PharmaSUG 2021 - Paper AP-106

Bookdown and Blogdown: Deep dive of the R packages and components needed to create documentation and websites.

Ben Straub, GlaxoSmithKline

ABSTRACT

This is a companion paper to the paper **Bookdown and Blogdown: Using R packages to document and communicate new processes to Clinical Programming** from the **Strategic Implementation** section. Here I will focus on some of the core programming components of building documentation and websites from the packages bookdown and blogdown. The reader will be given a detailed walked through of some of the core code that can help create simple documentation or a website. A public GitHub Repository and website have been established that uses both packages and will be available for the foreseeable future. Most of the example code discussed in this paper are housed in the GitHub Repository Deployment and use of these products is in the context of clinical programming but could be modified to fit any domain.

INTRODUCTION

As clinical programmers, we find ourselves needing to document our processes and deliver key results to our stakeholders at an ever-quickening pace. Building extensive guidance documentation, interactive dashboards or websites can be a time and resource consuming process. A clinical programmer faced with these constraints might choose to enlist others who are skilled in the arts of web, dashboard or document building and find themselves on the sidelines. But not so fast! The authors behind the R packages bookdown and blogdown have you in mind! These packages make it easy to take code, results, documentation, dashboards and so much more and bundle them up into interactive documentation and websites to present to your stakeholders. Let's get off the bench clinical programmers!

This paper assumes the reader has some experience with R and RStudio and the use of R packages. I try and make every effort to not hand-wave away things, but for the sake of brevity we will sometimes need to embrace hand-waving. Please note that there are a lot of resources available on using bookdown and blogdown. My paper is set in the context of clinical programming and so has a more specific use case while existing resources take a more general approach. This paper also has an associated GitHub Repo with links to example sites that I hope you will explore and use. Readers will need to have some basic understanding of GitHub to download the materials discussed in this paper.

OVERVIEW OF THE IMPLEMENTATION PROCESS

My deep-dive discussions on using bookdown and blogdown are both going to implement a similar process for getting each set up and running. The next few sections pertain to both packages and I will quickly discuss the following:

- RMarkdown - the workhouse of these packages
- Downloading Materials - not necessary to read paper, but helpful!
- Installing Packages
- Creating the bookdown or website locally
- Deploying the Site

RMarkdown - the workhouse of bookdown and blogdown

RMarkdown files will form the backbone for most of what we want to do here with bookdown and blogdown. Before we dive too deep into those, let's take a quick detour and look at an RMarkdown file

(.Rmd) . As its namesakes implies, it contains R and Markdown together in one file ready to be executed! Once you run the file, RMarkdown makes use of two packages, knitr and Pandoc to generate a new file output. knitr executes the R code in the file, while Pandoc helps to render the output into the appropriate format, which could be a docx, html, pdf and a few others. **Fun fact: This document you are reading was produced using RMarkdown.**

Using RStudio you can create an .Rmd file from the menu by doing File -> New File -> R Markdown. Once you work through a few prompts you will have a basic template set up for you. The template has three important pieces to it, which will play key roles in our bookdown and blogdown explorations: the YAML, the Syntax/Text and the Code Chunks.

The YAML - YAML Ain't Markup Language

At the top of a new .Rmd you will something like the following:

```
---
title: "Pharmsug Demo"
author: "Ben Straub"
date: "3/11/2021"
output: html_document
---
```

This is the YAML and it contains the metadata for your file. The three dashes --- are extremely important in how they are placed in the file, an extra space in front of them can cause you a lot of pain! Here the YAML lets you declare title, author, and date, but also allows you to control types of output. More options are available, and we will make great use of them later. For example, additional options in a YAML will allow us to create a table of contents in bookdown and in blogdown create custom tags and categories for blog posts.

The Syntax or Text

The syntax used within an .Rmd file is Markdown. The options are vast for formatting text within your document. I highly recommend RStudio's Guide on RMarkdown - - linked in the Recommended Reading. However, the key markdown pieces to note for our purposes are:

- # for section headers
- {} anchors for internal links,
- *, -, + for creating lists
- `` backticks for code highlighting.

I make use of these extensively, so be aware, but the possibilities are many for Markdown, I just scratch the surface.

