Test Quarto Book

Brad Cannell

2023-06-24

Table of contents

Preface

This is my first Quarto book. For now, I'm just using it for some experimentation. Eventually, I will probably want to add all of this to a GitHub repository. Right now, Olivia is in class and I'm content with just experimenting a little bit.

This is **not** my new R Notes book; although, I may move R Notes over to Quarto at some point. This is purely just a sandbox for playing with Quarto books.

Useful websites:

- Quarto book documentation

Rendering

You can render the files by clicking the Render button in RStudio. To render the HTML and PDF files at the same time, type `quarto render` into the terminal. You can also render Quarto files with a native R code chunk.

- The input argument: The input file or project directory to be rendered (defaults to rendering the project in the current working directory).
- The output_format argument: Target output format (defaults to "html"). The option "all" will render all formats defined within the file or project.

~~~
```{r}
#| Render with R
#| eval: false
quarto::quarto_render(output_format = "all")
```
~~~

Publishing to GitHub pages

This article is great. After committing, and making sure you are on the main branch, type `quarto publish gh-pages` in the terminal.

Part I

part_my_chapters.qmd

1 Book Options

A collection of notes on Quarto book options.

- Link to list of book options
- How do you add a cover image? Look at r4ds.
  - `cover-image: cover.jpg`
- How do you add last date rendered to the `_quarto.yml` file? "r Sys.Date()" doesn't seem to work.
  - Use the keyword today
- Can I add links to GitHub and/or social media?
  - Still need an answer here.
- Can I add links to the GitHub repo containing the books files?
  - Yes. See https://quarto.org/docs/books/book-output.html#sidebar-tools
- How do I add a favicon?
  - `favicon: cover.jpg`
- How do I add Google analytics?
  - Still need an answer here.
- Can I add a Google analytics badge to my GitHub README?
  - Still need an answer here.
- How do I preview the book in my web browser instead of RStudio's Viewer pane?
  - Just click the little gear icon next to the `Render` button in RStudio. Select `Preview in Window`.

2 Other little things

2.1 Pros and cons

Some of the things I like about working with Quarto (as opposed to bookdown) so far

- A preview of the book renders automatically.
- I can easily render only one chapter by opening that chapter's qmd file and clicking the Render button.

Part II

Authoring

3 Images

Links

- The Quarto documentation about images
- Example image in R4DS

3.1 Native Quarto figures

Here are some examples of adding figures.

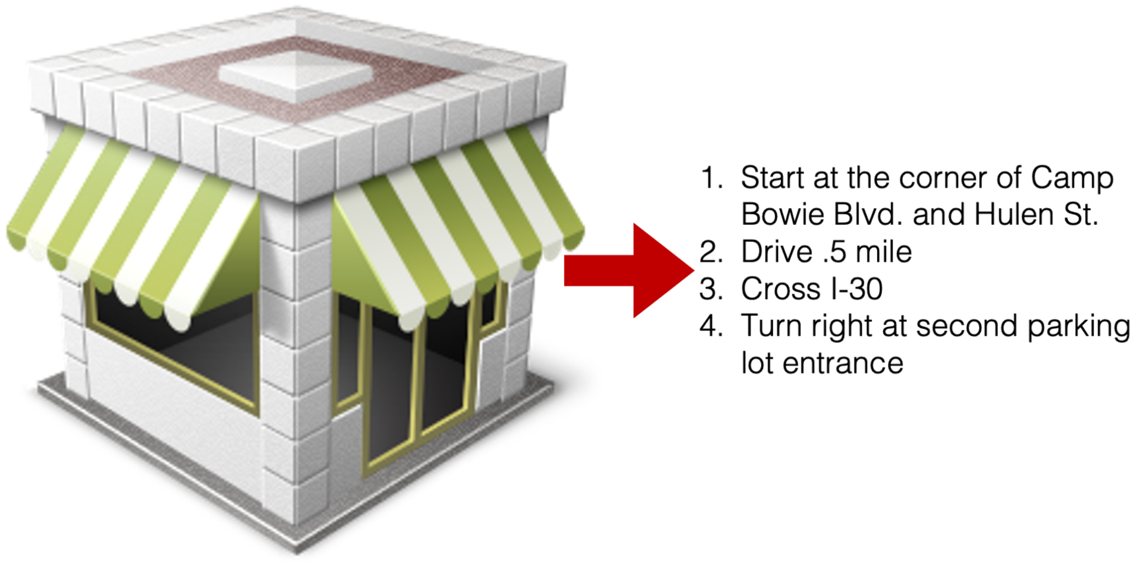
1. Start at the corner of Camp Bowie Blvd. and Hulen St.
2. Drive .5 mile
3. Cross I-30
4. Turn right at second parking lot entrance

Figure 3.1: Relative Path Directions

And I can cross reference the figure by typing `@fig-directions`. See Figure 3.2

! Important

For cross-references to work, the image must have a caption *and* a label.

