo.fork "${GITHUB\_EVENT\_PATH}");echo "::set-env name=fork::$fork"

And some of the further steps are skipped based on fork.

- name: Install Netlify CLI

if: env.fork == 'false'

run: npm install netlify-cli -g

**Conclusion**

In this post, we shared tips and things we learned from novice and (more) experienced perspectives on bookdown and R project management