The Code Chunks

The Code Chunks are where the R code is executed. At the start of the Code Chunk you will see the ```{r}. Within the {} we will make use of several tricks to suppress, execute and hide code, messages, warnings as well as customize images. There are over 50 code chunk options, but I only make use of a few. Please take a moment to note:

- echo - Code is printed or not printed
- eval - Code is evaluated when Chunk is run
- message - Functions and packages messages can be suppressed
- comment - Change how results are displayed

Here is the YAML example from above. I inserted it into a Code Chunk so it would format properly in the Word Document. Using echo=TRUE I tell RMarkdown to print out the lines of code but using eval=FALSE to tell R not to execute the code - as I would get an error.

```
```{r, echo = TRUE, eval=FALSE}
---
title: "Pharmsug Demo"
author: "Ben Straub"
date: "3/11/2021"
output: html_document
---
```
```

Okay! We quickly reviewed the YAML, Syntax options and Code Chunks within an RMarkdown file. Most of this will be glossed over in the coming storm but take heed that its referenced here and knowing some basics will help us ease into using bookdown and blogdown.

Installing the Packages

If you are going to follow along or have any interest in using these resources, then you are going to need to install the packages. To do that just use install.pacakges and then use library(pkg_name) so R knows that you want to start using those packages. You only need to install.pacakge once, but every time you fire up a new R session you will need to run the library command.

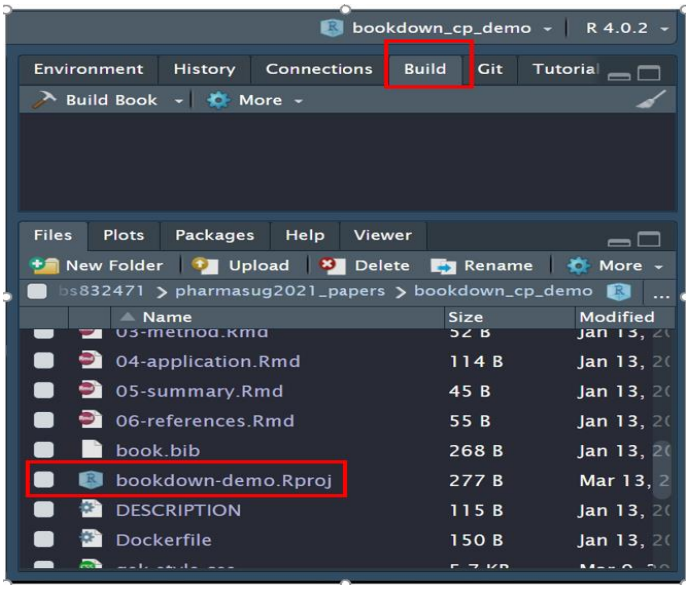
```
install.packages("bookdown")
install.packages("blogdown")

library(bookdown)
library(blogdown)
```

Downloading Materials

As someone who learned these packages by messing with the code in projects of much smarter people, I strongly recommend downloading the materials to play around with them. You can access the repo here : https://github.com/bms63/cp_site_demo.

I highly recommend learning the basics of git and GitHub and using them. They are powerful and useful tools. To download materials and troubleshoot Rstudio and git, I recommend https://happygitwithr.com/. If you have downloaded everything, or peaking at the GitHub Repository, then I would like to draw your attention to the .Rproj file. This is a huge help in configuring the RStudio environment with a Build Tab.



Please note the .Rproj file, which converts your RStudio environment to help optimize building either the bookdown or blogdown sites and allows version control. These are very handy if you are working on multiple projects within RStudio. Whenever I discuss building the website, I'm pressing either the Build Book for the bookdown or using the code blogdown::serve_site(). At some point, in the last year or so, the ability to use the Build Website tab in RStudio has stopped working. The code blogdown::serve_site() will create the local version.

Deploying the Site

In my company we use RStudio Connect to deploy our materials. It is super user-friendly and easy to use. For the purposes of these two demo projects I make use of Netifly. I highly recommend Allison Hill's blog on publishing a site to Netifly - https://alison.rbind.io/post/new-year-new-blogdown/#step-5-publish-site . She writes in the context of using blogdown too! You can publish bookdown and blogdown sites to Netifly.

IMPLEMENTING A BOOKDOWN

To give some context to the use of bookdown, let's discuss a scenario. A group of Clinical Programmers (CPs) wishes to present to their department the proper way to use R for Stage 2 QC of Displays. Most programmers are SAS programmers and do not have the time to hunt down all the resources to use R. However, R is being pushed as a new language that must be adopted. Here the intrepid group of CPs gathers all their proper R code, SOPs, guidance and collates it into guidance documentation using bookdown. These CPs decide upon a small first version with 3 chapters on Getting Started, Use Cases, and doing Production/QC comparisons.