3.2 Adding figures with Knitr

In R4DS (link above), Hadley et al. are still using `knitr::include_graphics("path")` to insert images even though the book has been converted to Quarto documents. When using Bookdown, Yihui gives four arguments for using `knitr::include_graphics("path")` instead of native markdown image formatting. So, we will likely continue to use them too. Here is an example image code chunk from R4DS:

~~~
```{r}
#| label: fig-ds-diagram
#| echo: false
#| fig-cap: |
#|   In our model of the data science process, you start with data import
#|   and tidying. Next, you understand your data with an iterative cycle of
#|   transforming, visualizing, and modeling. You finish the process
#|   by communicating your results to other humans.
#| fig-alt: |
#|   A diagram displaying the data science cycle: Import -> Tidy -> Understand
#|   (which has the phases Transform -> Visualize -> Model in a cycle) ->
#|   Communicate. Surrounding all of these is Communicate.
#| out.width: NULL

knitr::include_graphics("diagrams/data-science/base.png", dpi = 270)
```
~~~

Here, I'm adding my own image with `knitr::include_graphics("path")`.

And I can cross reference the figure by typing `@fig-directions`. See Figure 3.2

! Important

For cross-references to work, the image must have a caption *and* a label.

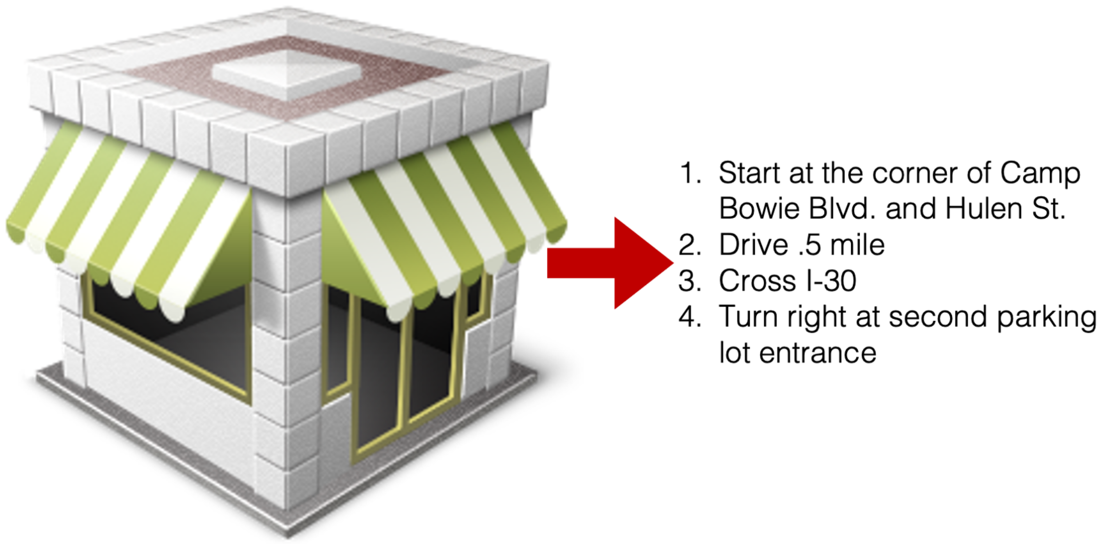
1. Start at the corner of Camp Bowie Blvd. and Hulen St.
2. Drive .5 mile
3. Cross I-30
4. Turn right at second parking lot entrance

Figure 3.2: Relative Path Directions.

4 Gifs

In R4Epi, we use quite a few gifs. That makes rendering the book to pdf format challenging. What happens when we add a gif to a qmd document and render it to pdf?

The following code works great for HTML format, but it messes up pdf format.

```
![A gif about file paths](img/file_path_gif.gif){#fig-file-paths}

And I can [cross reference](https://quarto.org/docs/authoring/cross-references.html) the g

::: {.callout-important}
For cross-references to work, the image must have a caption _and_ a label.
:::
```

What if I use `knitr::include_graphics("path")`?

~~~
```{r}
#| label: fig-file-paths
#| echo: false
#| fig-cap: |
#|   A gif about file paths.
#| fig-alt: |
#|   A gif about file paths.

knitr::include_graphics("img/file_path_gif.gif")
```

And I can [cross reference](https://quarto.org/docs/authoring/cross-references.html) the f

::: {.callout-important}
For cross-references to work, the image must have a caption _and_ a label.
:::
~~~

No, this doesn't work either. Here is a pretty good discussion on the topic. It looks like the easiest route may be to use conditional content. Given the limited number of gifs in R4Epi, this shouldn't be too big of a problem.

