Website building

This page was last updated Mon Jan 06.
[Download this page as a PDF]

Learning Objectives

- Recognize the minimal requirements to build a website using R and Markdown.
- Build a basic level website using R and Markdown
- Publish this website to GitHub
- Become aware of how the `blogdown` R package can be used to create responsive websites
- Download and compile a starter website based on blogdown.

Collaborative notes: https://hackmd.io/@norcalbiostat/ichps2020_website_notes

---

Getting started: Basic Site

R Markdown websites like this one can be built with a minimum of two files: a `_site.yml` and an `index.Rmd`.

The `_site.yml` file controls the overall structure of the website, such as the order of the navigation bar and the color theme.

When these files are *rendered* into HTML files, they can be hosted on GitHub for free. Then after turning on a few settings in your GitHub repository (repo) these files can be viewed as a magnificent webpage.

We are going to employ the "GitHub first, R Studio second" approach.

1. Create a public Github repo.

Do together

- From your *Repository* page, click the green [NEW] button.

- (*Suggestion*): Name this repo something like `ichps_website` so you know it's a playground.
- keep it a PUBLIC repo
- YES initialize with a README

Click 'Create Repository' to finish, but keep this page open. We'll come back to it a few times.

2. Create a project in R Studio and clone your new repo into this project.

Do together

- Top right corner in R Studio click: *New Project –> Checkout from version control –> Git*
- Go back to GitHub, and click on the green [Clone or Download] button.
- Click the clipboard icon to copy the URL to your repo.

- Paste this URL into the *Repository URL* box in R Studio.
  - **(*Strongly Suggested*)**: Name this project the same as the repo name.
  - Subdirectory: Your path, your choice. Just don't put a repo inside a repo.
- Click *Create Project*

3. Create necessary files in R Studio.

Do together

- In the *Terminal* window (tab to the right of the *Console*), create the `_site.yml` and `index.Rmd` files using the `touch` command. Type the following commands in the Terminal one at a time, pressing Return or Enter after each line to execute.

  ```
  touch _site.yml
  touch index.rmd
  ```

  You can view these files in the *File* pane (lower left). You can open supported file types by clicking on the name of the file from this pane.

Do together

- Open both the `_site.yml` and `index.Rmd` files now.

4. Configuring the `_site.yml` file.

This is the file that controls your overall site's configuration. If you are familiar with R Markdown files, this is the same type of 'language' that is used in the header area to control the document structure.

YAML headers, and this file, are very particular about spacing and tabs. To avoid unnecessary grief we are going to copy/paste code from Chapter 10.5.1 of the bookdown website.

Do together

- Navigate to the bookdown website by clicking the link above.
- Mouse over the first code chunk in section 10.5.1, and click the *Copy to Clipboard* icon in the top right corner.
- Paste this code into your `_site.yml` file.
- Remove lines 7 and 8 since right now we don't have an "About" page.
- Give your website a name and title!
- Add the following line on a new line `2` (between `name` and `navbar`).
  - This ensures that all rendered HTML files are saved in the top level directory.

```
output_dir: "."
```

5. Configure your index page.

This is your landing page. That is, the first page people see when they go to your website. You can add a title to this page using a YAML header as is shown in the bookdown website, or you can use a pound sign `#` as a first level header to make this title.

Do together

- Create a title for your page using *either* a YAML header or a `#` sign. Using both is redundant.
- Write a sentence or two welcoming your viewers to your website.

6. Building & previewing your website locally.

Since we created the `.yml` file manually after creating the R Project, your project doesn't quite know yet that it has a website to build.

Do together

- Save your work and close down R studio.
- Navigate to your repository folder on your computer, and open the project by double clicking on the R project file icon (a cube with the letter R).
  - *Note: This is always how you should open your R Project files*

There should now be a **BUILD** tab in the top right pane.

- Click this tab, and then click the *Build Website* button.

An HTML file should now appear in your viewer pane.

- Click the 'Show in new window' button to see your site in a full browser window.

Notes, Tips and Comments -

- Clicking "Build Website" will render every `.md` and `.Rmd` file in your top level project folder into an HTML file.
  - This is important to keep in mind when creating a website with a lot of pages.
  - Code files in sub-folders will not be automatically rendered.
- You can re-render individual files by clicking 'Knit'.
- You can leave this browser window open as you work, refreshing the page to see new changes.
- There are some basic themes you can app

```
output:
  html_document:
    theme: readable
    highlight: pygments
```

Do together

- Try this now. Make a change to your `index.Rmd`, knit and refresh your browser window.

