CBW's Bookdown Template Documentation

Last Updated: 2024-12-16

Contents

Chapter 1

CBW's Bookdown Documentation

Welcome to CBW's documentation for creating a workshop website using Bookdown. Bookdown is an R package that is used to build books, and in our case, the websites hosting CBW's workshops!

You only need to know markdown and whatever coding language you will be using to learn bookdown!

Note, this is the documentation to create a workshop using *bookdown*. If **Jupyter Book** suits you better, see here.

If you don't know which one to use, click here to learn more!

Chapter 2

Getting Started

Bookdown is an open-source R package that helps write books and articles. We will be building our bookdown-based workshop websites using this, specifically the gitbook template. (This is just the name of the specific template style, you will be working in a workshop template that CBW has prepared for you!)

If you're ready to start making a workshop website in bookdown, let's setup your device (PC, laptop)!

First, let's explain installations.

2.1 Installation

Since bookdown is an R package, you will need R. Plus, our ideal IDE (integrated development environment i.e. the platform we will be working in) is RStudio.

1. **Download and install R** here. (You need R 3.6.0+ installed for RStudio.) Follow the instructions for your operating system (Linux/macOS/Windows).

   > Check: You can check if windows was installed properly on *macOS* by running the command `R` in terminal. On *windows*, –must test–.
   >
   > Note: We will not be using the R console, instead, we will be using RStudio!

2. **Download and install RStudio** here. Scroll down to find downloads for non-macOS.

Note: While installing, you may be asked whether to install the 32-bit or 62-bit RStudio version. Download the version that matches your PC.

3. **Install Git, if you don't already have it.** Git is a tool that will help us with version control when editing your workshop. Linux and macOS computers tend to have Git installed. Windows computers must install Git. However, make sure to double check if you already have Git, so that you don't have to install it again! Check if you have Git by running this command in terminal/command prompt:

```
git --version
```

```
## git version 2.39.5 (Apple Git-154)
```

If your output looks like "git version X.X.X …", you already have git! Move onto step 4!

However, if your output says "Git is not recognized" or a similar statement (such as the one provided below), you do not have Git, so you must install it as well.

```
'git' is not recognized as an internal or external command,
operable program or batch file.
```

Installing Git on macOS

- When you ran `git --version`, it will have prompted you to install Git. Follow these instructions.

Installing Git on Windows

- Go to the Git for Windows installer and download Git. Then, install it with all the default settings.

Click here for instructions on **installing Git on Linux**.

Double Check!

Check that your install worked! Re-run "git --verison" and check that you get your git version! (On Windows, this may look like "git version 2.47.1.windows.1").

Note: If you installed Git while having a Command Prompt/Windows PowerShell window open, close this window and open a new one to run "git --version". This acts as a refresher to Command Prompt/Windows PowerShell.

4. **Install the bookdown R package**: Open RStudio and in the console (in the bottom left window of RStudio) run the following command: `install.packages("bookdown")`.

    - **Download all the packages you will need!** Do this by running the following command:

    ```
    install.packages(c("magrittr", "stringi", "stringr", "reticulate", "tinytex","servr"))
    ```

    You may have to approve some downloads. Say "yes" and enter your password when necessary. This tells bookdown you are okay with downloading and installing these packages in their default locations. Finally, **run the following command**

    ```
    tinytex::install_tinytex()
    ```

We're ready to start working with CBW's bookdown workshop template now!

Go to the next step

Now that you're done the installations, it's time to go to the next step: creating a **new** workshop, if you're a RC, or seeing your deployed website (skip to step 4), if you're on the workshop team. The order of this documentation is provided in the sidebar on the left. You can also click the arrows below, to go to the previous or subsequent page!

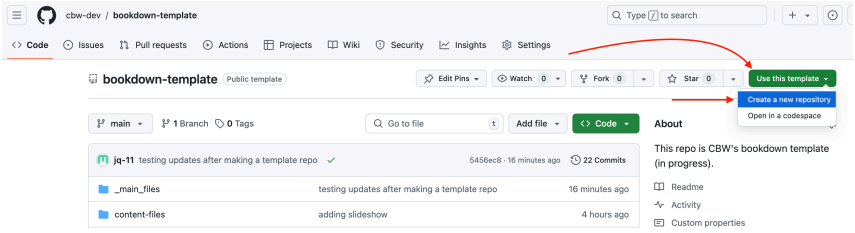
Chapter 3

New Workshop: Creation [RC] and Deployment

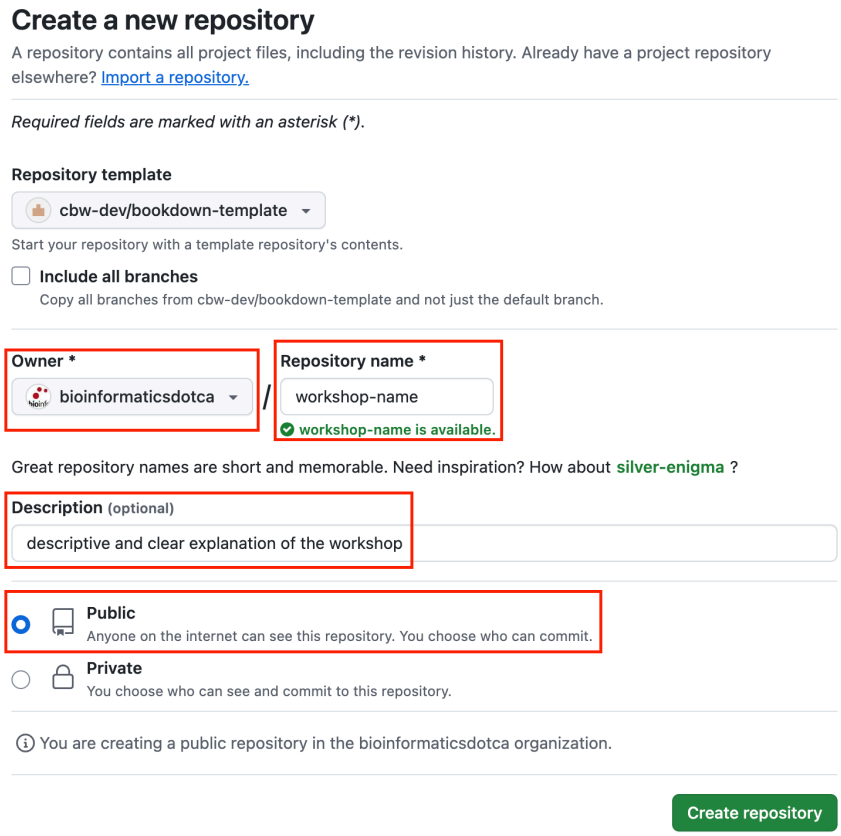
Certain aspects of the setup for workshops will be different depending on your role. Headers ending in "[RC]" are for Regional Coordinators. Headers without "[RC]" are assumed to be relevant to both RC and workshop teams.

3.1 Setup the Workshop [RC]

1. First, let's go to the bookdown template.
2. Click on the "Use this template" green button, which is to the left of the title of the repository "bookdown-template". Then, press the dropdown option: "Create a new repository", as seen below.



3. You will be brought to a "Create a new repository" page. Fill out the blanks as seen below. That is, change the owner to "bioinformatics.ca" [NOTE FOR TESTING PURPOSES: use cbw-dev], make it public, fill in the repository name and description according to CBW Guidelines. "Include all branches" does not need to be selected.



This may take a couple seconds to generate. After it loads, you will be brought to a new repository for the new workshop!

Now, let's turn this into a website - let's deploy!

3.1.1 Workshop Repo VS Workshop Website

Now, you have made a repository that holds what GitHub needs to make our website (the basic workshop template). Essentially, the template has already been configured so that the html files that make up our website go into a folder called `docs`. We need to tell GitHub to look at the `docs` folder to find our website files and make it available to see online (a.k.a deploy it). This is what we mean when we say CBW uses GitHub pages to deploy our website.

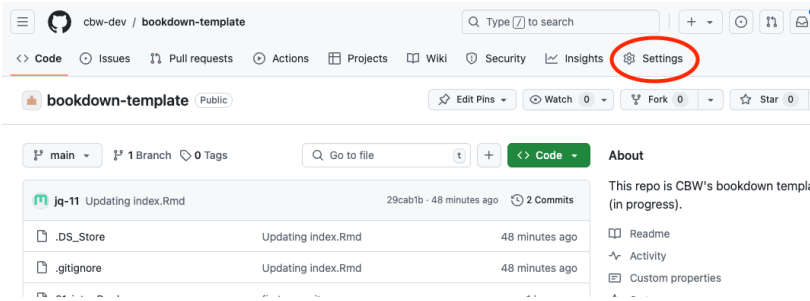
Distinction:

GitHub (ex. https://github.com/cbw-dev/bookdown-template) holds your repo, which has version control for all your files!

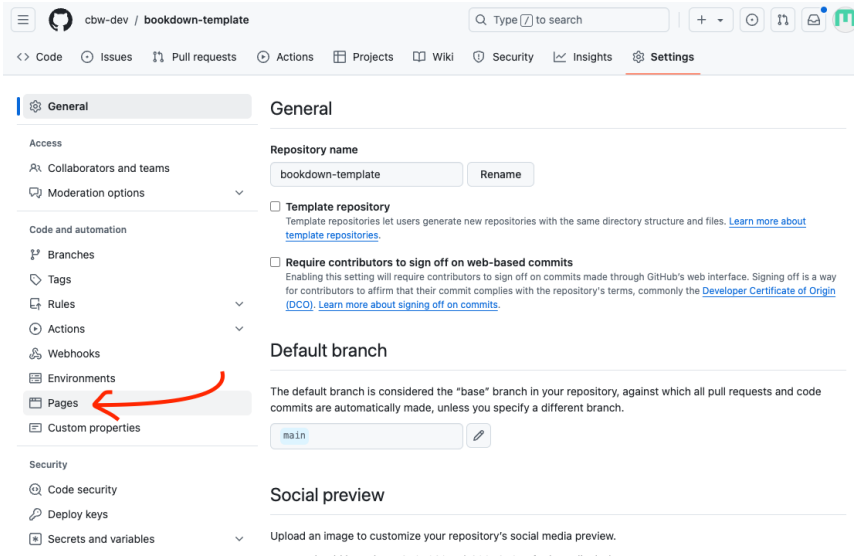
The deployed website (ex. https://cbw-dev.github.io/bookdown-template/) has the workshop online.

3.1.2 How to Deploy Your Workshop Website

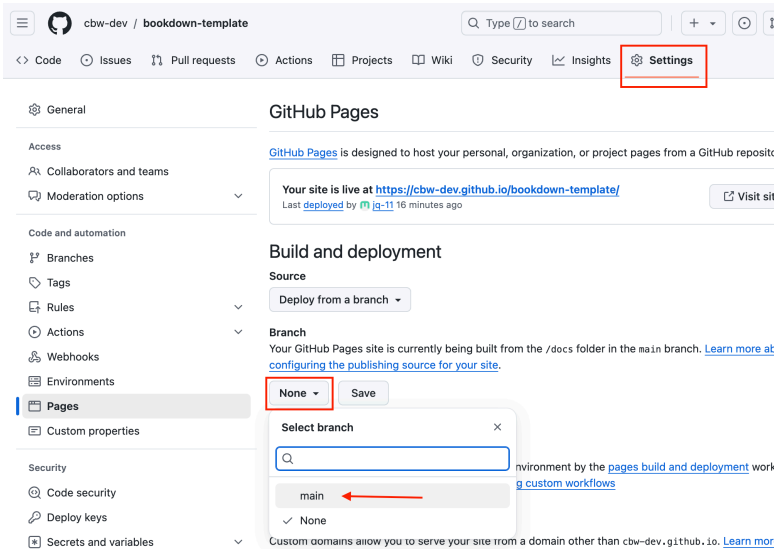
1. In the top navigation bar, select settings.



2. Then, go to the pages sidebar.

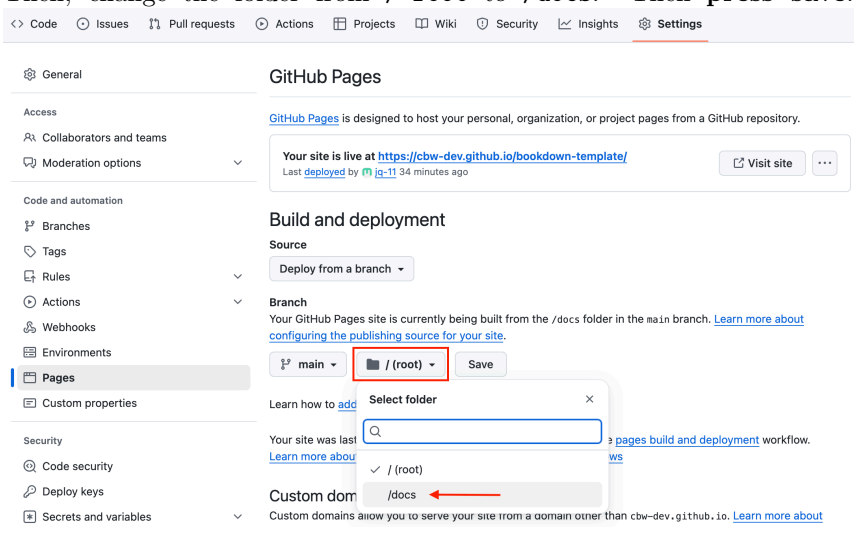


3. "Deploy from a branch" is already selected, which is what we want. We must change the branch from "none" to "main". Select the "None" drop-



down button and select "main".