So, start by conditionally displaying the gif if the output format is html.

~~~
::: {.content-visible when-format="html"}

```{r}
#| label: fig-file-paths
#| echo: false
#| fig-cap: |
#|   A gif about file paths.
#| fig-alt: |
#|   A gif about file paths.

knitr::include_graphics("img/file_path_gif.gif")
```

:::
~~~

Which renders as…

Then, conditionally add a thumbnail image of the gif when the output format is pdf.

- Create a duplicate of the gif in Finder.
- Add "_thumb" to the end of the file name (before the file extension).
- Open the duplicate file in preview.
- Use Preview's sidebar to keep only one of the thumbnail images from the gif (select and delete the rest).
- Click File > Export and export to png.
- Delete the duplicate gif.

~~~
::: {.content-visible when-format="pdf"}

```{r}
#| label: fig-file-paths
#| echo: false
#| fig-cap: |
#|   A thumbnail of gif about file paths.
#| fig-alt: |
#|   A thumbnail gif about file paths.

knitr::include_graphics("img/file_path_gif.gif")
```
~~~

```
:::
```

Which renders as…

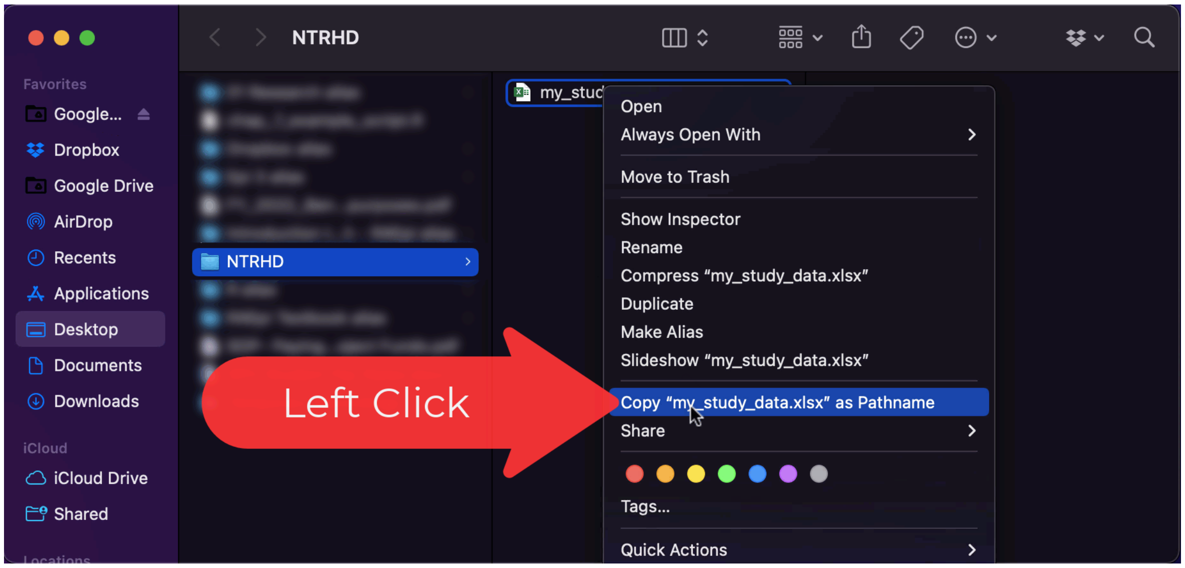

Figure 4.1: A thumbnail of gif about file paths.

5 Cross References

When authoring the book, it is common to need to cross reference book parts (i.e., chapters, sections), figures, tables, and I'm going to go ahead and include citations here too.

5.1 Book parts

Link to the official documentation

Gotcha's: - Sometimes I get errors like `WARNING: index.html: Unable to resolve crossref @sec-crossrefs`

5.2 Citations

Built-in example: See Knuth (1984) for additional discussion of literate programming.

6 Adding call out boxes

In R4Epi, we sometimes add special boxes for side notes and warnings. How do we add those into qmd books?

What do those boxes look like when we render the book to pdf format?

Does Quarto have some built-in boxes? It looks like it might?

- How do you add the call out boxes (i.e., important, etc.)?
  - See Quarto documentation on callout blocks

```
::: callout-note
Note that there are five types of callouts, including: `note`, `tip`, `warning`, `caution`
:::
```

Note

Note that there are five types of callouts, including: `note`, `tip`, `warning`, `caution`, and `important`.

```
::: callout-warning
Callouts provide a simple way to attract attention, for example, to this warning.
:::
```

Warning

Callouts provide a simple way to attract attention, for example, to this warning.

```
::: callout-important
## This is Important

Danger, callouts will really improve your writing.
:::
```

This is Important

Danger, callouts will really improve your writing.

```
::: callout-tip
## Tip With Title

This is an example of a callout with a title.
:::
```

Tip With Title

This is an example of a callout with a title.

```
::: {.callout-caution collapse="true"}
## Expand To Learn About Collapse

This is an example of a 'collapsed' caution callout that can be expanded by the user. You
:::
```

Expand To Learn About Collapse

This is an example of a ‘collapsed’ caution callout that can be expanded by the user. You can use `collapse="true"` to collapse it by default or `collapse="false"` to make a collapsible callout that is expanded by default.

