Scientific Blogging with R and Blogdown

Aiden Huffman

July 19, 2023

Setup

Blogdown is an extension of Bookdown which allowed individuals to produce books using R and RStudio. The resulting PDF documents combined text, mathematics and code to produce full and comprehensive introductions to a variety of subjects. Blogdown attempts to solve the problem of how to create websites which also contain these components. The official book for Blogdown was released in 2018.

To get started, you will need to make sure you have a copy of R and RStudio. Afterwards, you will want to start up RStudio and install the Blogdown package. Blogdown uses Hugo to construct static websites. Fortunately, Hugo will be included when we run the following command:

```
install.packages('blogdown')
```

After installing Blogdown, we can use RStudio to start a new blogdown project:

File >> New Project >> Website using Blogdown

We will need to choose a name for the directory that contains our project, but otherwise leave the options unchanged for now. Once you become more comfortable, you will be able to fill in several options, pick a theme and indicate where you would like to store the website locally. Take a moment to look over the Blogdown and Bookdown documentation. To ensure the website works on the servers provided by the University of Waterloo, we will need to update the newly created configuration files to use relative URLs. Depending on the configuration file format for your theme, add or edit the the following property:

config.yaml

```
baseurl: /~<UWuserid>
relativeURLs: true
```

config.toml

```
baseurl = /~<UWuserid>
relativeURLs = true
```

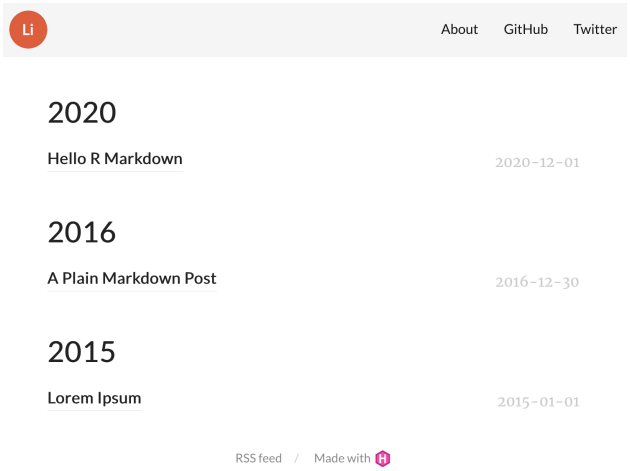

Figure 1: The home page of the template website generated by Blogdown with the default settings.

Next we should run the following commands:

```
blogdown::build_site(build_rmd = FALSE)
blogdown::serve_site()
```

This will produce the website locally and host it on your device; while the *build_rmd* option tells Blogdown whether or not to build RMarkdown documents. By running these commands after any major changes we can see how the website will look when when it is ultimately hosted at the University of Waterloo. This will also give us a chance to debug before the website is live. Furthermore, the Blogdown documentation presents additional options which prevent it from recompiling RMarkdown posts that have not changed. This is particularly useful for blogs with many RMarkdown posts, or for posts with large compilation times.

After running *blogdown::build_site()*, Blogdown will generate some folders in our project. The most important one is the public folder which stores the generated website; see Figure 2. We will use github to track changes to this new folder and sync them with the University of Waterloo servers. Switching to the terminal window in RStudio, we execute the following commands:

```
cd public # Enter public folder

# Build github repo
git init
git add --all
git commit -m "Initial commit"
```

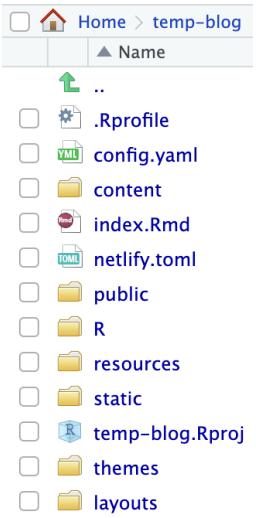

Figure 2: File structure after build.

This initializes a local git repo on your device. Next we need to connect the local repository to the University. The instructions to get set up can be found here. We will assume some familiarity with git and Github and that you are able to make the remote repository on your own. Once this is finished, link the remote repository to the local one by returning to RStudio and running the following commands:

```
git remote add origin <url-to-repo>
git push --set-upstream origin main
```

We can now pull the repository to the University of Waterloo server:

```
ssh <UWuserid>@linux.student.math.uwaterloo.ca
mkdir public_html
chmod 2755 public_html # Restrict write access to owner
cd public_html
git clone https://github.com/<username>/<repo-name>.git .
chmod -R 700 .git # Restrict access to .git to owner
```

You should see your website at `student.math.uwaterloo.ca/~<UWuserid>`. Moreover, the website should look identical to the one we built and served locally. At this point there are plenty of free themes available through Hugo. Not everything will work out of the box, so consider this the opportunity to experiment.