4. Then, change the folder from `/ root` to `/docs`. Then **press save**.

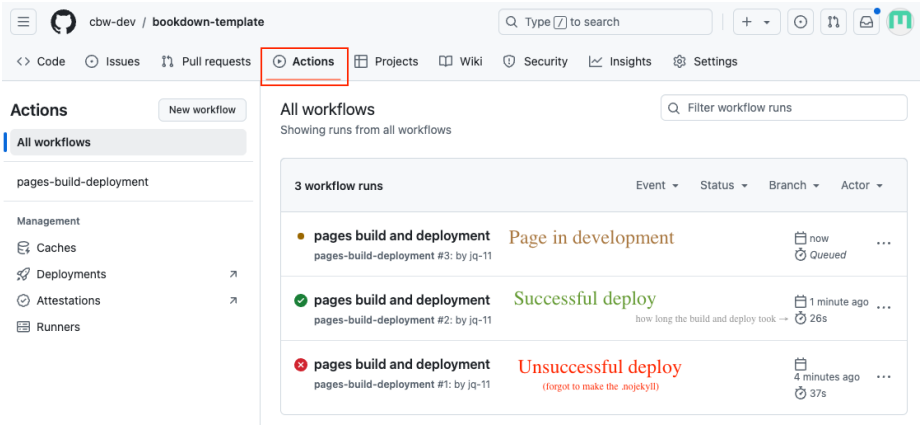


Great! Now we're waiting on the page to build and deploy, which should take less than a minute.

3.2 Check Your Deploy and See your Website!

To see updates, go to the **Actions** page (found along the top navigation bar. This will help you understand how the deploy is working, and if it succeeded or

failed.



You can click **pages build and deployment** for updates.

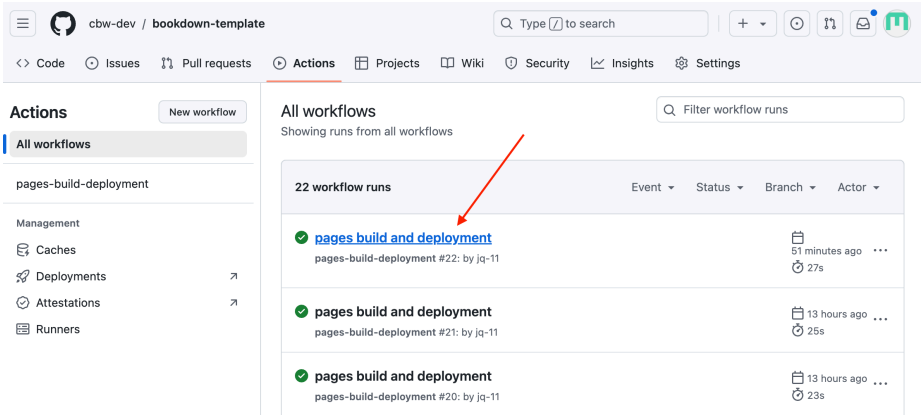
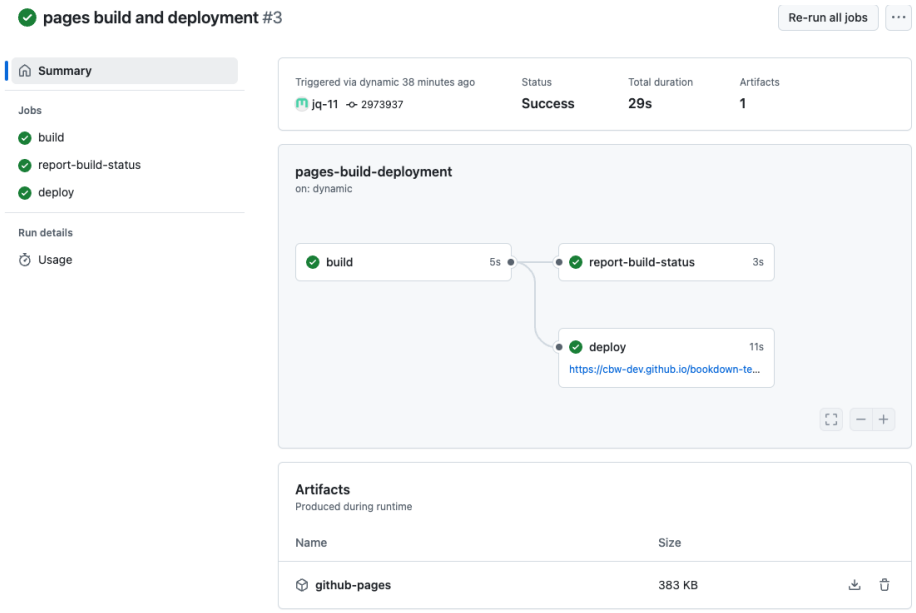


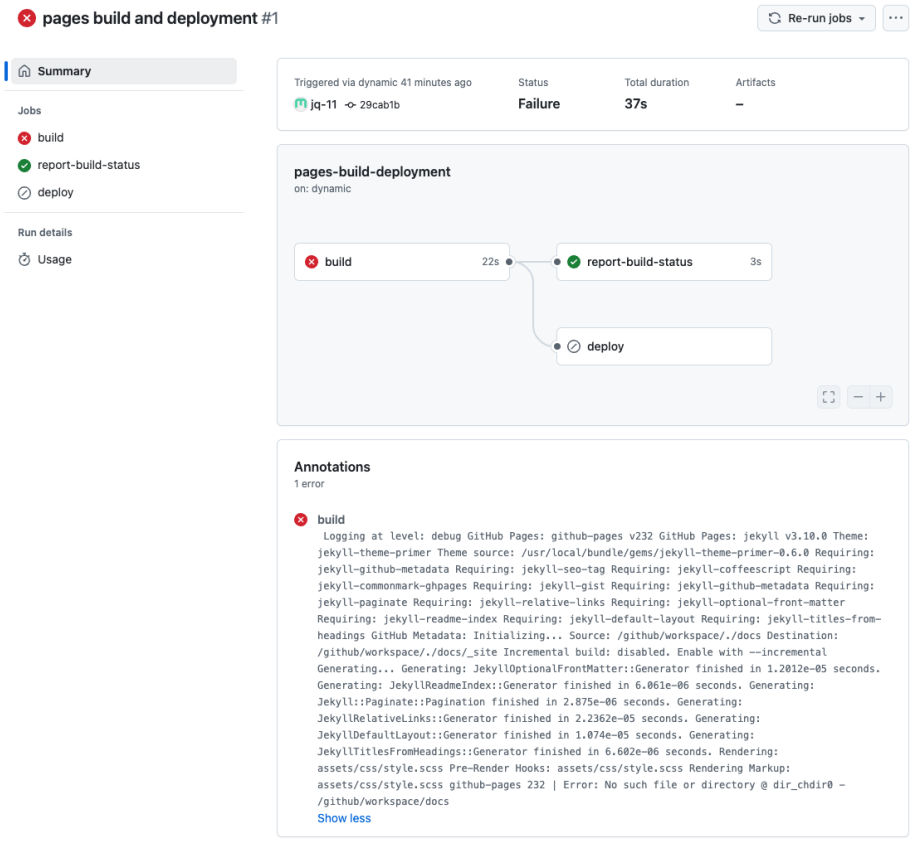
Figure 3.1: where to click for pages build and deployment information

A successful deploy will have a green checkmark next to it. You can inspect the 3 steps: build, report-build-status, deploy. Once it's done deploying, you can find the website at the link provided under "deploy"!

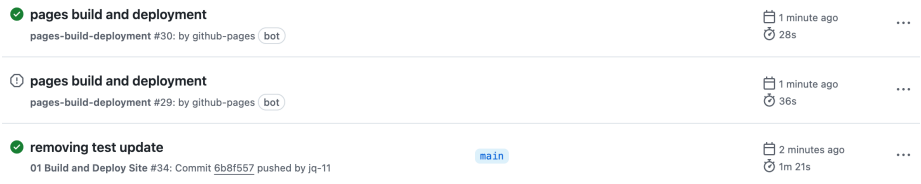


A failed deploy will have a red cross next to it. Clicking through the steps can help you determine what went wrong in the deploy.

Warning: A website can build properly, but may not deploy properly! It is a good idea to check after making big changes.



A Very Specific Build and Deployment Warning



This is a very specific (and unlikely) warning. It occurs when 1 deploy hasn't finished, but another deploy began. THIS IS NOT A CONCERN. This is a warning message you do not have to worry about!

3.3 Setting Up Team Access [RC?] (Nia will fill this in)

Chapter 4

Recurring Workshop: Copying [RC] and Deployment

Certain aspects of the setup for workshops will be different depending on your role. Headers ending in "[RC]" are for Regional Coordinators. Headers without "[RC]" are assumed to be relevant to both RC and workshop teams.

How's this different than the New Workshop: Creation [RC] and Deployment page?

You should be on this page if CBW already has a version of this workshop as a website, and you mainly wanted to reuse it (maybe with some edits).

4.1 Recreate the Workshop

We will be creating a copy the existing workshop and workshop website by git forking the repository that hosts the existing workshop website. (This is explained more thoroughly as you actually create your website!)

How to Git Fork

1. Go to the pre-existing workshop repository under the bioinformaticsdotca GitHub. For example, if we wanted to create a new workshop version of Analysis Using R 2024 (AUR 2024), we would go to this page:

2. Click "Fork" as shown below.
3. You will be brought to this page. You need to update the following parts: the owner (change to bioinformaticsdotca, as shown), the new repository name (follow CBW Guidelines) and the description. The fork will automatically have "Copy the `main` branch only", which it should be (do not deselect this). Then, click the green `fork` button in the bottom right, as highlighted below.
4. You will be brought to your new repository, which will host the updated version of the pre-existing workshop!

Now it's time to generate a website from your newly forked repository!

4.2 Deploying Your Workshop Website

Deploying your (turning these files into a) website is actually the exact same as deploying for a new workshop website.

Read all content starting from Workshop Repo VS Workshop Website on the **New Workshop**: Creation [RC] and Deployment page

4.3 Making Updates

Follow the Command Line, SSH Connection & Git Clone and So What Do These Files Mean? pages to figure out how to start editing your workshop. However, since it is a recurring workshop, you may not need to edit much.

If you are only editing the name, date and other small details , consider only editing the _output.yml and index.Rmd file.

See Formatting Your Content - Markdown for an explanation in how to edit the actual files, and more on Markdown syntax!

Chapter 5

Command Line, SSH Connection & Git Clone

If you're familiar with the command line and have already established a SSH connection, continue to git cloning your workshop template locally.

5.1 Introduction to the Command Line

This is for those who have no (or extremely little) experience with the command line.

Using the command line, you can use text commands to interact with your computer's operating system. For us, we will be using it to move around our folders and to git clone our workshop into our computer, so we can work on it using RStudio!

> Note:
>
> Do not be worried about using terminal, espeically git commands in terminal! Once we are all setup, we will never have to touch the terminal and write these commands again!

5.1.1 Terminal, Command Prompt and Windows PowerShell

We can use the command line using certain tools and applications. Terminal is a Unix-based (meaning Linux and macOS computer already have it) application that allows you to access the command line. Similarly, Command Prompt

(CMD) and Windows PowerShell give access to the command line on Windows computers. However, Terminal, Command Prompt and Windows PowerShell differ in what commands are accepted. The same commands we give to Terminal may not work in Command Prompt and/or Windows PowerShell.

> Note: Windows PowerShell tends to be more advanced than Command Prompt, and often can accept more commands that are accepted by Terminal than Command Prompt.

5.1.2 Common Commands (for us)

We won't need to know that many commands, but for easy navigation and understanding, here is what you (generally) need:

Note:

Commands are written below as headers, with an explanation provided beneath.

They follow the format: "[Linux Command] OR"[Windows Command]"

pwd OR echo %cd% "pwd" stands for "print working directory". It is a command that works in **Linux** and **Windows PowerShell**. The equivalent in **Command Prompt** is "echo %cd%". For example, below our output is where in my folders the current .Rmd file that makes up this website is:

```
pwd
```

```
## /Users/jqiu/Documents/bookdown-docs
```

ls OR dir "ls" is a command that works in **Linux** and **Windows PowerShell**. It's short for "list" and outputs all the files and folders in the directory (folder) you are currently in. (Note: this code only shows 4 files to save space!)

```
ls
```

```
## 01-getting-started.Rmd
## 02-git-clone-new.Rmd
## 03-git-fork-old.Rmd
## 04-terminal-and-git.Rmd
```

A similar command in **Command Prompt** is `dir` (short for "directory"), which also outputs the files and folders in your current directory (along with timestamps)!

cd "cd" stands for "change directory". The command produces no output, but it allows you to go to a different directory than the one you're currently in. For example,

```
pwd # recall: pwd tells us where we currently are
cd img # img is a folder in bookdown-docs
echo "now switching directories" # outputs the following string
pwd
```

```
## /Users/jqiu/Documents/bookdown-docs
## now switching directories
## /Users/jqiu/Documents/bookdown-docs/img
```

> **Tip**:Typing "cd" and then hitting the `tab` key will give you the available directories you can go to from where you are, or what you have currently typed in. If there is only one option, hitting `tab` will fill in your command with that option. (This works when typing in any file location into your command line, not only using "cd"). On macOS, the terminal will give you a list if there are multiple options. On Windows, both Command Prompt and Windows Powershell will fill in potential options, and you can hit tab multiple times untill you find your desired file destination.

