Technically, we structured the content as a bookdown gitbook. It was Stef’s first foray into the glorious process of publishing a book with bookdown, and Maëlle’s second[1](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#fn:1). And oh, we learned a lot.

**Stef (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.

Part way through the project, we got more strict about [assigning issues to ourselves](https://help.github.com/en/github/managing-your-work-on-github/assigning-issues-and-pull-requests-to-other-github-users) and [tying them to specific milestones](https://help.github.com/en/github/managing-your-work-on-github/about-milestones) like [“official release”](https://github.com/ropensci-org/blog-guidance/milestone/3) or [“nice to have one day”](https://github.com/ropensci-org/blog-guidance/milestone/1). This helped us work asynchronously toward a common goal, showed some light at the end of the tunnel, as well as helping us (try to) avoid [scope-creep](https://en.wikipedia.org/wiki/Scope_creep).

**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 “[opinionated tour of RStudio](https://rladiessydney.org/courses/ryouwithme/01-basicbasics-1/)".
* I used happygitwithR “[Connect RStudio to Git and GitHub](https://happygitwithr.com/rstudio-git-github.html)” to do exactly that.  
  Years ago I had already used [happygitwithR](https://happygitwithr.com/) to set up my git/GitHub from square-one and my https credentials.
* Maëlle and I both worked on [feature branches of master](https://ropensci.org/blog/2018/04/20/monkeydo/#gitflow).  
  On any given day we each might work on mulitple chapters of the book. Because each book chapter is generated from one Rmd file (more on that in [tip #4](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#4-how-to-fill-up-a-bunch-of-rmd-files-to-make-a-book)), and you don’t want one pull request to be *too* complex, I would create a separate feature branch for any chapter or file I worked on. I would periodically push my work to GitHub in a pull request (more in [tip #3](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#3-how-to-usethis)) with ["*Draft*” status](https://github.blog/2019-02-14-introducing-draft-pull-requests/) so Maëlle could see I was not yet seeking her review. At the end of a day’s commits, I would update the pull request status to “*Ready for review*” and assign Maëlle as reviewer. She would review, edit, and merge my pull requests and open new branches of her own for me to review.
* Adding “Fix #54” in a pull request description [automatically closes issue](https://help.github.com/en/github/managing-your-work-on-github/linking-a-pull-request-to-an-issue) #54 when the pull request is merged into master.

**3. How to usethis**

*"[usethis](https://usethis.r-lib.org/" \t "_blank) is a workflow package: it automates repetitive tasks that arise during project setup and development, both for R packages and non-package projects."*  
The trickiest part, for me, was commiting to setting up and forcing myself to use usethis pull request helpers despite not really understanding how it would help me.

From previous work I had already followed happygitwithr to set up my [GitHub authentication via https](https://happygitwithr.com/credential-caching.html) and a [GitHub Personal Access Token (PAT)](https://happygitwithr.com/github-pat.html) in my .Renviron.  
I followed usethis [setup instructions](https://usethis.r-lib.org/articles/articles/usethis-setup.html) to edit my .Rprofile so that its functions would be available without explicitly calling the package.  
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](https://github.com/ropensci-org/blog-guidance/blob/master/_bookdown.yml) file that references those Rmd files
* Create links inside the book with {#anchor}. In the [authorcontent.Rmd](https://github.com/ropensci-org/blog-guidance/blob/master/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](https://github.com/ropensci-org/blog-guidance/blob/master/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](https://github.com/ropensci-org/blog-guidance/tree/master/templates) or [checklist](https://github.com/ropensci-org/blog-guidance/tree/master/checklists) 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.

These bookdown books were helpful in exploring those comparisons:

* rOpenSci Packages: Development, Maintenance, and Peer Review [source on GitHub](https://github.com/ropensci/dev_guide) *cf* [deployed book](https://devguide.ropensci.org/);
* R for Data Science [source on GitHub](https://github.com/hadley/r4ds) *cf* [deployed book](https://r4ds.had.co.nz/).

**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 (thus my calendar reminder in [tip #2](https://ropensci.org/blog/2020/04/07/bookdown-learnings/#2-how-to-collaborate-in-rstudio-github)) and needing to install the [development version of usethis](https://community.rstudio.com/t/trouble-with-credentials-in-usethis-pr-init/35136).  
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),  
I saw a [tweet of Alison Hill’s about starting a bookdown project from RStudio](https://twitter.com/apreshill/status/1137183351125577728) which looks handy.  
[Slide of Alison’s showing how to start a bookdown from RStudio](https://arm.rbind.io/slides/bookdown.html#11).