To make this happen, they will need to focus on the following files of the bookdown:

- index.Rmd: Front Page of the book
- _output_yml, _bookdown_yml, style.css, toc.css: Presentation of output
- 01-getting_started.Rmd: Chapter 1 of the book discussing using R in the Department
- 02-use_cases.Rmd: Chapter 2 Using R to do Stage 2 QC
- 03-df_cmpr.Rmd: Using diffdf to do dataframe comparisons

Please look at the public repo. It contains way more files, but I think these are the most important ones!

Building the Book

By clicking the Build Book tab, you are telling the bookdown to start building your book. It firsts merges all .Rmd files by the order of filenames, e.g., 01-getting_started.Rmd will appear before 02-use_cases.Rmd. Filenames that start with an underscore _ are skipped. If there exists an .Rmd file named index.Rmd, it will always be treated as the first file when merging all .Rmd files. The reason for this special treatment is that the HTML file index.html to be generated from index.Rmd is usually the default index file when you view a website. It also tells bookdown to grabs formatting and takes a look at _yml files to customize and set up appropriately.

The index.Rmd file

```
---
title: "R Programming Guidance Documentation"
author: "Ben Straub"
date: "`r Sys.Date()`"
site: bookdown::bookdown_site
output: bookdown::gitbook
### truncated ###
---
```

The index.Rmd file allows you to declare the title of the project. Remember the YAML from the RMarkdown discussion. Well, here it is at the top of our index.Rmd file. This is where I declare the title of my book, my name, the date as well as how I want the book to be outputted. There are a lot of other possible options, but we will only discuss the first three. Here I am calling my book **R Programming**

Guidance Documentation, listing out myself, **Ben Straub**, as author and using R's built in Sys.Date to create the date. There are several other inputs you can play with, but those are for advanced users. You can choose different ways to output the book as well. Please note, 01-getting_started.Rmd, 02-use_cases.Rmd, 03-df_cmpr.Rmd do not have YAMLs at the top. Here the index.Rmd asks as an almost global YAML

_output.yml, _bookdown.yml, style.css, toc.css

Below is the code for the _bookdown.yml. file. Note, that it starts with a _ and so is not compiled into the book. Here, I can declare the file name for the book, customize the language, chapter heading and tell it to remove merge files once building is completed. A simple but very important of the code base.

```
book_filename: "bookdown-cp-demo"
language:
  ui:
    chapter_name: "Chapter "
delete_merged_file: true
```

Below is only the first 10 lines of the _output.yml file. This is where you customize the look of your book with css files. Very important for branding! Here I also place an image to represent pharmasug above the TOC as well as link out to the associated website for this paper. I have created a pharmasug theme within the css files. You can modify the [style.css, toc.css] files to include images of your organization and style the colors appropriately. Images referenced will need to placed in the img folder. The Github Repo has the files, [style.css, toc.css], and shows where to modify to get colors appropriate to your organization.

```
bookdown::gitbook:
  css: [style.css, toc.css]
  config:
    toc:
      before: |
        <li><center><img src="img/tempsnip.png" alt="logo" width="30%"
height="30%"style="margin: 0 0 0 0"></center></li>
      after: |
        <li><a href="https://bookdown.org/yihui/bookdown/"
target="blank">Bookdown Website</a></li>
    edit: https://github.com/rstudio/bookdown-demo/edit/master/%s
    download: ["pdf", "epub"]
```

01-getting_started.Rmd, 02-use-cases.Rmd, 03-df-cmpr.Rmd

I highly recommend viewing the public site and GitHub Repo for this bookdown. As this is a bit of a whirlwind tour of these three files. For each of these files I make use of the #, ## and ###. These are referenced within the Table of Contents as well as in the section. Below is the top of the page Chapter 1 in Rmarkdown. Notice the {intro}, which is an anchor and useful for linking internally within the document and allowing the ability to link externally to a specific page. Below the header is just some example text from the file.

```
# Getting Started with R {#intro}

Getting started with R at Pharma Inc.  can be intimidating.  Those within the
R Implementation group have curated a select group of resources to help ease
into the use R.  We have also created several use case example of creating
displays within our company and demonstrate how to comparison
```