File Location Shorthands When referring to file addresses, there are helpful shorthands! Here's a summary: - `.` = Current Directory - `..` = Parent Directory - `~` = Home Directory

Here's an example (recall, cd produces no output!):

```
pwd
echo -e # creates a line

echo "Current Directory Example"
cd .
pwd
echo -e

echo "Parent Directory Example"
cd ..
pwd
echo -e

echo "Home Directory Example"
```

```
cd ~
pwd
```

```
## /Users/jqiu/Documents/bookdown-docs
##
## Current Directory Example
## /Users/jqiu/Documents/bookdown-docs
##
## Parent Directory Example
## /Users/jqiu/Documents
##
## Home Directory Example
## /Users/jqiu
```

mkdir "mkdir [directory address]" stands for "make directory". Essentially, mkdir will make an empty directory (folder) at a specified location. For example: `mkdir test` would create a folder named "test" in our current directory. The following commands do the same thing. Note: `mkdir ./test` does the same thing.

rmdir "rmdir [directory address]" removes an *empty* directory. For example, `rmdir test` would delete the directory we just made!

rmdir -r OR rmdir /s "rmdir -r [directory address]" (Terminal & Windows PowerShell) and "rmdir /s [directory address]" removes a directory recursively, meaning it deletes all the contents of the folder as we all as the folder in itself. Be careful, you can not restore a directory you removed using "rmdir"!

Up [ ] and Down [ ] Arrows One of the most useful tips for using the command line is to use your up [↑] and down [↓] arrow keys. Using the up [↑] key gives you the previous commands you typed, and the down [↓] arrow returns you to your earlier commands.

5.2 Creating the SSH Connection

We need to create an SSH connection. You have already set this up if you have been git cloning, pulling from and pushing to GitHub. If you have, continue to git cloning. If you haven't, keep reading!

Essentially, we're doing these steps to update and receive updates from our GitHub repository, with security!

Follow the following 3 main steps. Each of these subheaders links to GitHub's official docs, if you would prefer to follow them instead! (Below is the simplified version of the instructions, if you've already been working with GitHub/SSH connection and want to make a new one, consider using the official docs.) The official docs may be more up-to-date.

If you do decide to use GitHub's docs, ***choose the page*** *(the different tabs are clickable at the top of the page) that matches your system (Mac/Windows/Linux).*

> Note: You can do these commands anywhere in your files. You do not need to be in your home directory.

Output VS Commands

In our instructions (and the GitHub docs), output is prefaced by a ">" sign.

In the GitHub instructions, commands start with "$" sign. In Linux, the terminal tends to give you a "$" to indicate where to run your command. Thus, GitHub uses a "$" to indicate that what comes after is the command you should run. **The "$" is not part of the command.** (We do not do this in our instructions).

5.2.1 Generating a new SSH key

1. Open Terminal.
2. Copy and paste this text into your terminal. **Replace the email given below with your GitHub email address** (the email address you used to sign up for Github). *Keep the quotations in your command.* Press enter to run the command.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

You will get this output:

```
> Generating public/private ALGORITHM key pair.
> Enter a file in which to save the key (/Users/YOU/.ssh/id_ALGORITHM):
```

Press enter. (This uses a default file and default file location.)

If you have already created a SSH key and you are asked to rewrite another key, look at the GitHub Docs for specific steps.

3. Type a secure passphrase (make up a password) when prompted with:

   ```
   > Enter passphrase (empty for no passphrase): [TYPE YOUR PASSPHRASE]
   ```

Before you freak out,

this passphrase is so secretive that won't see it being typed. You won't see a cursor moving and you won't see instead of the characters you're typing. Rest assured, your computer is receiving your text.

If you make a mistake, it's best to hit the "delete" bar many times, and retype.

   ```
   > Enter same passphrase again: [TYPE THE SAME PASSPHRASE]
   ```

   You should keep note of this passphrase for your own use. We (should) never have to use it again after finishing these steps.

   You will get specific output telling you information about your public key and key fingerprint. This is specific to the SSH connection you just made!

5.2.2 [Linux/Mac] Adding your SSH key to the ssh-agent

These are the instructions if you have a Linux or macOS computer. Go here for the Windows instructions.

1. In terminal, run the following command:

   ```
   eval "$(ssh-agent -s)"
   ```

   You will get this output:

   ```
   > Agent pid 59566
   ```

   (Your number will most likely be different than the one above.)

2. If you're using macOS Sierra 10.12.2 or later additions, you need to modify your `~/.ssh/config` file.

   1. Check if you have a `~/.ssh/config` file: Run the following command:

```
open ~/.ssh/config
```

2. If you get the following output:

   ```
   > The file /Users/YOU/.ssh/config does not exist.
   ```

   Create the file using the touch command: run the command given below

   ```
   touch ~/.ssh/config
   ```

3. Edit your `~/.ssh/config` file using the following instructions. (You can use any text editor you would like, such as vim). Below we use nano as a text editor.
   - Run `nano ~/.ssh/config`
   - Add the following lines to this file.

   ```
   Host github.com
    AddKeysToAgent yes
    UseKeychain yes
    IdentityFile ~/.ssh/id_ed25519
   ```

   - Exit nano: `ctrl + X`
   - Type "Y" and hit enter to save changes, when asked the following

   ```
   Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
   ```

3. Return to terminal. Run the following command:

   ```
   ssh-add --apple-use-keychain ~/.ssh/id_ed25519
   ```

   (You may be asked to enter your passphrase again. This is the same passphrase as before.)

5.2.3 [Windows] Adding your SSH key to the ssh-agent

1. Right click on Windows Powershell (you can search for it in your search bar on your taskbar) and select "Run as administrator".

2. Run the following commands:

   ```
   Get-Service -Name ssh-agent | Set-Service -StartupType Manual
   ```

```
Start-Service ssh-agent
```

3. Open a terminal window (without running as administrator). Run the following command, and **replace YOU with your GitHub username**:

```
ssh-add c:/Users/YOU/.ssh/id_ed25519
```

5.2.4 Adding a new SSH key to your account

1. Copy the SSH public key: Run the following command to copy the content of the ~/.ssh/id_ed25519.pub file to your clipboard:

   On Mac/Linux:

   ```
   pbcopy < ~/.ssh/id_ed25519.pub
   ```

   On Windows:

   ```
   clip < ~/.ssh/id_ed25519.pub
   ```

2. Go to your GitHub account on the GitHub website. Click on your profile picture (icon in the upper right). Then, select **Settings**.
3. Under the "Access" section, click **SSH and GPG keys**.
4. Click **New SSH key** or **Add SSH key**.
5. In the "Title" field, add a descriptive label for this key you are creating (ex. if this is your personal laptop, you can call the key: "Personal Laptop").
6. Leave the type of key as "authentication" (rather than "signing"). For our purposes, selecting authentication is fine.
7. In the "Key" field, paste (we are pasting what we copied in step 1).
8. Click **Add SSH Key**.
9. If you are prompted, confirm access to your GitHub account.

Finally, we're all done! We've created a SSH connection between your device and GitHub!

Thankfully, we only need to do these steps once! Additionally, most security questions are only asked the first time, so when you work on your workshop in the future, you will not have to redo these steps or confirm authentication.

5.3 Getting the Template on Your Local Computer - Git Clone!

1. Navigate to where in your local file system you want to have your workshop in Terminal/Windows PowerShell/Command Prompt.

Recommended Workshop Location:

CBW recommends that you create a folder within your Documents folder called "CBWGitHub", which is where you will place your CBW workshop project files.

In Finder/File Explorer:

- Navigate to 'Documents'
- Create a folder and name it "CBWGithub"

Copy the file address of the CBWGithub folder:

- On **Windows**
  - Using File Explorer, find your "CBWGithub" folder.
  - Right click the "CBWGithub" folder and press "*Copy as path*".
    - If you are currently inside the "CBWGithub" folder, you can right click on it's name in the header and press either "*Copy Address*" or "*Copy Address as Text*"
- On **Mac**
  - Go to the folder holding your "CBWGithub" folder (recommended to be your Documents folder)
  - Right-click (or tap your mouse pad with 2 fingers) the "CBWGithub" folder and press the "Options" keyboard key (in the bottom left).
  - While holding the "Options" key, go the the 4th section from the top, and click "Copy "CBWGithub" as Pathname".

You may want to paste your file address somewhere where you can quickly find it, so it is easier to navigate to in the future.

Navigate to your workshop folder in Terminal:

Navigate using the "cd" command. For example, if you used the recommended instructions, you should run "cd" and paste (since you already copied the file address).

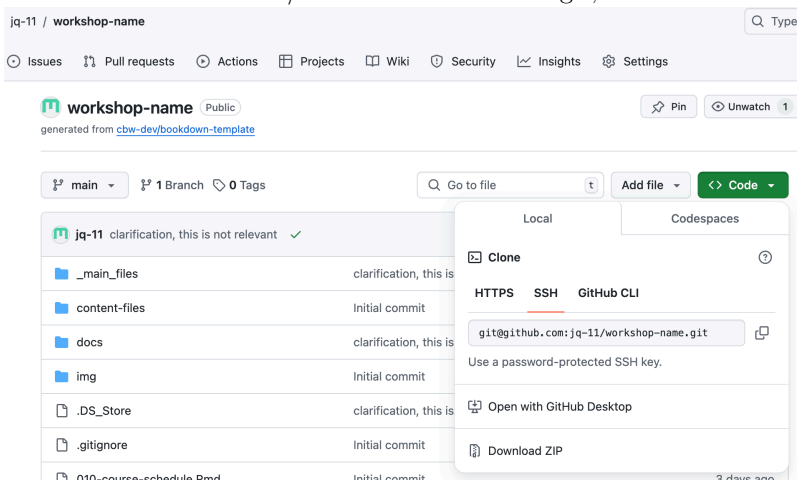
If you didn't use the recommended path and folder name, you can use "cd + tab" (where tab is the keyboard key, "tab") to try to find your path and folder.

2. Return to your workshop repository on GitHub. Press `< > Code`, which is the leftmost tab in the header bar on GitHub.

   Find the ssh for your workshop repository:

   a) Click the green button entitled `< > Code` ands see the drop down options.

   b) Click the SSH tab, as seen below, and then copy the text below it. The text should be something like git@github.com:bioinformaticsdotca/WORKSHOP-NAME.git, as seen below.



   c) **Edit** and **run the following command** in Terminal/PowerShell/Command Prompt, within the folder you want the workshop folder to be in. (Recall that we navigated there in step 1.)

EDIT THE FOLLOWING COMMAND!

You can essentially type "git clone" and then paste the SSH url, and then hit enter. Below, you must delete the entire "git@github.com:bioinformaticsdotca/[YOUR WORKSHOP NAME].git" text, and replace it with the text you copied.

```
git clone git@github.com:bioinformaticsdotca/[YOUR WORKSHOP NAME].git
```

3. You should be ready to go! With your given permissions, you should be able to git push (put your local edits on GitHub) and git pull (pull edits on GitHub to your local computer) fine!

Git Version Control Tip!

Consider having only one team member (or perhaps your RC) make git pushes or control pull requests. To avoid merge conflicts, designate 1 team member to control actual changes to your workshop repo. Other team members can fork or create branches, and create a pull request that the designated team member can check and overlook.

But what do any of these files mean? Which ones do I edit? Which ones shouldn't I edit? How do I open this in RStudio? And how exactly is a page made from all these files??? It's time for you to go to the next page :D

5.3.1 Oops, I Git Cloned the Wrong Repository and I Want to Delete it from my Local Computer!

That's ok! To delete the entire local repository and the folder itself, run the following command:

```
rm -fr folder-path
```

where "folder-path" is a file address to the git cloned folder/repository that you want to delete.

Chapter 6

So What Do These Files Mean?

Ok now we have our workshop locally (on our computer), which is made up of all these files and folders?

Before we dive deep into what to do with these folders, let's explore how bookdown actually works and how to understand RStudio.

6.1 Bookdown Simple Explanation

Here is a general summary of how Bookdown creates html websites from .Rmd files.

Essentially, knitr renders and runs all the code, and the outputs are converted into markdown. After knitr, we essentially have a bunch of only markdown files.

Pandoc translates this markdown into html, so that we get a website! It can be helpful to know when and how these packages work, to help debug later on!

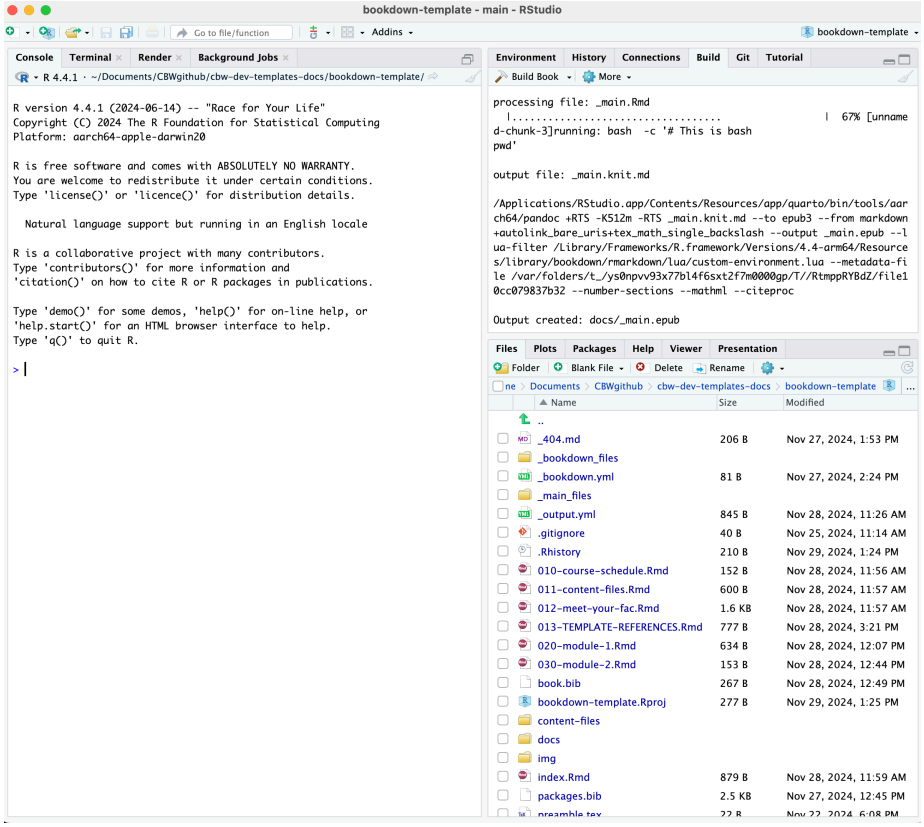
Now, let's figure out RStudio. Skip to file setup if you already know how to use RStudio (and it's built in git control window).