7. Push to GitHub & publish your website.

Let's get your test site up for the world to see.

Do together

- In the *Terminal* window, stage and commit your files to version control by executing the following commands:

```
git add -A
```

This will *stage* all files that have been changed and/or added. *Staging* is the process of adding a file to be tracked under version control.

```
git commit -m "first commit"
```

Every commit needs a message. Try to make it informative, yet brief.

```
git push
```

This will *push* your changes up to GitHub's servers online. You may have to enter your github username and password here. See happy git with R for help storing your github credentials. **Do together**

- Go back to your GitHub repo page and refresh - you should now see that your repo now contains the new code, and HTML files that you have been working on.
- Click *Settings* in the top right of the navigation bar in your repository.
  - Scroll down to the **GitHub Pages** section.
  - Under *Source*, select *Master branch.* (*Note: Chapter 3.3 of the bookdown manual has instructions for publishing something other than the top level folder for a repo, such as a* `public` *folder*)

After the page refreshes, you will see the URL to your website in at the top of GitHub pages section.

- Click on this link to see what your public facing website looks like!
- Copy this URL, we're going to paste it in the repo settings so it can be found easy.
- Go back to your main repo page by clicking on the name of your repo at the top of the GitHub page.
  - Click **Edit**
  - Add a short description, and paste the URL into the *Website* field.

8. Adding Content.

Let's add an example analysis project to showcase on our website.

Do together

- Go back to R Studio.
  - Create a new R Markdown file using *File -> New R Markdown* so we can get the example R Markdown content that includes a plot.
  - Save this file with a name such as `project.Rmd`
  - Knit this file to produce a HTML file.
- Add a link to the project HTML page to your `_site.yml` configuration file by adding a new `text` line. Be mindful about the spacing here.

```
- text: "My Projects"
  href: project.html
```

- Rebuild your website by clicking the *Build Website* button in the *Build* tab in the top right pane.
- View your changes locally, make a change if you see fit.
- Add your changes to version control from the Terminal:

```
git add -A
git commit -m "add project"
git push
```

- Refresh your live webpage on GitHub to see the results!

Examples

- Course webpage for an Applied Statistics class at CSU, Chico https://norcalbiostat.github.io/MATH315/index.html
- Project webpage https://norcalbiostat.github.io/chem_ss/
- Personal website: http://www.emilyzabor.com/index.html

Next phase: Blog Aware

The "simple" website that was built above can be extended and enhanced well enough with some CSS and Rmarkdown wizardry. However, sometimes you may want a little more of a 'modern', 'dynamic' or 'responsive' feel, or to include a blog in your website. Here are some examples:

- https://georgecushen.com/
- https://chicodatafest.netlify.com/
- http://datascience.csuchico.edu/

This is when we enter the realm of [Hugo], - the self proclaimed "most popular open-source static site generator".

Hugo is it's own language, and for advanced usage of Hugo-based websites, some understanding of how Hugo works and reading *those* docs may be required.

For now, we are just going to show you how to get started in this realm of fancy-ness using the R package `blogdown`. Then you can go break it later. The super official blogdown documentation is linked in the References section at the bottom of this page.

If you did not do so beforehand, install the `blogdown` package, and use it to install Hugo using `blogdown::install_hugo()`.

1. Create a new Github Repo and connect it to an R Project.

This is step 1 and 2 from the first part of this workshop.

2. Create a new site with a specified theme

The `blogdown` package will setup the necessary files, folder structures and theme files from a pre-specified them that is available on github. The example we will start with is a very simple site containing a few pages and a blog.

```
blogdown::new_site(theme='yihui/hugo-lithium')
```

This could take up to a minute to download all necessary files.

When it is done, your website will be rendered and appear in the bottom right viewer pane. Click the 'Show in new window' to open it in your full browser.

3. Explore configuration files.

Let's start by looking at the files that are contained in the root folder of this project.

- `index.Rmd`: Nothing really is there! This is one of the primary differences between this type of site building and the one we just finished. Content is handled in a very different manner. Hugo + Blogdown aim to help you focus on content creation and these programs put the pieces together for you.
- `config.toml`: Functions similar to the `_site.yml` file, you can think of TOML as having the same functionality as the YAML, but with different syntax. TOML uses a key-value pair of options, such as

```
title = "A Hugo website"
```

Areas of the website, like the menu (navbar), are controlled in sections that start with brackets and look like the following:

```
[[menu.main]]
    name = "About"
    url = "/about/"
```

Do together

- Remove the twitter link
- Change the GitHub link to point to your GitHub account
- Give your site a different title.