In 02-use-cases.Rmd, I have developed some R code to create a table from the CDISC Pilot Study Data and displayed it with several interactive table using function from the kableExtra package. Below is the code to load the packages and reading in the SAS datasets .

```
# Use Cases {#use-cases}

# Load Packages
library(diffdf)
library(dplyr)
library(tidyr)
library(haven)
# Access the Adam and dddata folder
# Load in the datasets from HARP
adae_orig <- read_sas(adamdata, adae)
adsl_orig <- read_sas(adamdata, adsl)
```

In 03-df-cmpr.Rmd I showcase how to do a Proc Compare using the diffdf package. Here we compare two files and output a .lst file, which is essentially a text file.

```
# Using diffdf for Comparison {#diffdf}

diffdf(t_ae_sum, q_t_ae_sum_tsp,
       keys = c("AESOC", "summaryLevel", "AEDECOD"),
       file = "qc_t_ae_sum.lst")
```

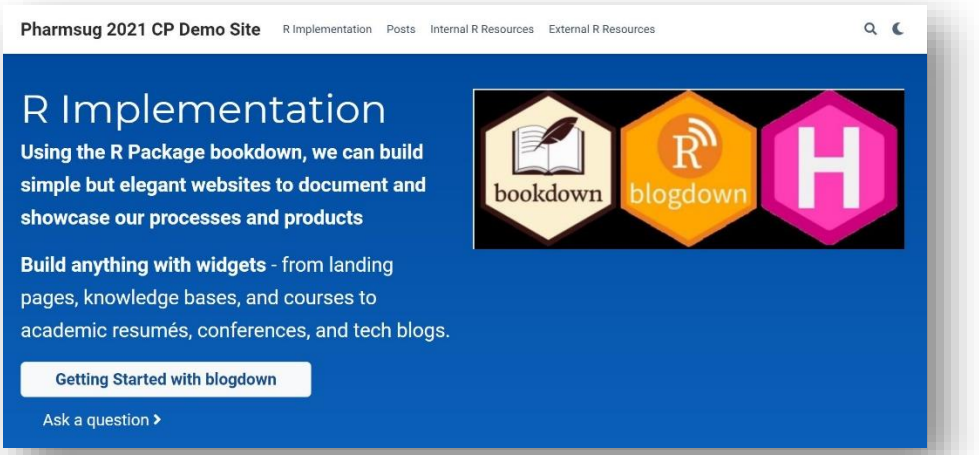
While the Clinical Programmer has successfully deployed her guidance document, she is asked about other resources internally and externally. She could continue to link into this document, but it might start to not fit the scope of Stage 2 QC. However, she knows of another package called blogdown to build a simple website to link all her company's stuff together!

IMPLEMENTING A BLOGDOWN

Blogdown is an awesome R package that lets you create websites using Hugo and R. I will go over Hugo soon, but why create a website using R? Why not just create a website from another service? To me, it's a steppingstone from doing data analytics to delivering those data analytics, documentation and guidance to your stakeholders. A website is a great way to do this but getting set up in another software and environment is tedious and time consuming. I'm also interested in minimizing the about of environments that I must navigate through to deliver something to my stakeholders. Being able to stay in one platform, ala RStudio, has many advantages – e.g. keeping all my files in one place, easy access to documentation, one username/password, etc. Finally, as my last selling point, I like to think that blogdown plays as an intermediary for you between two different worlds - data analytics and web development.

Below you will find an image of the Demo Site I built using blogdown. The code is available on the GitHub Repo and the website is publicly available. You can find the website linked in the ReadMe for the Github Repo. I tried to streamline and purge as much content from this demo site/code base so an enterprising clinical programmer could grab the repo code and quickly make use of it. Again, the end goal is to give you some code and an idea of where to go with this package.

The following code walkthroughs are all focused on the example site. Please note that blogdown and Hugo's websites are vast! You can make a much more dynamic website with their tools. From my experiences, the sheer volume of information and code available was a bit too much. I hope this streamline website and code base provides you with an easier jumping off point



Hugo and themes

Hugo is the static site generator for blogdown. Hugo has 100s of themes to choose from. The choice of themes can help differentiate the intention of your site. For example, there is a documentation theme, which could help present your documentation in a concise and pointed way. However, maybe you want to showcase code that is being developed for a specific use or have lots of external and internal resources that fit general themes, then a blog theme might be a better choice. A great feature of the Hugo and blogdown combo is that each have similar folder and file structures, but their presentation of materials can be very different. For my purpose, I have chosen the Academic theme and trimmed it down quite a bit for a fictitious clinical programming department's internal website!