6.2 Opening Your Bookdown Project in RStudio

Enter the folder you just git cloned using Finder/File Explorer, it should be titled "[workshop-name]". Right click on [workshop-name].Rproj and press "Open

in RStudio". There is only one file with this file extension. The .Rproj file is what you will open each time you want to work on this workshop! You must explicitly **open the .Rproj** file to build properly!

A RStudio window should open up and look something like the image below.



6.3 Explaining RStudio

In the bottom right, we have all of our files and subfolders. These files will be explained below. This window also contains helpful views, like "Viewer" and "Plots". We will touch on these later.

Try opening `index.Rmd`: a new pane will open in the top left that shows the contents of `index.Rmd`. This is where we will be editing our files! Notice, the "Knit" button.

In the bottom left, we have our console and other debug related windows (such as terminal!). Any code we run will appear in the console. We can access the terminal (just like editing in the Terminal app) under the "Terminal" tab.

In your top right, we have a different window with more different views. The most relevant windows to us are the "Build" and the "Git" windows.

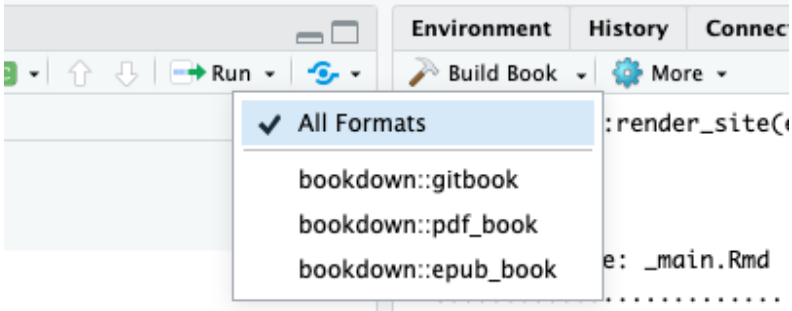
> No "Git" Window?
>
> Try closing (and maybe even restarting RStudio) and then reopening it. A "Git" tab should appear to the right of the "Build" tab and to the left of the "Tutorial" tab.

6.4 Build the Book

Try pressing "Build Book" within the "Build" window. Your build window is going to fill up with text, and soon, a website is going to pop-up as your new window. This is the website you will be editing to create your workshop!

> Note: If it's your first time building, you may have to approve some downloads and installations, like we did in Getting Started!

By building the book, **all** of these files were compiled and converted to .html files, that create a website. Each time we make local changes to our files and we want them to appear in our website, we need to rebuild the book. Note that each time we build our book, the files we edited will be saved first (we don't have to save before building!).

> Note: By default, RStudio will choose to build "All Formats":
>
> 
>
>
> You can choose to only produce a gitbook (the first option). This can be helpful if you are encountering errors claiming that "bookdown::render_book() failed to render the output format 'bookdown::pdf_book'/bookdown::epub_book"'. Additionally, not producing all output formats can decrease your build time significantly! (However, not being able to produce the other ouputs may suggest there is an underlying bug. It is ideal to be abble to produce all outputs.)

6.4.1 Other Ways to Build Your Book

1. Build the book from the R console:

```
bookdown::render_book()
```

2. Press the keyboard buttons: `cmd + shift + B` (macOS) OR `ctrl + shift + B` (windows)

6.4.2 Knit Your Book

Building can take a long time. If you are editing just one file, you can press the "Knit" button that is at the top of the window with your file. This will run the code in the page, and show you what that page would look like in the website (as well as saving that file).

> Note: Other pages in your website will not update.
>
> A quicker way to knit is using the keboard controls
> `cmd + shift + K` (macOS) OR `ctrl + shift + K` (Windows)

6.4.2.1 Knit VS Build

Building creates the website using **ALL** the .Rmd (and other) files. This is why it takes so long. Knitting creates/updates **only** the .Rmd file you're currently working in and reruns your code. (Note: You can't knit a non-.Rmd file!)

Before pushing to GitHub, it is a good idea to build, in case of faulty links or small bugs that occur if you only knit or preview.

6.4.3 Preview Your Book

If you want live updates to your changes, you can preview the page as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in "Preview book", or from the R console (in the bottom left window):

```
bookdown::serve_book()
```

But which files do we edit? Well alas, it's time to discuss the file setup.

6.5 File Setup Explanation

Recall:

Hence, all the .Rmd files create a page in the produced website! To create more pages, you will be creating more .Rmd files.

Here is a tree diagram of the bookdown template setup. (Note that this mimics how RStudio displays the files, if your RStudio shows these template files in a different order, right click your file window and press "Reload")

> **Note:** only relevant files or files you might be concerned about are explained in blue. If there is no explanation, you can safely assume you do not have to worry about it.

```
bookdown-template
  _404.md # This becomes your 404 page. This is the only file that can become a page that can al
  _bookdown.yml # a config file, you most likely won't have to edit it unless you want additiona
  _bookdown_files
  _main_files
      ... # [in case of confusion: image and pdf output files]
  _output.yml # a config file, RC/workshop team will have to edit a few things (the workshop nam
  .gitignore
  .Rhistory
  010-course-schedule.Rmd # --> "Course Schedule" section
  011-content-files.Rmd # --> "Pre-workshop Materials" and "Computing Setup & Downloads" section
  012-meet-your-fac.Rmd # --> "Meet Your Faculty" section
  020-module-1.Rmd #--> "Module 1", "Lecture, and "Lab" section
  030-module-2.Rmd
  README.md # README file that has some helpful explanations, consider reading this before start
  book.bib # citation file
  bookdown-template.Rproj # always open this file in RStudio, opening this helps bookdown unders
  content-files # store all files from your modules here (ex. data sets, empty code worksheets)
      sample-pdf.pdf
  docs
      ... # html files & other generated content
  img # folder where you should store all your images
      bioinformatics.ca-logo.svg
      faculty # store all images of faculty in this folder
          michelle-brazas.jpg
          nia-hughes.jpeg
      favicon.ico
      sponsors # store all sponsor logos/titles in this folder
           Your-Sponsor-Here.svg
  index.Rmd # landing page --> "Welcome" section
```

```
packages.bib # R generated citation file
preamble.tex
style.css # css styling options are defined here - feel free to add your own styling
```

__output.yml & __bookdown.yml 2 important files are the `_output.yml` files and the `_bookdown.yml` files. They help tell bookdown what we want and what to do, especially when making our website.

You will only have to edit _output.yml a bit. The _output.yml file creates the table of contents/sidebar you see on the left of the workshop. It is written in HTML, which is why it looks so different. There are 3 things either the RC or the workshop team will need to do:

1. Change the workshop name: [YOUR WORKSHOP NAME] on line 7.
2. Add your sponsors: Replace "Your-Sponsor-Here.svg" with the file name of your sponsors name/logo. Remember to place these files inside the ./img/sponsors/ folder.
3. Replace your workshop repo edit link. This allows users to suggest edits to your work on GitHub. Follow these instructions (they are also provided in the template).
    1. Go to your index.Rmd page on GitHub, Copy and paste the link to it.
    2. Copy and paste the link to it, excluding the "index.Rmd" ending.
    3. Add `/%s` to the ending.
    4. Replace the above link.

Organizing the Table of Contents The order of the sidebar is completely dependent on the alphabetical order of the files (see your bottom right window pane! The order is generally the same as the order shown in the RStudio file window pane. If not, right click the window and press "Reload"). Our template has numbering first, to help ease our understanding and organization of files. The only file without this is `index.Rmd`, since as the landing page we can not change the file name.

index.Rmd The only .Rmd that must have some configuration details is the landing page: **index.Rmd**. This is what fills up the beginning of the index.Rmd file, surrounded by the ---, which tells bookdown the configuration information.

Follow the instructions in the template's index.Rmd file - you must fill in the title, author, date and url. The description, cover-image and github-repo can be changed, but aren't mandatory.

.Rmd Files Let's move on to discussing the breakdown of the .Rmd files. Each new page is defined by a new header, which starts with `#`, each sub-header has increasingly more `#` symbols (##, ###, and more all create smaller sub-headers). A "main" header only has 1 `#` symbol. These headers define new pages, as well as the title in the sidebar referring to that page.

Hence, if we have multiple "main" headers in 1 .Rmd file, 1 .Rmd file will encode multiple pages. Try to have only 1 single # as a header on one page! (You may get warnings otherwise).

However, if a new file only has sub-headers (2 `#` or more), the sub-header will appear on the page that was defined by the previous .Rmd file (in alphabetical order - see the order of the "File" window pane).

In our template, we want the introduction material to appear on the same page. Hence, the introduction files (other than 020-course-schedule.Rmd, which starts the page) all only have sub-headers.

Note: Due to this, you may receive this warning:

> Warning message: In split_chapters(output, gitbook_page, global_numbering, split_by, : You have 6 Rmd input file(s) but only 4 first-level heading(s). Did you forget first-level headings in certain Rmd files?

You can ignore this warning message!

How to Create a New File You can create new files anywhere. You should only be making new website pages, so you should also only be making .Rmd files. (Creating new .md files in hopes of making a new page usually ends up with weird bugs, stick to .Rmd files).

1. Under the "Files" tab of the lower right window pane, there are many options to modify your files. You can use these buttons instead of your File Explorer. Click "New Blank File".
2. Click the second dropdown option: R Markdown
3. Give your file a unique name that matches CBW's Bookdown template file naming convention. That is, "0XX-[description].Rmd". Decide what X should be depending on where you want the page to show up on your sidebar. Name your file a short but descriptive name, with hyphens (-) to split up words.
4. Start editing your new file in the upper left window!

404 Page That being said, there is one file that can be a markdown (.md) file: the _404.(R)md file. If website user goes to a webpage that belongs to our website but no longer exists, they will get this 404 page instead. Put links to your landing page for your workshop or the bioinformatics.ca homepage here!

CBW Organization Folders We want our workshops to be easy to navigate and understand. Hence, images or files that are required for your lessons should be placed in `./img/` and `./content-files/` respectively. Within these files, try creating sub-folders within these folders to help organize groups of files. For example, if you have a bunch of data sets, try creating a "datasets" folder within `./content-files/`.

Subfolders already exist in `./img`: `./img/faculty/` and `./img/sponsors/`. Fill these with photos of your faculty and images of the names'/logos' of your sponsors, respectfully!

Bookdown's Wesbite Building Folders Bookdown makes some folders to store .html files, among other files that help create the website. Hence, you should not (need to) touch:

- _bookdown_files
- _main_files
- docs

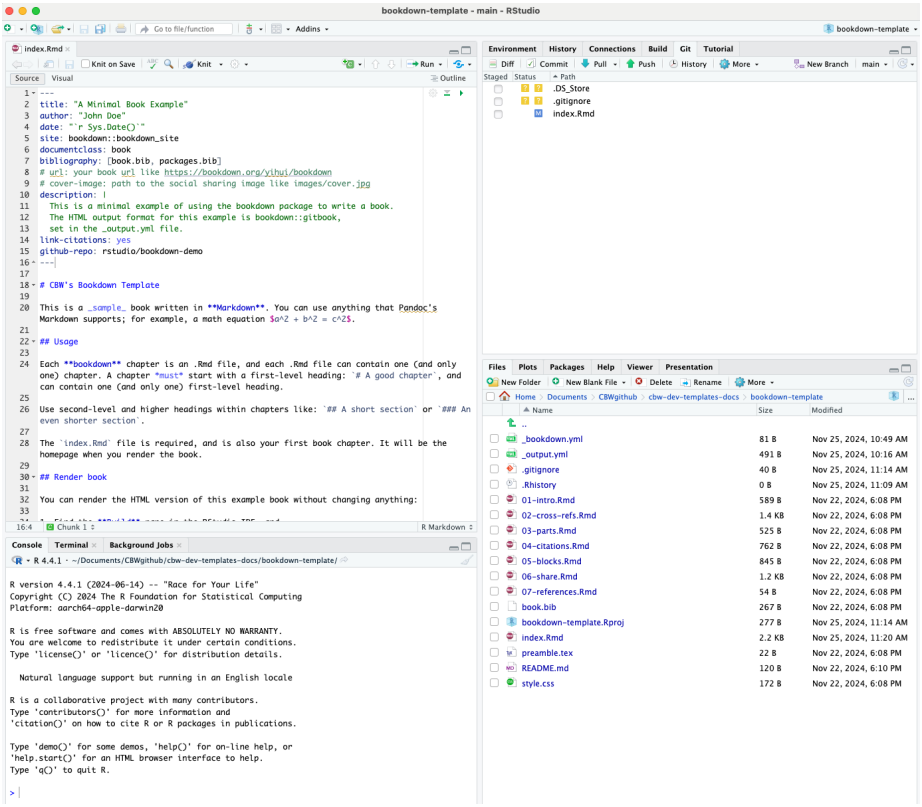
> **Debugging Tip**: If you do get an error that seems like it's coming from one of these files, there was probably an error that resulted from bookdown not being able to process something. Double check that your syntax, formatting, file names and related files are correct.

> **Debugging Tip 2**: Each time we build, we produce a "docs" folder. If we have already built once, we just add new files to it. HTML files that were produced from .Rmd that were deleted are not removed. Hence, if you see errors from files that no longer exist in your root project folder, try deleting the whole docs folder, and rebuilding.