**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**

I found how to write code that’ll be loaded before each chapter rendering thanks to [Christophe Dervieux’s answer on an old RStudio community thread](https://community.rstudio.com/t/bookdown-caching/43652/2).

One simply needs to create an [R script called \_common.R at the root of the bookdown project](https://raw.githubusercontent.com/ropensci-org/blog-guidance/master/_common.R).  
Here’s ours below.  
It contains chunk options, magrittr loading, and two helper functions for rendering templates.

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.  
We wanted [alternative text](https://www.w3.org/WAI/tutorials/images/informative/) but no caption.  
The header option figure\_caption didn’t work.

So if bookdown doesn’t really support having alternative text but no captions for figures, what did we do? Thanks to a good tip by [Romain Lesur](https://github.com/RLesur), I wrote CSS to remove all captions.

.caption {

display: none;

}

The lines above live with other styling stuff in a file called style.css, that we refer to in [the \_output.yml config file](https://github.com/ropensci-org/blog-guidance/blob/cae5887f392b4137eec50467779e9bba973b0b69/_output.yml#L3).

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

We achieved that using [GitHub Actions](https://help.github.com/en/actions), a new CI service by GitHub.  
If you’re curious about it, I’d recommend watching [Jim Hester’s talk from the RStudio conference earlier this year](https://resources.rstudio.com/rstudio-conf-2020/azure-pipelines-and-github-actions-jim-hester), and having a look at [the exploration book](https://ropenscilabs.github.io/actions_sandbox/) written by participants of the [Oz UnConf 2019](https://ropensci.org/blog/2020/02/05/ozunconf19/).  
Our GitHub Actions workflows make good use of [Jim’s actions and examples](https://github.com/r-lib/actions).

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. Code below, [log example](https://github.com/ropensci-org/blog-guidance/runs/471978792?check_suite_focus=true);

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 https://${{github.actor}}:${{[secrets.GITHUB\_TOKEN}}@github.com/${{github.repository}}.git](http://secrets.GITHUB_TOKEN%7D%7D@github.com/$%7B%7Bgithub.repository%7D%7D.git) HEAD:gh-pages --force

* after each commit in a pull request from a fork, the book is built so we’d notice something breaking the Rmd. [log example](https://github.com/ropensci-org/blog-guidance/runs/466995315?check_suite_focus=true);
* after each commit in a pull request from the repo, the book is built and deployed to a Netlify preview whose URL is posted in the pull request checks. [Log example](https://github.com/ropensci-org/blog-guidance/runs/471968808?check_suite_focus=true), [Direct link to the check with the preview link](https://github.com/ropensci-org/blog-guidance/runs/471974005?check_suite_focus=true).

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. I got the idea from [a thread on the Netlify forum](https://community.netlify.com/t/deploying-preview-web-hook-via-api/3320/2).

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"

* To post the preview URL in the pull request, the workflow uses [a GitHub check run](https://developer.github.com/v3/checks/runs/). You can only create and patch them as a GitHub app. Luckily when in a GitHub Actions runner the default token has these permissions! My only regret is that trying to post the preview URL as a “details” URL for that check didn’t work (GitHub resets it to the check page on GitHub Actions) but it’s not too bad.

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

When clicking on “Details” one gets to [a check page where the preview link is prominent](https://github.com/ropensci-org/blog-guidance/runs/471974005?check_suite_focus=true).

* 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, I use jq on the raw info about the build, idea I got [from Vanessa Sochat](https://github.com/r-universe/vsoch/blob/2bf10b1e30f59f5fab64ec2b7332526c47f1f4d3/.github/workflows/pull-request-update-packages.yml#L26).

- 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

If you’re feeling motivated to add GitHub Actions deployment to your bookdown project, be sure to note that Emil Hvitfeld wrote an [excellent guide to deploying your book on Netlify with GitHub Actions](https://www.hvitfeldt.me/blog/bookdown-netlify-github-actions/), with screenshots!

**Conclusion**

In this post, we shared tips and things we learned from novice and (more) experienced perspectives on bookdown and R project management whilst working on the new [rOpenSci Blog Guide for Authors and Editors](https://blogguide.ropensci.org/).  
We’re almost as happy with our new skills as we are with the Blog Guide itself!