Making Content

A new post can be generated with *blogdown::new_post()*. We can then edit the file type and begin to put our content into the post. The extension for an RMarkdown document is .rmd; while, for a Markdown document we use .md. RMarkdown documents allow us to combine Markdown with code and mathematics. As mentioned, when we run *blogdown::build_site(build_rmd = TRUE)*, Blogdown will turn the RMarkdown documents into something that Hugo can serve as a webpage, as well as run all of the code in the document from scratch. By combining mathematical content with the large number of data analysis and visualization suites available in R or Python we can easily convey meaningful content which is informative and aesthetically pleasing. We provide a list of links which may be helpful on your journey:

- RMarkdown
- Blogdown
- Reticulate (Python in R)
- Tidyverse (Data manipulation and visualization suite)
- kableExtra (Fancy tables)
- DT (Paged tables)
- data.table (High speed data tables)

RMarkdown uses triple (```)'s to denote a block of code, while mathematical content can be entered using the usual LaTeX syntax. Finally, we can use Markdown syntax to configure the output with headings, links, images, etc. We provide an example of RMarkdown code for a posting in Figure 3 and the resulting web page in Figure 4.

LaTeX and Code

````
`r inline code`

```{r}
# A code block
```

$ inline math $

$$
display math
$$
````

Markdown

```
Heading             # H1
                    ## H2
                    ### H3

Bold                **bold text**

Italic              *italicized text*

Blockquote          > blockquote

Ordered List        1. First item
                    2. Second item
                    3. Third item

Unordered List      - First item
                    - Second item
                    - Third item

Code                `code`

Horizontal Rule     ---

Link                [title](https://www.example.com)

Image               ![alt text](image.jpg)
```

````
Scientific blogging used to be a particularly difficult task. It was difficult to produce
figures and videos, let alone include mathematical content. For instance, if we would like
to discuss something like finding the  eigenvalues of a $2\times 2$ matrix, this is not
easily done without MathJax or KaTeX. Fortunately, the R-package `blogdown` in combination
with a good theme can handle all of this for us:
$$
\begin{align}
\text{det}\left(\lambda I - \begin{bmatrix}
a_{11} & a_{12}\\
a_{21} & a_{22}
\end{bmatrix}\right) &= (\lambda-a_{11})(\lambda-a_{22})
- a_{12}a_{22}\\[1ex] &= \lambda^2 - (a_{11}+a_{22})\lambda
+ a_{11}a_{22} - a_{12}a_{22}\\[1ex] &= \lambda^2
- \text{Tr}(A)\lambda + \text{det}(A)\\[1ex]
\lambda &= \frac{\text{Tr}(A)\pm\sqrt{\text{Tr}(A)^2-4\text{det}(A)}}{2}
\end{align}
$$
We can also present a snippets of code. Below we show how a pipeline, `%>%` from the dplyr
package in tidyverse, can be used to pass data into ggplot.

```{r, echo=TRUE}
library(dplyr)
library(ggplot)
x = data.frame(value=rnorm(10001))
x %>% ggplot(aes(x=value)) + geom_density()
```

Setting `echo=FALSE` in a code chunk will hide the block but still show its results. In
fact, there are many options we can use in order to configure what is actually displayed.
The RMarkdown Cookbook is a good resource for this. We can even use Python inside RMarkdown
with the `reticulate` package

```{r}
library(reticulate)
# Switch to Python and load the Anaconda environment general
use_python("~/opt/miniconda3/envs/general/bin/python")
```

```{python, echo=TRUE, results='hide', fig.keep='all'}
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
x = np.linspace(-np.pi, np.pi, 101)
plt.plot(x, np.sin(x), color='tab:blue')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.title('$y = \\sin(x)$')
plt.show()
```
````

Figure 3: Sample RMarkdown code

Making a scientific blog at UW

2 min read

2023-03-30

Scientific blogging used to be a particularly difficult task. It was difficult to produce figures and videos, let alone include mathematical content. For instance, if we would like to discuss something like finding the eigenvalues of a 2×2 matrix, this is not easily done without MathJax or KaTeX. Fortunately, the R-package `blogdown` in combination with a good theme can handle all of this for us:

$$\begin{aligned} \det\left(\lambda I - \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) &= (\lambda - a_{11})(\lambda - a_{22}) - a_{12}a_{22} \\ &= \lambda^2 - (a_{11} + a_{22})\lambda + a_{11}a_{22} - a_{12}a_{22} \\ &= \lambda^2 - \mathrm{Tr}(A)\lambda + \det(A) \end{aligned}$$

is zero only if

$$\lambda = \frac{\mathrm{Tr}(A) \pm \sqrt{\mathrm{Tr}(A)^2 - 4\det(A)}}{2}$$

We can also present snippets of code, below we show how a pipeline, `%>%`, can be used to pass data into ggplot.

Figure 4: The first part of the web page generated by the RMarkdown code in figure 3