6.6 Push to GitHub via RStudio

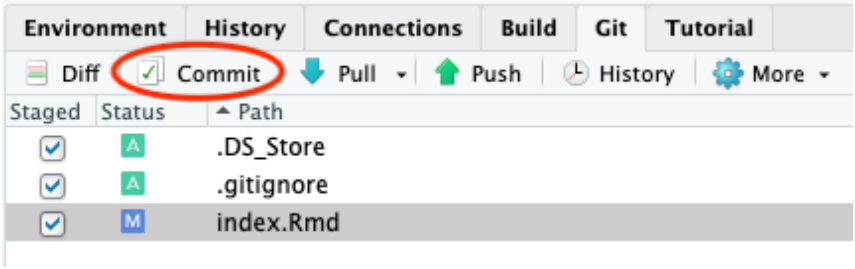
Now, we know what our files mean and how to edit them. How do we get this onto GitHub? We can write git commands into our Terminal/Command Prompt, or alternatively (and more easily), RStudio has a built-in git interface.

Now, we will be able to see a Git window in the top right. Click "Git" to open this window.

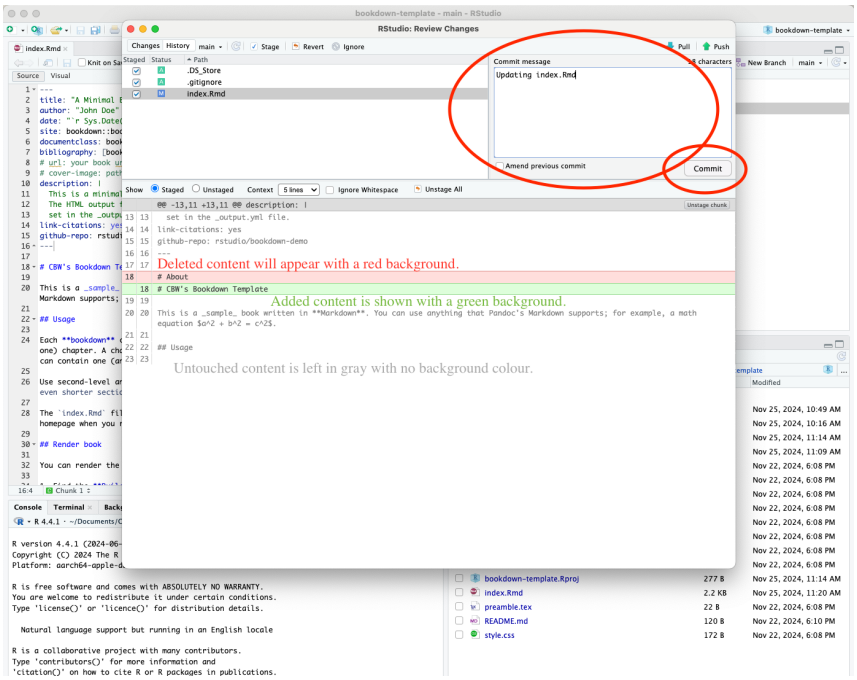


Let's say we only edited `index.Rmd`, now we see the newly edited files. Changed files that need to be updated on GitHub will show up in this window, like how `index.Rmd` is seen above. (Do not worry too much about `.DS_Store` and `.gitignore` do.) Let's try to push this change to GitHub.

1. Select all the edited files.



2. Then, click the Commit button, which appears above your selected items. A window pane will appear (shown below).



3. Add a commit message in the corresponding box, and then press commit below it.

4. A new window will show up, detailing your updates. Close this window and then press **Push** to push your updates to GitHub.

Now, we're done! We should see the updates on GitHub now. Also note, if we ever want to pull updates from GitHub, there is also a **Pull** button in the Git window within RStudio!

Git pushing puts your edits onto GitHub, *git pulling* takes the edits make on GitHub, and brings them to your local computer. For example, if one of your

workshop team members made an edit, you want to have that edit on your computer before you start editing! It's a good idea to do this (git pull - upper right window, "Pull" button) before you start editing, in case somehow your edits conflicts with their edits.

Git pushing will automatically update the website, you can see the updates and progress in the actions window we saw previously. (Check out your website on the web once it's done deploying!)

Chapter 7

Formatting Your Content - Markdown

Now we know which files to edit, but how can we edit these files? How do we format our text, links, headers, …?

7.1 Visaul R Markdown

All our files are made up of code and markdown syntax, that becomes formatted into our website. However, if you do not feel comfortable using Markdown (or you just want an easier way to write your pages), see this guide on using Visual R Markdown.

It can be significantly easier to use Visual R Markdown, especially for those who are used to working in Google Docs!

If you want to use Markdown, a basic syntax guide is provided below!

> Is this not enough for you?
>
> Bookdown has thorough documentation for specific elements, customizations, and further explanations. For simple elements, everything is provided below. However, there are many more possibilities with Bookdown! Click here to read more!

> **IMPORTANT NOTE**: Now that you're straying away from CBW's Bookdown template, there's a chance you may run into errors. See this page for help!

7.2 Chapters

As mentioned earlier, headers are defined by a # before the title. Subheaders get increasingly more nested as add more # symbols before it. For example,

```
# Hello
```

would create a chapter title. Since there is only one `#` symbol, this would also create a new page. Again, try to keep only one chapter title per .Rmd file.

```
## Subheader
```

7.3 Subheader

This is what a subheader would look like.

```
### Subheader
```

7.3.1 Subheader

This is what a subheader with 3 `#` symbols would look like. You can add as many `#` symbols as you would like! There is also no limit to the the number of subheaders you can have.

An unnumbered subheader

Chapters and sections are numbered by default. To un-number a heading, add a `{.unnumbered}` or the `{-}` at the end of the heading. For example, the above subheader was written like this:

```
### An unnumbered subheader {-}
```

7.4 Parts

Notice how index.Rmd in our CBW Bookdown template has `# (PART) Introduction {-}` (followed by `# Welcome`). This creates the "Introduction" section on the sidebar.

There are already 2 main parts in the template: the introduction and modules sections.

If you want to add more parts, simply paste this: `# (PART) [Part Title] {-}` into a new .Rmd file, at the top of the file, before the main `# header` for that page/file.

Note: If you would like to have a un-numbered part: Use this syntax: `# (PART\*) [Part Title] {-}`

Another Note: A similar un-numbered part is called an "Appendix" and Bookdown. Like a part, paste `# (APPENDIX) [Appendix Name] {-}` it above the "main" header you would like it to be before. All subsequent headers will be described via letters rather than numbers. (If your following headers are un-numbered, an Appendix looks the same as an un-numbered part).

7.5 Text Formatting

Before we go ahead with much more formatting possibilities, let's get the basics of markdown formatting options for basic text. (Tip: There are many resources online that can help give more information!)

Bold

```
**bolded text**
```

bolded text

Italics

```
*italicized text*
```

italicized text

Subscript

```
H~2~O
```

H_2O

Superscript

```
Na^+^
```

Na^{+}

Footnotes

`^[]` is the formatting for a footnote. For example

```
Here is a footnote: ^[This is a footnote.].
```

Here is a footnote: [1].

(Scroll to the bottom of this page to see it.)

Horizontal Line

```
---
```

Blockquote

```
> blockquote
>
> --- someone who blockquotes
```

> blockquote
>
> — someone who blockquotes

You can put emojis to help these standout! For example:

> **Warning:** Do not push the big red button.

Unordered List

[1] This is a footnote.

```
- unordered
  - list
    - element
- cool!
```

- unordered
  - list
    - element
- cool!

  Warning: Bookdown currently can not render Markdown to-do lists properly. Alternatively, use this tip instead!

Ordered List

```
1. First item
    1. a <!-- For nesting inside a list, put 2 tabs before the element -->
    2. b
2. Second item
3. Third item
```

1. First
    1. a
        1. c
        2. d
    2. b
2. Second
3. Third

Code

There are 2 ways to write code:

1. `` `code` `` which looks like this: `code`
2. Or long pieces of code

````
```
1 + 1  # a line of code
```
````

which would render into the following on your website

```
1 + 1  # a line of code
```

Find more code options and information in the How to Render Code section.

Images

The basic notation for a file is:

```
![Alt Text](img/.../...)\
```

For example:

```
![bioinformatics.ca logo](img/bioinformatics.ca-logo.jpg)\
```

Note: Markdown does not actually require the backslash "\". However, since Bookdown uses Pandoc, we should add a backslash so that the alternative text does not become a caption for the image. This occurs when the image is not directly following or followed by text. Here is where it's discussed in the Pandoc documentation!

> Note: Before adding the "\", Bookdown will show you the image that you are referring to (double-click the file address link) and it will remain on screen, within your .Rmd file. *You can use this to double check the file address is correct.* After adding the "\", you can double-click to check the file address is pointing to the right image (and it should hover below), but it will not remain on screen.

Links

Links to a webpage are as follows:

```
[Text you want to hyperlink](website link)
```

For example,

```
Click [here](https://bioinformatics.ca/) to see the bioinformatics.ca main page!
```

Click here to see the bioinformatics.ca main page!

Links to Other Pages/Sections within your Workshop Website

Note these steps work for all headers, not just main (1 `#`) headers:

1. Label the heading: `# Hello world {#nice-label}`.
   - If you do not want to include a label, an automatic label is made: `# Hello world` = `# Hello world {#hello-world}`.
   - To label an un-numbered heading, use: `# Hello world {-#nice-label}` or `{# Hello world .unnumbered}`.
2. Next, reference the header in either 2 ways:
   - If you want the number referring to your header, use: `\@ref(label-name)`. For example:

     ```
     Please see Citations \\ref{citations}.
     ```

     Please see Citations 7.6.

     > Note: If you use this reference style for an unnumbered header, "**??**" will be used as the hyperlinked text.

   - If you want a customizable text hyperlink, use the following:

     ```
     See the [Citations](#citations)!
     ```

     See the Citations!

7.6 Citations

Bookdown can help us create citations!

First, tell bookdown your sources!

```
# automatically create a bib database for R packages
knitr::write_bib(c(
  .packages(), 'bookdown', 'knitr', 'rmarkdown'
), 'packages.bib')
```

This R code created a file called 'packages.bib' (which is in the bookdown template) which is now filled with a bunch of bibliography entries. 'book.bib' also has a manually entered bibliography entry. We must tell bookdown the files with bibliography information via the YAML `bibliography` key in index.Rmd. A bibliography entry essentially looks like:

```
@type{key,
  title = {...},
  author = {...},
  organization = {...},
  year = {...},
  url = {...},
  ... # see more possible fields at https://en.wikipedia.org/wiki/BibTeX
}
```

`@type` refers to what type of source it is (ex. an article, book, manual, ...). `@key` refers to the name we refer to it by in in-text citations.

For example, `@R-bookdown` renders to Xie [2024]. Adding square brackets also adds parentheses once it's rendered: so `[@R-bookdown]` becomes [Xie, 2024].

Note: The RStudio Visual Markdown Editor can also make it easier to insert citations!

On any page that you cite, a **References** section will be autogenerated at the bottom of that page (scroll down)!

Note: When you cite, Bookdown also generates a citation list that it adds at the end of the last page of your website. However, Bookdown does not create a title, so on your very last page, add a `# Reference` main header. (To avoid having a "2 main headers in the same file" error, you can create a new .Rmd file with only `# Reference`).

7.7 Equations

Equations are written in latex. You can refer to them with a label you specify after writing it out!

```
\begin{equation}
  f\left(k\right) = \binom{n}{k} p^k\left(1-p\right)^{n-k}
  (\#eq:binom) # the label!
\end{equation}
```

$$f\left(k\right) = \binom{n}{k} p^k\left(1-p\right)^{n-k} \tag{7.1}$$

You may refer to using `\@ref(eq:binom)`, for example: `see Equation \@ref(eq:binom)` becomes "see Equation (7.1)". The number (7.1) depends on the chapter and the number of equations already in said chapter (the first equation will be numbered `[chapter number].1`).

7.8 Theorems and Proofs

We also have specific syntax for theorems and proofs, for example, this code:

```
::: {.theorem #tri}
For a right triangle, if $c$ denotes the *length* of the hypotenuse
and $a$ and $b$ denote the lengths of the **other** two sides, we have
$$a^2 + b^2 = c^2$$
:::
```

Theorem 7.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

$$a^2 + b^2 = c^2$$

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem 7.1.

Learn more about styling and syntax for theorems and proofs here!

7.9 Callout blocks

CBW's Bookdown template comes equipped with a "warning", "tip" and "note: callout blocks!

Here are the code you will use to add them to your template and the rendered output.

```
:::: {.redbox data-latex=""}
::: {.center data-latex=""}
**WARNING!**
:::

Write your warning text here.
::::
```

WARNING!

Write your warning text here.

```
:::: {.bluebox data-latex=""}
::: {.center data-latex=""}
**NOTE!**
:::

Write your note here.
::::
```

NOTE!

Write your note here.

```
:::: {.greenbox data-latex=""}
::: {.center data-latex=""}
**TIP!**
:::
```

```
Write your tip here.
::::
```

TIP!

Write your tip here.

Formatting Tip:

Put `<br>` on the line after `::::` to create a line between your callout and the next piece of content. If this line break is too big, try adding this instead `<p style="font-size: 8px;"></p>`, and change the font size to increase/decrease the line height.

This formatting is a div that is specified in Pandoc (recall: a library Bookdown uses to create our .html website pages). Bookdown will help colour your divs so that you hopefully won't forget to include the proper amount of colons. This ends your div properly. If you stick to our callouts, you will need 4 colons at the end (`::::`).

Known Error:

If you do not end your div, all following code will not render (as if your book ends with the unfinished div)!