Again, I highly recommend checking out the public website for this paper. The bookdown and the website are linked together. You can grab the materials from the Github Repo or just look at them. If not, then you can use the following command and blogdown will generate a similar file structure in your current working directory. Be prepared it creates a lot of files and folders.

```
library(blogdown)
new_site(theme = "wowchemy/starter-academic")
```

I am going to peek at the following files and folders within the website that has been created by blogdown: config.toml, config folder, content folder, post folder and an .Rmd blog post file Most of the files and folders will be aptly name with their purpose – for example the post folder will contain the blog posts that you make. However, sometimes the folders and files will be a bit confusing depending on which Hugo theme you choose – for example the academic theme has both a config.toml file and a config folder! Past blogdown/Hugo composition had everything regarding the configuration of the website in the config.toml, but recent updates have been seeing more things being split up into smaller files. It makes sense, just sometimes hard to find what you are looking for the first time!

config.toml

I think of the congfig.toml file as the global command center for the website. It has a lot of possibilities, but for now I just want to draw your attention to the title and the baseurl. The title is what people will see when they click on the link to your website. Here I have chosen **Pharmasug 2021 CP Demo Site**. You can see it at the top, if you have the website open. The baseurl is also important. You need to insert a / if you want the site to go live later.

```
---
 theme: "starter-academic"
```

```
title: Pharmsug 2021 CP Demo Site # Website name
baseurl: '/' # Website URL
copyright: '' # Footer text, e.g. '© {year} Me'

defaultContentLanguage: en
hasCJKLanguage: false
defaultContentLanguageInSubdir: false
removePathAccents: true

module:
  imports:
    - path: github.com/wowchemy/wowchemy-Hugo-modules/wowchemy-cms
    - path: github.com/wowchemy/wowchemy-Hugo-modules/wowchemy

### Way more below
---
```

config folder - menus.yaml

Navigation is key to any website. You want your users to be able to quickly find the materials. For the example site, I have 4 sections that link to materials below the initial homepage or link internally to different pages. You can also link to external websites. The most important part of the menus.yaml file are the url and the weight. If you want to link to something on the initial home page then you use a # symbol and then the section of the site. If you want to link to something internally, then you need to use the name of the document with a / at the end of it. The weight controls the ordering of the links.

```
---
  main:
    - name: R Implementation
      url: '#about'
      weight: 10
    - name: Posts
      url: '#posts'
      weight: 20
    - name: Internal R Resources
      url: internal/
      weight: 30
    - name: External R Resources
      url: courses/
      weight: 40
---
```

The config folder also allows you to adjust other parameters in the website in the languages.yaml and params.yaml file. For example, you can date, time and contact information within the params.yaml file.

content folder

This is the meat of your website. Below you will find a representation of the folder structure for most of what is in the contents folder. There is a lot to explore here, but I am just going to focus on the posts folder as this is where the RMarkdown files for blogdown exist. However, within the academic Hugo theme you can make great use of all these folders and their files to really liven up your site.

```
# content/
# ├── admin
# ├── authors
# ├── courses
```