Part III

Working with Code

7 Code chunks

These are my notes on working with code chunks.

Here are some other helpful resources:

- Need to add a link to a comprehensive guide

7.1 Code Run Time

We were interested in understanding how much caching code chunks could potentially speed up the rendering process for books. Initially, the code to test out caching code chunks was part of the `test_quarto_book` project. However, it seemed like the `quarto_render()` function works differently in the context of a book project and was making testing difficult. Specifically, `quarto::quarto_render("slow_code_default.qmd")` seemed to render the entire book instead of just the `slow_code_default.qmd` document. Therefore, we created a separate Quarto project to run these tests – `test_quarto_cached_chunks`.

The bottom line is that caching actually causes the render to take longer to process on the first run. However, it appears to speed up subsequent renders. This makes sense.

7.2 Naming code chunks

Use the `label` code chunk option. For example, `#| label: slow-code-cached`

7.3 Showing entire code chunks

When you want to display and entire code chunk in the rendered output, as opposed to executing the code chunk, surround it with four backticks.

Also, make sure to use double curly braces `{{}}` around the language name identifier. If you don't, funky looking code will be output.

~~~~~
````
```{r}
#| A code chunk for display
#| eval:false
1 + 1
```
````
~~~~~

8 Sourcing qmd files

We can “source” or programmatically render (as opposed to interactively rendering with the `Render` button) qmd files – including inside of code chunks in other qmd files. Here is an example code chunk.

```
quarto::quarto_render("slow_code_cached.qmd", quiet = TRUE)
```

Part IV

Publishing

9 PDF

Notes on exporting the book to pdf.

Not only how to render it to pdf, but is there an option

According to this, it looks like we can make a pdf downloadable by adding `downloads: [pdf, epub]` to `_quarto.yml`. In the config file for this test book, we added `downloads: pdf` to the book options section.

Important: It looks like the option above only creates the link on the HTML page to download the pdf version of the book. However, a new pdf version of the book isn't automatically rendered when we click the render button. You have to specifically render a pdf version separately from an html version.

We may want to think about creating an internal file with notes and a chunk that renders html and pdf when executed.

~~~
```{bash}
#| eval: false
quarto render
```
~~~

Even easier, you can do this with a native R code chunk.

- The input argument: The input file or project directory to be rendered (defaults to rendering the project in the current working directory).
- The output_format argument: Target output format (defaults to "html"). The option "all" will render all formats defined within the file or project.

~~~
```{r}
#| Render with R
#| eval: false
quarto::quarto_render(output_format = "all")
```
~~~

10 Publishing

This chapter is about getting the book files on GitHub and creating an HTML version of the book for public consumption. I'm starting with GitHub Pages because it seems like it should be the lowest hanging fruit.

10.1 GitHub repository

The first step to publishing a book online is to put it into a GitHub repository. Originally, I started my book project by creating a new project, clicking new directory, then Quarto book. After creating the book project, I created a repository for it in GitHub and then tried to use GitHub's on-screen instructions to push the book files to the repo. However, I kept getting errors and wasn't ever able to make it work. I feel like this shouldn't be the case and was probably just some weird one-off. So, don't give up on that process just yet.

What I eventually got to work was this process. - I created the repo on GitHub (with nothing but a README file) first. - I cloned the repo to my computer. - I clicked new project > existing directory. - The downside is that RStudio didn't give me the option to make it a Quarto book project. - In the terminal, following this guidance, I typed `quarto create-project test_quarto_book --type book`. - Make sure the terminal is set to the project's directory. - That added all the built-in book stuff, but in a folder inside the current folder. You can move all of the files you're interested in over if you want. Since I already had the files I had been previously working with (but wasn't able to push to GitHub), I moved them over and worked with them.

10.2 GitHub Pages

Useful websites:

- https://quarto.org/docs/publishing/github-pages.html

That website discusses three methods for publishing the book with GitHub Pages.

1. Render to Docs. The easiest and most straightforward. However, checking in the `_book` directory makes for messy diffs.

2. The publish command. Requires a little set up on the front end, but gives more control.
3. GitHub Action. You might prefer this if you want execution and/or rendering to be automatically triggered from commits. However, it seems like the most complicated option.

I'm going with option 2 for now.

10.3 Netlify

Useful websites:

- https://quarto.org/docs/publishing/netlify.html

Part V

Built-in Chapters

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

A Example appendix

This is just an example appendix.