Warning about Callout Blocks

Our Callout blocks are written with white text. Hence, code does not appear well in our callout blocks, since it has a light white/grey background.

Creating your own callouts can be difficult, and you should account for making sure both the wesbite and pdf (which uses LaTeX, so you will likely have to edit `preamble.tex`) generate properly.

The R Markdown Cookbook provides more help on how to use custom div blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html.

7.10 TIP: Markdown Not Working/Not Enough? Use HTML!

Recall from this diagram that all files are rendered into .html files for the final official website.

Both Knitr and Pandoc will ignore HTML code, so if a certain part of Markdown's formatting isn't working, or you're not satisfied with Markdown's formatting options, use HTML/CSS formatting instead!

Here are some examples:

1. Markdown has no formatting options for underlining. Hence, underline text in a .Rmd file with the following HTML syntax:

   ```
   <u> underlined text</u>
   ```

   underlined text

2. Bookdown currently renders to-do lists incorrectly. Instead, you can use the following HTML code:

   ```
   <div style="list-style-type: none;">
     <label><input type="checkbox"> check box</label><br>
   </div>
   ```

   check box

Chapter 8

How to Render/Compile Code

There are many ways to render and compile code using Bookdown!

Here are most of the "language engines" (programming languages) available to render, run and compile in Bookdown!

```
names(knitr::knit_engines$get())
```

```
##  [1] "awk"         "bash"        "coffee"      "gawk"        "groovy"
##  [6] "haskell"     "lein"        "mysql"       "node"        "octave"
## [11] "perl"        "php"         "psql"        "Rscript"     "ruby"
## [16] "sas"         "scala"       "sed"         "sh"          "stata"
## [21] "zsh"         "asis"        "asy"         "block"       "block2"
## [26] "bslib"       "c"           "cat"         "cc"          "comment"
## [31] "css"         "ditaa"       "dot"         "embed"       "eviews"
## [36] "exec"        "fortran"     "fortran95"   "go"          "highlight"
## [41] "js"          "julia"       "python"      "R"           "Rcpp"
## [46] "sass"        "scss"        "sql"         "stan"        "targets"
## [51] "tikz"        "verbatim"    "theorem"     "lemma"       "corollary"
## [56] "proposition" "conjecture"  "definition"  "example"     "exercise"
## [61] "hypothesis"  "proof"       "remark"      "solution"
```

8.1 Rendering Code

Just rendering code refers to Bookdown formatting code so that our users can view it. The code does not run or compile. This is very similar to what is shown

in 020-module-1.Rmd of the bookdown template. An example is shown below.

````
```
insert code here
```
````

which appears as

```
insert code here
```

These are called "code chunks".

8.2 Rendering Code with Highlights for Specific Languages

Bookdown also highlights (specific to the language) the outputted code for the ease of understanding. To do so, add the name of the language (of the many shown above) after the first 3 ` symbols. Note that the language is not case sensitive (so both "r" and "R" would render the same below). For example:

````
```r
x <- 42
x
```
````

renders into:

```r
x <- 42
x
```

Notice that since <- is how to assign values to variables in R, it is highlighted. If we were to replace "r" with a different language, this may not be highlighted (depending if "<-" is a relevant symbol in that language.)

8.3 Rendering and Compiling Code

To have the code actually compile, we need to put "{" and "}" brackets around the language, which we still put after the first 3 ` symbols. For example:

````
```{r}
x <- 42
x
htmltools::HTML('<b>LEFT`RIGHT</b>') htmltools::HTML('<b>LEFT`RIGHT</b>') htmltools::HTML('<b>LEF
```
````

renders and compiles into:

```
x <- 42
x
```

```
## [1] 42
```

The second bar you see is the output of R command!

You can run these "code chunks" without previewing or building the book! To the right of your code chunk, there are 3 symbols (as seen below).

The gear symbol leads to more chunk options click on it to see more options.

The down arrow symbol runs all previous chunks and the current chunk. This is helpful if previous chunks define a variable that you will need in your current chunk.

> **Note:** Hence, we can use variables from previous chunks!

The right-pointing arrow symbol runs the current code chunk. A rectangular box will appear below your code chunk, with the output of your code.

Additionally, notice that in your bottom left window, any code you run also runs in your console. Both of these ways of checking your code happens for all languages! If you are previewing your book, you must re-preview it, since running the code becomes the new action for your console, instead of previewing the book.

Note!

If you want to run a certain language, you must have the language installed! R and python are already setup for you. (Python relies on the RStudio interface option, reticulate - you can reconfigure this if you'd like.) You may have to configure new languages so that they can run in RStudio, but generally, they should be able to run automatically.

Exiting the Console

If you're running R, you don't have to exit the console (we already work in the R console when using Bookdown). However, if you're running a different language (for example, python), you will remain in the Python console, and must exit it to run Bookdown commands. You can look online for different ways to exit certain consoles, but generally running "quit" will return you to the R console.

Now we know how to both render (just show) and compile (see the output) of our code!

8.4 Code Chunk Options

Bookdown has options that we can include, to help with certain options for our code to appear and what it may produce.

Generally, the typical structure of adding code chunk options is shown below.

```
```{r, option=VALUE, option=VALUE, ...}
code
```
```

Some common code chunk options are:

- `echo`
  - `echo=TRUE` shows the code output (by default, it is on).
  - `echo=FALSE` hides the code output.
- `eval`
  - `eval=TRUE` runs the code (default).
  - `eval=FALSE` skips the chunk and does not execute the code.
- `results`
  - `results='markup'` runs the code (default).
  - `results='asis'` leaves the output with no additional formatting.
  - `results='hide'` does not show the output.
- `include`
  - `include=TRUE` includes both the code and the output on your website (default).

- `include=FALSE` excludes both the code and the output on your website.

Click here to find more code chunk options!

8.5 Code Chunks for Code-Generated Figures and Tables

We can use code chunks to help specify certain ways for code-generated figures and tables to appear, and how to link to them!

Generally, this will look like:

````
```{r reference-name, option=VALUE, option=VALUE, ...}
code that generates an image/table
```
````

Then, we can refer to the figure or table generated by the code in this chunk by using `\@ref(fig:reference-name)` or `\@ref(tab:chunk-label)`

For example, an R-generated image can be made using a similar code chunk to the one shown below:

````
See Figure \@ref(fig:nice-fig).

```{r nice-fig, fig.cap='Here is a nice figure!', out.width='80%', fig.asp=.75, fig.align='center
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```
````

This renders into:

See Figure 8.1.

```r
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

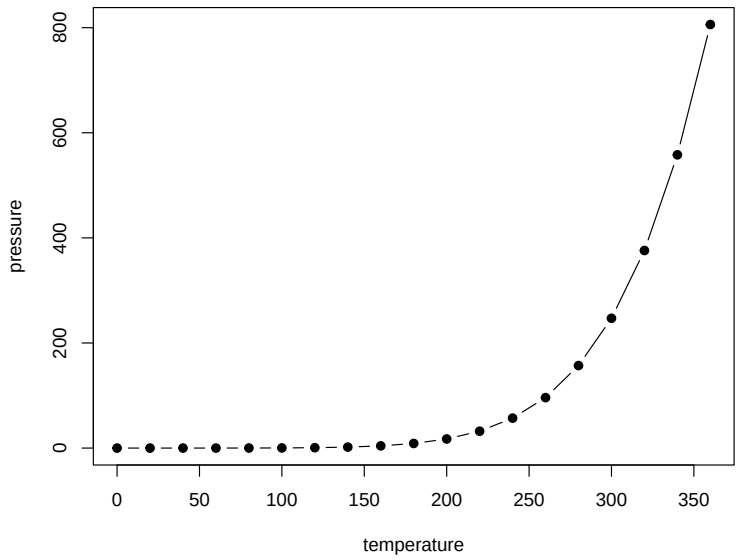


Figure 8.1: Here is a nice figure!

Here is an example of a table generated by R from a code chunk:

````
Don't miss Table \@ref(tab:nice-tab).

```{r nice-tab, tidy=FALSE}
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```
````

This renders into:

Don't miss Table 8.1.

```
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

Table 8.1: Here is a nice table!

temperature	pressure
0	0.0002
20	0.0012
40	0.0060
60	0.0300
80	0.0900
100	0.2700
120	0.7500
140	1.8500
160	4.2000
180	8.8000

You can find more inforamtion on figures and tables in Bookdown here!

Chapter 9

Brain Dump / FAQ

Congrats you got through most of CBW's Bookdown Documentation!

9.1 Danger Zones

Here are some common mistakes that can lead to problems - danger zones!

1. Forgetting to pull your workshop teams recent edits before you edit!

   Ex. Forgot to pull before editing?

   - Your git window will say (at the top of the window) your branch is ahead of the main branch
   - You will probably have to deal with a merge conflict
   - [this is pretty in-depth]

9.2 Potential Errors & Bugs

Trying to build and getting **"Exited with status 1."**? Bad news: Your book is not building properly. Good news: here are some *(hopefully)* helpful debugging tricks.

- **"Error: LaTeX failed to compile _main.tex."** - Essentially, something in your code is not allowing the output to be produced properly. Something in your code is producing incorrect output that is not allowing output to be produced. > TIP: Try to comment out recent changes. Try to decipher what exactly caused the build to fail. This will help you find your bug.

- **[X].html not found** - This may result in bad syntax that creates a faulty file. First, fix whatever bug caused that faulty file to occur. Then, rebuild. If you continue to get this error, it may be because the rebuild did not delete the previous faulty html file. Try deleting the docs file and rebuilding. (This can be a good debug solution in general!)
- If you get a bug where your website builds into a website that looks like it is made using very simple html, you may have to change your permissions. If you see **"Permission denied"** in your warning messages, trying running this command `chmod -R u+w docs` in terminal, in the folder containing your docs folder [CLARIFY ?]
- **"Could not produce X output"** - by default, bookdown builds all possible formats: gitbook, pdf, epub versions. We only need the gitbook, so if you're having issues, change your build settings to only creating the gitbook [CLARIFY + INCLUDE IMAGES]
- A very common bug is **"missing X package"**, just install it using this command in your console (the bottom left window in RStudio) `install.packages("missing package name, include these surrouding quotations")`
- Not being able to produce a specific format - If you're getting one of the following errors:
  - bookdown::render_book() **failed to render the output format** 'bookdown::pdf_book'
  - bookdown::render_book() failed to render the output format 'bookdown::epub_book'

  There is an easy fix! Press the dropdown options in the "Build Book" button and select "bookdown::gitbook". We only need to produce the gitbook (the html pages!) so it's ok if we can not produce the other pages. (However, it is ideal if we could produce all format types, since not being able to may lead to more errors.) This was likely due to creating something in one of the other formats that does not work in that format. For example:

  ```
  - **"File '..._svg-tex.pdf' is missing."** is a bug that only occurs when produc
  ```

9.3 Ease of Use

9.3.1 Wrapping Your Code

Sometimes, RStudio defaults so that there is no text wrapping when viewing your files. If you notice that you have to scroll horizontally to see the long lines in your files, turn on "Soft-Wrap Long Lines".

1. In your upper left corner of your RStudio Window, you will see options "File", "Edit", "Code". (If you are in full-screen on a mac, you will have to bring your cursor to the upper left corner before seeing these options). Click "Code".

2. Go to the 5th option "Soft-Wrap Long Lines" and select it.

9.3.2 Visual R Markdown

If you are having difficulties using Markdown Syntax, RStudio has a work around! See Visaul R Markdown for more information.

9.4 FAQs

1. I made a change via GitHub and and it's not appearing on the website. Why can I not see my edit?

   Let's recall this image.

   We build our **HTML** files using Bookdown. This means, any edits we make have to be local, so we can build our website and have the changes in our HTML files. GitHub Pages (which generates our website) only looks at the HTML pages, so even though there is a change on GitHub, it is on the .Rmd files. Hence, we don't see it on the actual website!

   Next Step: Git Pull your change locally. Build the website. Git Push and then see your edits once the website finishes deploying!