Were you able to see the results of each of the changes you made on the live site?

4. Explore and modify static content files.

Open the `content` folder. Right now there is only an `about.md` file. **Open this file**.

We see a familiar YAML header, and a body of text.

Do together

- Change the title and content.
- Save this file, go back to your local site and refresh the live preview to see the changes.

This folder is where we can add more static pages, such as a CV. Since we are not using any R code here, we can stick with the simpler markdown only file format.

Do together

- Copy the `about.md` file, rename this as `cv.md`.
- You can do this in R Studio by clicking the box to the left of the file name in the Files window, then clicking *More –> Copy*
- Change the title, add a few bullet points and save.
- Go back to the `config.toml` file, and add a new '`[[menu.main]]` section as shown below so that this link shows up in the navbar.

```
[[menu.main]]
    name = "CV"
    url = "/cv/"
```

5. Blog posts

On your newly updated website, clicking on the icon in the top left. This will take you back to the landing page. For this theme, the landing page is a blog (instead of a static page such as `About`). Clicking on one of these posts takes you to the page that contains the post itself.

The content files for posts are stored under `content/post/` folder. If you look there now you'll see some markdown (`.md`), some R Markdown (`.Rmd`) files, and their associated `.html` output files.

Creating new blog posts

- Option 1: Copy/paste/modify one of the example files already in this folder.
- Option 2: Use the R Studio Addins to setup the page structure for you.
  - **This is the one we're going to use**
  - Using the UI allows you to easily set catgories, tags and the like.

Do together

- Click *Addins –> New Post*
- This will likely require you to install and/or update new packages including (but not limited to) `fastmap`, `miniUI`, `htmltools`, `shiny`.
- If you run into seemingly endless package installation loops, restart RStudio and re-try.
- Give your post a title, author name, a category and a tag.
  - Categories and tags are optional, but for this example go ahead and set them.
- Select the `.Rmd` format for this example.
- Select Done

Let's do a quick analysis of the `cars` data set to add some simple content for display.

- Don't knit this file. Just save.

(Instructors note: Intro, summary, pander, sentence, plot, sentence, inline R, hide code)

6. Serve Site

We've been "cheating" a little so far in that our site has already been "Served". When we started the new site in step 2, not only did `blogdown` download the theme files and exmaple site, it also ran `blogdown::serve_site()`. This is what allows for the live preview.

At a later point when you want to come back to working on your website, opening the project file itself will not initiate a `serve_site()`. You will have to do this manually. Again, there is an **Addin** for this.

7. Push to GitHub.

Go ahead and save your work, add, commit, and push your content to your github account.

8. Hosting / Deploying your website.

If you have your own server, and you are familiar with website deployment then you can roll your own as you are used to. For those of us that have absolutely no idea how to host a website, a very good option that is very user friendly (and free) is Netlify.

You *could* use GitHub Pages as we did previously to host a Hugo based website, but there are some tweaks that you have to do to get GitHub to work with/around Jekyll in the way Blogdown/Hugo builds the sites. If you want to learn more about the why's, go read the docs at https://bookdown.org/yihui/blogdown/deployment.html

Do together (Time pending)

Closing comments

- Hugo has a TON of themes: https://themes.gohugo.io/ . Not all work with `blogdown`, and not all are easy to work with. Here is a blog post by Peter Baumgartner on important thoughts on choosing a theme to work with.
- The theme shown here has a very simple TOML configuration file. Each theme is different and unique. Reading the documentation (generally contained in the readme) is critical.

Read the Docs (References)

- Github
  - https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
  - https://swcarpentry.github.io/git-novice/
  - https://happygitwithr.com/
- Static Sites
  - R Markdown sites from R Studio https://rmarkdown.rstudio.com/lesson-13.html
  - Chapter 10.5 in the Blogdown book https://bookdown.org/yihui/rmarkdown/rmarkdown-site.html
- Blog aware sites
  - `blogdown`: Creating Websites with R Markdown https://bookdown.org/yihui/blogdown/ , https://github.com/rstudio/blogdown
  - Getting started with the Academic theme: https://sourcethemes.com/academic/docs/get-started/