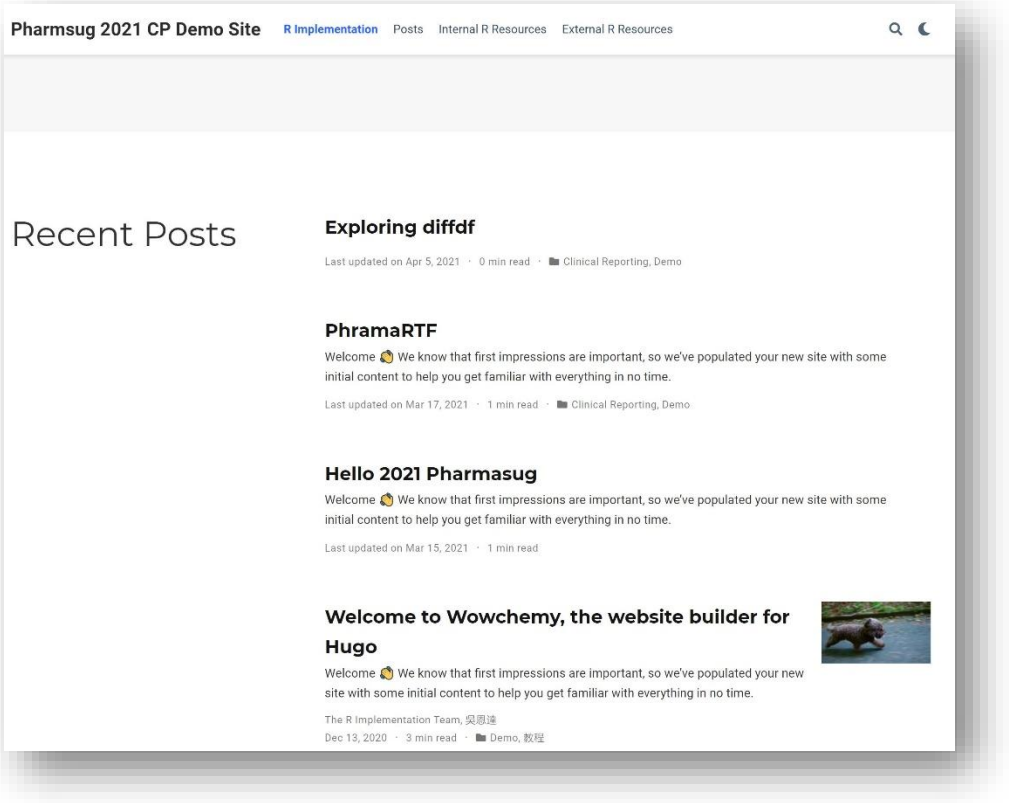
```
# ├── home
# ├── internal
# ├── post
# ├── project
# ├── publication
```

Making a blog post in RMarkdown

Where did the RMarkdown go? The last few sections have discussed exotic files such as .yaml, .toml and Hugo themes. Well have no worry! We are back to the RMarkdown. The advantage of using blogdown for deploying a website is it is an easy interface with R. Making a post to your new blogdown is super easy! To do that I recommend the following two ways: use the below code snippet or the blogdown Addin. Now we can make posts that can showcase our exploration of R!

```
blogdown::new_post(title = "Hello 2021 Pharmasug",
                   ext = '.Rmd',
                   subdir = "post")
```

Users coming to your site can scroll down to the Recent Posts section or click the Posts section at the top menu. Below is the snapshot of our “Hello 2021 Pharmasug” Post with a couple of other filler posts.



When you tell blogdown to create a new post for you, you are telling Hugo to create a page bundle. Each post gets its own bundle, or folder. Inside the post bundle is where all your static images, static data files like image and .csv files should go. For the moment, the “Hello 2021 Pharmasug” post is lacking on content, but the .Rmd file is where you would write your R Code and text.

```
# content/
#  ├── posts/
#      ├── 2021-03-15 – hello 2021-pharmasug
#          ├── index.en_files
#          ├── index.en.html

#          ├── index.en.Rmd
```

At the top of the index.en.Rmd file in your recently created post you find something very familiar. The YAML! The different parameters are self-explanatory except the slug. The slug is just for Hugo to be able to uniquely identify the blog post.

```
---
title: Hello 2021 Pharmasug
author: The R Imps
date: '2021-03-15'
slug: hello-2021-pharmasug
categories: []
### truncated ###
---
```

A whirlwind tour! Just to recap. We discussed how to create a new site with the Hugo academic theme and went over the core file and folder structure that was used to build the Demo Site. We also discussed how to create a blogdown post and tool a peek at its structure.

CONCLUSION

Bookdown and blogdown are powerful packages that can help clinical programmers deliver guidance, results, documentation and a whole lot more to your stakeholders. The authors behind these R packages have those in mind who are inexperience in the area of web development but have critical insights to deliver! I hoped that this paper provided you with some simple guidance and ideas on how to implement either a bookdown or blogdown website. If you can get comfortable with using RMarkdown's YAML, Syntax and Code Chunks then you are 90% on your way to mastering bookdown. Blogdown is more complicated and requires a bit more grit if it is your first experience with web development. However, the authors of blogdown and the creators of Hugo have really taken simplified the process for you.

RECCOMENDED READING

- BLOGDOWN: https://bookdown.org/yihui/blogdown/
- BOOKDOWN: https://bookdown.org/yihui/bookdown
- Anything written by Alison Hill: https://alison.rbind.io/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ben Straub
GlaxoSmithKline
1250 S. Collegeville Road
Collegeville, Pennsylvania, US, 19426-0989
Email: ben.x.straub@gsk.com