Part I

DEVELOPERS GUIDE

Chapter 10

Build Site

10.1 How to edit _bookdown.yml

- add a new line, `output_dir: "docs"` to _bookdown.yml
- build the site
- add a `.nojekyll` file into the produced docs folder

10.2 How to edit _output.yml (RC)

- after *before:* change your workshop name link
- after *edit:* put the link to the workshop repo, and end the link with `/%s`
- save

10.3 Mandatory "index.Rmd" landing page

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: `# A good chapter`, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: `## A short section` or `### An even shorter section`.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

10.4 Build the book:

- "Build" button in RStudio IDE /OR/ bookdown::render_book()
- Preview the book: - updates on saves in viewer window bookdown::serve_book()

Before building

10.4.1 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select "All formats" if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

10.4.2 Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in "Preview book", or from the R console:

```
bookdown::serve_book()
```

Chapter 11

Git Instructions

11.1 How to Make a Git Repo (RC)

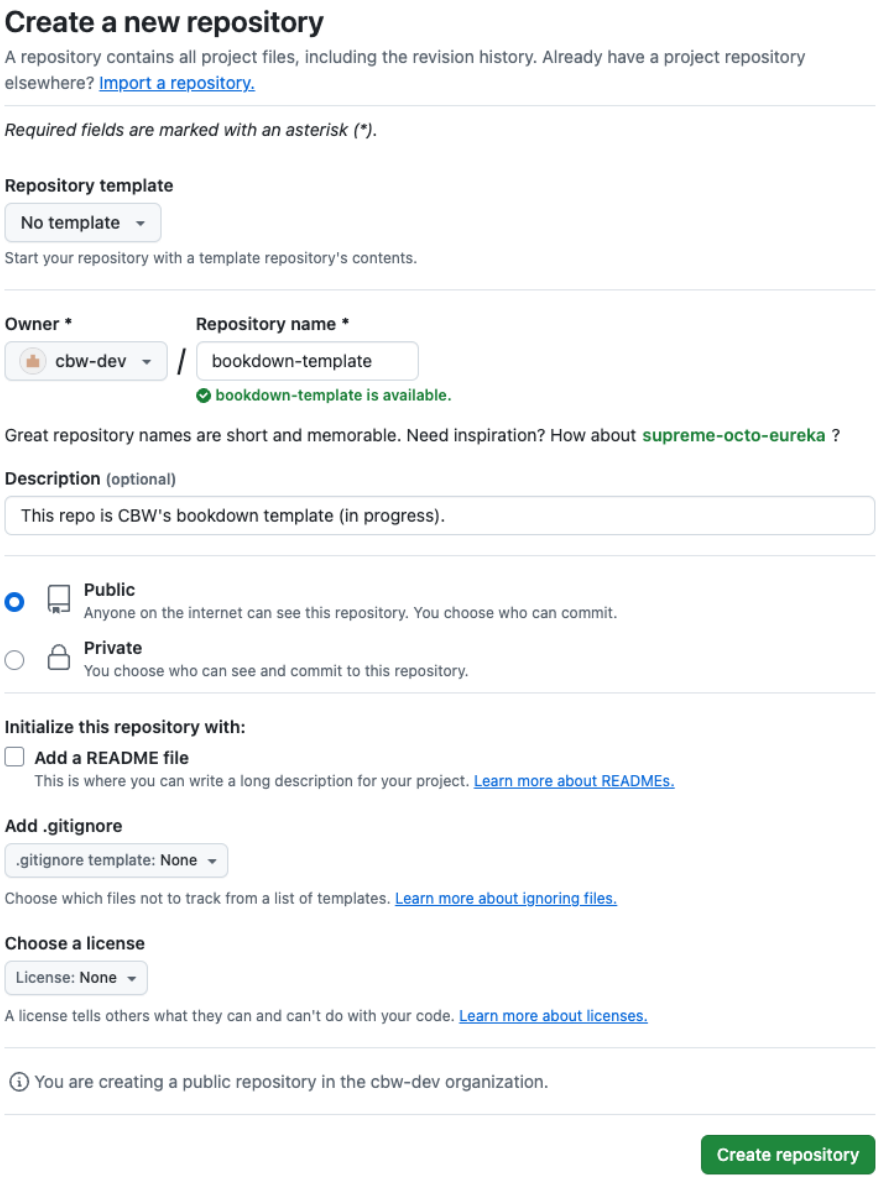
1. Go to https://github.com/cbw-dev (CHANGE?) and scroll to your repositories.

2. Click the green "New" button to the right of the repositories search bar.

3. Create the new repository. Give it a *name* and *description*. Select *Public* instead of private, as shown below.

Warning

MAKE NAMING CONVENTION



4. Click the green *Create repository* button at the bottom.

Now, we already have a local project. Now we want it on GitHub, so everyone on your team can make changes to the workshop! Let's make the GitHub connection (i.e let's add our local code to GitHub!)

11.1.1 How to Make the Git Connection (Adding your Local Repo to GitHub)

After the previous step, you will be brought to this page.The only things that will differ are the name of the repo.

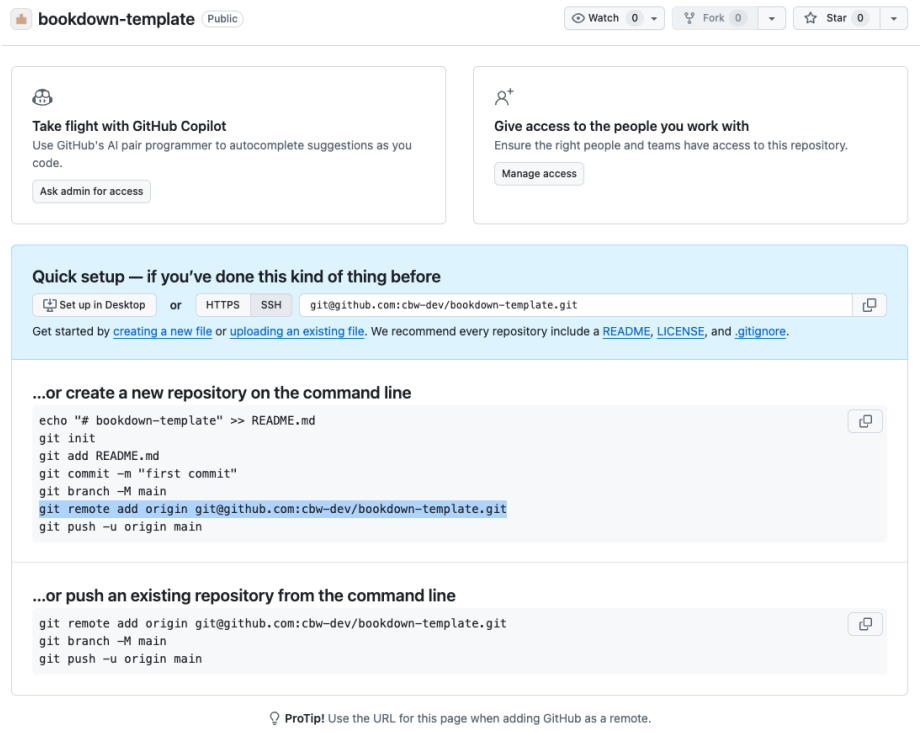


Figure 11.1: landing page after making a git repo

5. Open Terminal (Mac) or Command Prompt/Powershell (Windows).
6. Go to where we created the bookdown project.
7. Once inside the folder with the project. Let's make the git repo. First we initialize: `git init`. (Put this into terminal and press enter.)
8. Let's add all the files: `git add *`
9. Let's commit these files, with a descriptive message to help make it clear to others what we just did. For now, our message can be simple: `git commit -m "first commit"`. (Put this into terminal and press enter.)
10. Next, put this into terminal and press enter: `git branch -M main`.

11. **Important:** This step is why I highlighted that specific text above. Copy that command, and put it into terminal. Generally, it will look something like this: `git remote add origin git@github.com:cbw-dev/NAME-OF-YOUR-REPO.git`

12. Next, put `git push -u origin main` into terminal and press enter.

All the steps are shown below.

```
jqiu@ML8034-JQIU ~ % cd Documents/CBWgithub/cbw-dev-templates-docs/bookdown-template
jqiu@ML8034-JQIU bookdown-template % git init
Initialized empty Git repository in /Users/jqiu/Documents/CBWgithub/cbw-dev-templates-docs/bookdown-template/.git/
jqiu@ML8034-JQIU bookdown-template % git add *
jqiu@ML8034-JQIU bookdown-template % git commit -m "first commit"
[main (root-commit) 44aae1d] first commit
 15 files changed, 266 insertions(+)
  create mode 100644 01-intro.Rmd
  create mode 100644 02-cross-refs.Rmd
  create mode 100644 03-parts.Rmd
  create mode 100644 04-citations.Rmd
  create mode 100644 05-blocks.Rmd
  create mode 100644 06-share.Rmd
  create mode 100644 07-references.Rmd
  create mode 100644 README.md
  create mode 100644 _bookdown.yml
  create mode 100644 _output.yml
  create mode 100644 book.bib
  create mode 100644 bookdown-template.Rproj
  create mode 100644 index.Rmd
  create mode 100644 preamble.tex
  create mode 100644 style.css
jqiu@ML8034-JQIU bookdown-template % git branch -M main
jqiu@ML8034-JQIU bookdown-template % git remote add origin git@github.com:cbw-dev/bookdown-template.git
jqiu@ML8034-JQIU bookdown-template % git push -u origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 11 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (17/17), 5.74 KiB | 2.87 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cbw-dev/bookdown-template.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
jqiu@ML8034-JQIU bookdown-template %
```

Figure 11.2: all the git steps typed out into terminal with results

11.2 Updating GitHub via RStudio

Now, close your RStudio session, and reopen it.

Now, we will be able to see a Git window in the top right. Click "Git" to open this window.

Let's say we only edited `index.Rmd`, now we see the newly edited files. (Do not worry too much about `.DS_Store` and `.gitignore` do.) Let's try to push this change to GitHub.

13. Select all the edited files.

14. Then, click the Commit button, which appears above your selected items. A window pane will appear (shown below).

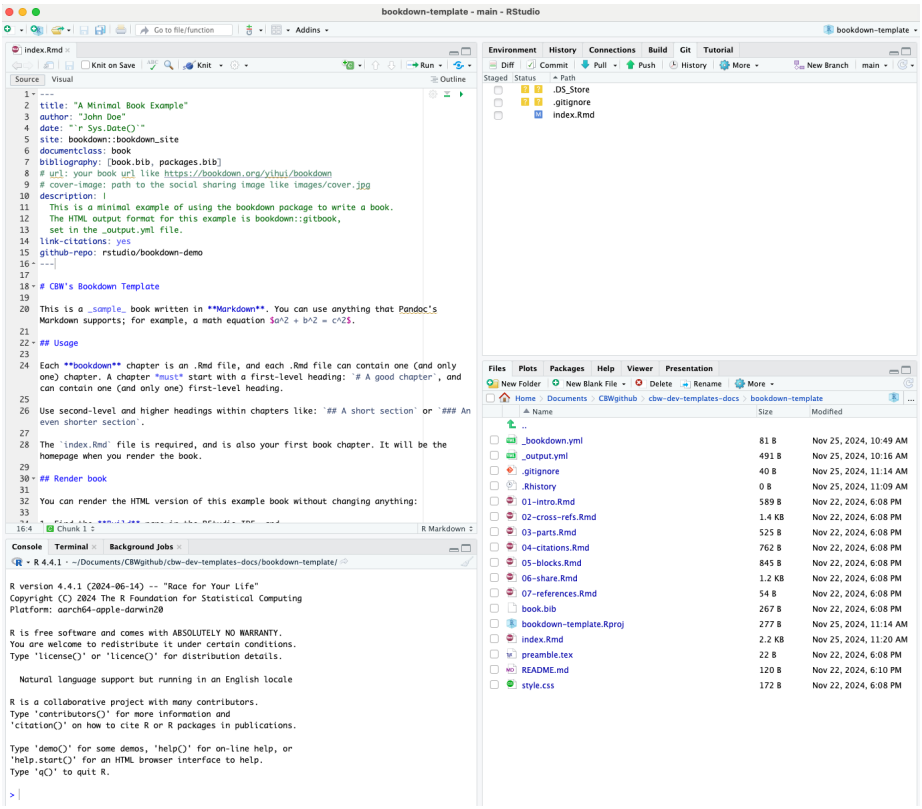


Figure 11.3: RStudio with Git window open

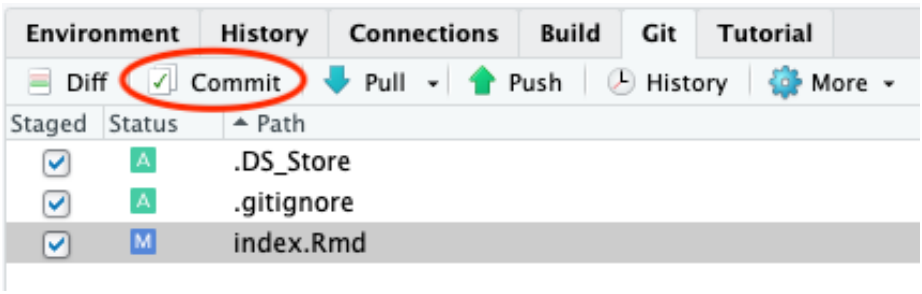


Figure 11.4: selected files in the git window and the commit button highlighted

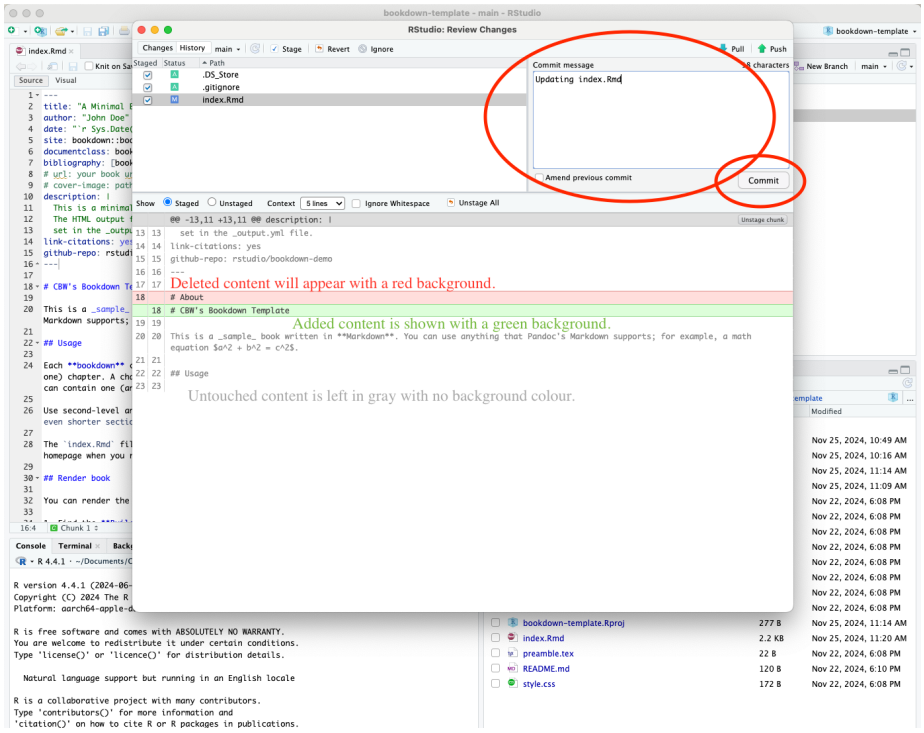


Figure 11.5: git commit window pane

15. Add a commit message in the corresponding box, and then press commit below it.

16. A new window will show up, detailing your updates. Close this window and then press **Push** to push your updates to GitHub.

Figure 11.6: post git commit window

Now, we're done! We should see the updates on GitHub now. Also note, if we ever want to pull updates from GitHub, there is also a **Pull** button in the Git window within RStudio!

Chapter 12

How to Deploy Your Workshop Website

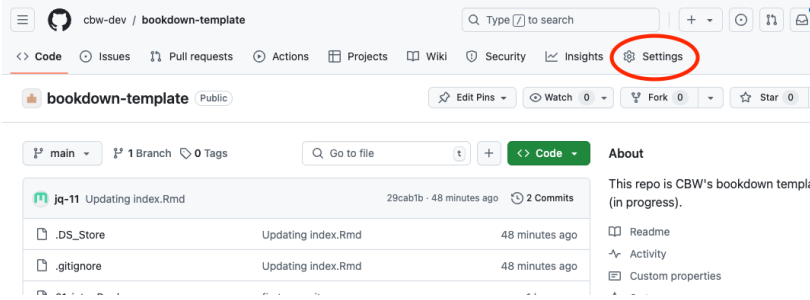
Let's recap.

We've made a bookdown project that builds into a website. We've reconfigured the output to go to a folder called "docs" (output_dir: "docs"). We've pushed our content onto github, and also made a ".nojekyll" file, which we placed into docs.

Now in our ./docs folder, we have a bunch of html files that make up our website. We want GitHub to look at these files in the docs folder and host the website for us!

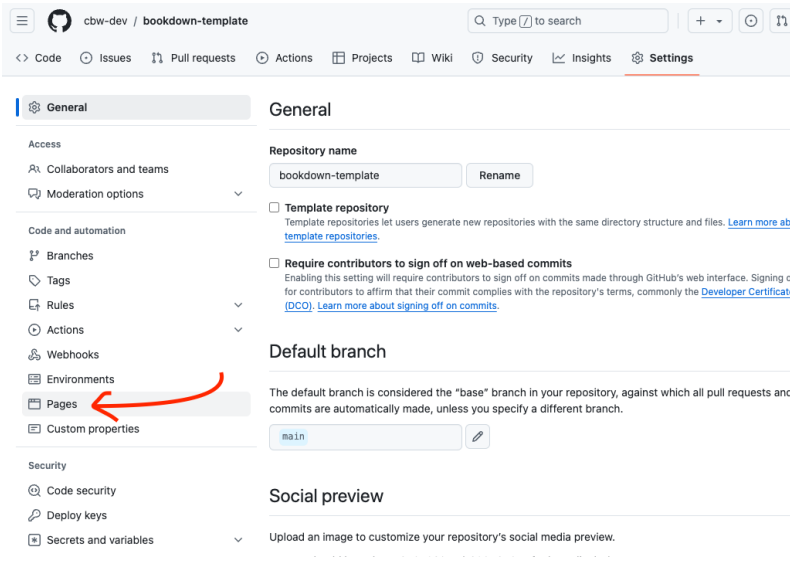
We deploy our website using GitHub pages. GitHub pages uses jekyll, so the .nojekyll file tells it to no longer rely on jekyll. Now, all we need to do is tell GitHub pages to deploy (create/update the website) from our docs folder.

1. Go to your repo on GitHub.

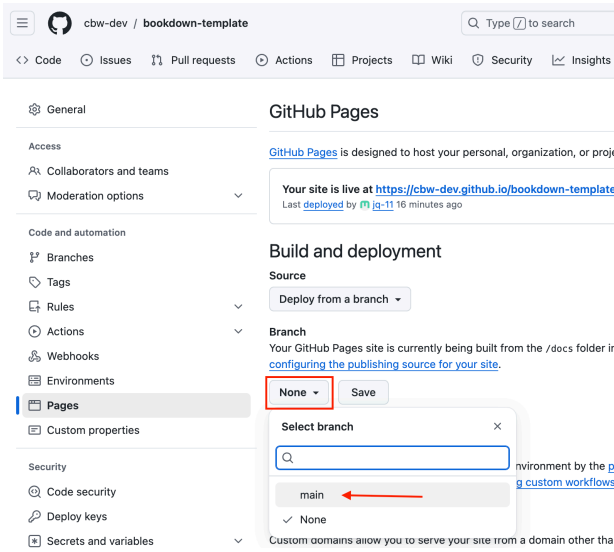


2. In the top navigation bar, select settings.

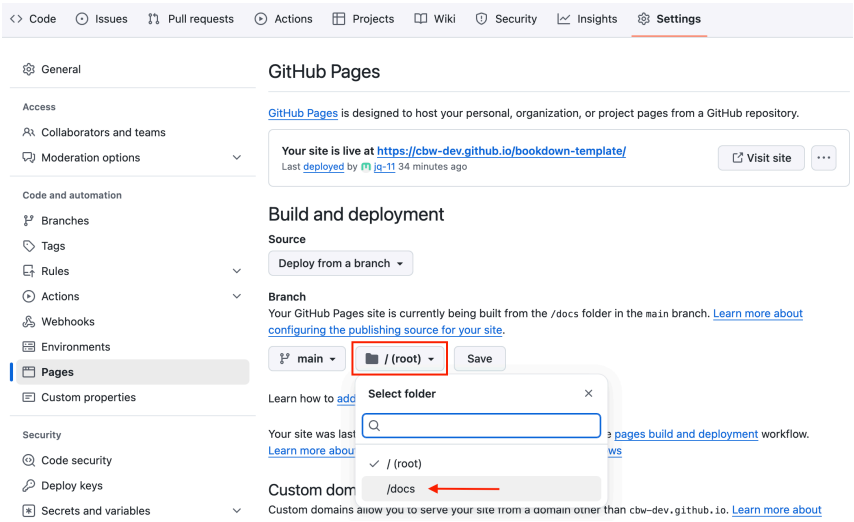
3. Then, go to the pages sidebar.



4. "Deploy from a branch" is already selected, which is what we want. We must change the branch from "none" to main.



5. Then, change the folder from `/ root` to `/docs`. Then press save.



Great! Now we're waiting on the page to build and deploy, which should take less than a minute.

To see updates, go to the **Actions** page (found along the top navigation bar. This will help you understand how the deploy is working, and if it succeeded or failed.

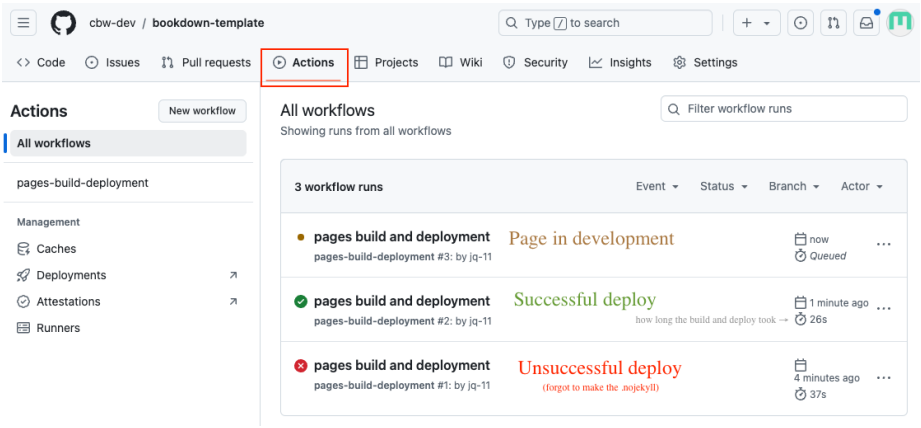
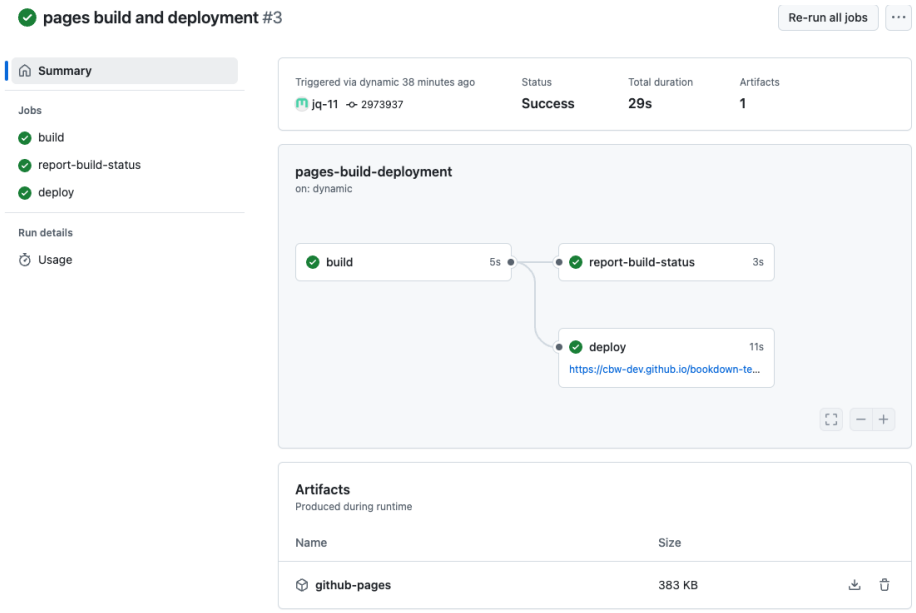
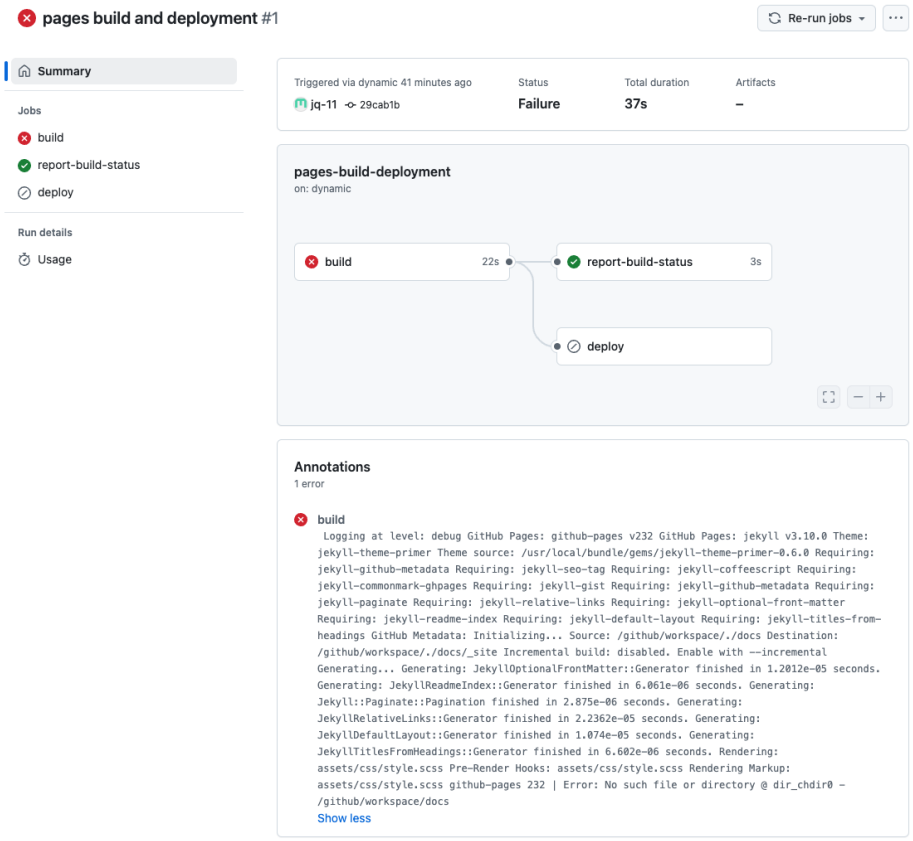


Figure 12.1: Image showing the different possibilities of deploy a github page

You can click **pages build and deployment** for updates. It will give you errors (which may not be very clear) or the link of your deployed page!


pages build and deployment #3
Re-run all jobs
Summary
Jobs
build
report-build-status
deploy
Run details
Usage
Triggered via dynamic 38 minutes ago
jq-11 2973937
Status
Success
Total duration
29s
Artifacts
1
pages-build-deployment
on: dynamic
build
5s
report-build-status
3s
deploy
11s
https://cbw-dev.github.io/bookdown-te...
Artifacts
Produced during runtime
Name
Size
github-pages
383 KB



Click around to explore more!

Chapter 13

Appendix [Links for Developers]

For the developers of this guide, here are helpful sources.

13.1 GitHub Repo

The Bookdown Docs are hosted from the cbw-dev/bookdown-docs public repository.

13.2 Diagram Links

Most diagrams shown on these docs **should** be in this Google folder.

13.2.1 How to Edit a Google Drawing

1. Open the drawing.
2. Edit.

   > Tip: Editing size of the Google Drawing is very easy when you drag the diagonal lines in the bottom right corner.

3. To publish, go to "File > Share > Publish to Web > Embed". Copy the "Medium" *(optional)* image size HTML code. Paste where you want it to appear in your .Rmd file.

4. Add alt text to the img embed code: Add `alt="Alt Text"` after `src = ...` and before `>`.

   Warning:

   If you're putting a screenshot into your Google Drawing, the resolution of the image will look very poor when it is rendered. You can fix this by simply making the image extremely big.

Note: Lots of the GitHub clone/deploy/RStudio explaining images are currently locally in the repository, and not on the Google drive.

References (the below citation list is auto-generated since we showed examples of citations)

Bibliography

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2024. URL https://github.com/rstudio/bookdown. R package version 0.41.