This is an interactive self-paced presentation. Shapes like these are clickable, and will take you different places.

(except these specific ones, they're just for illustration. Click the green arrow at the bottom to advance)

Make a website

Using RMarkdown and Deploying to Github
4th ESLR Workshop

Dr Chrissy Cuskley (christine.cuskley@ncl.ac.uk) | ccuskley.github.io | @nerdpro

This tutorial will walk you through creating a website using RMarkdown, which we saw in the DataVis workshop, and deploying it using Github. The goal today is for you to make and deploy your website from scratch, so rather than put the tutorial within RMarkdown, you'll do everything yourself from nothing.

a) To make the content we'll use RMarkdown

b) To host it, we'll use GitHub

These are two entirely separate steps, and they don't need to go together!

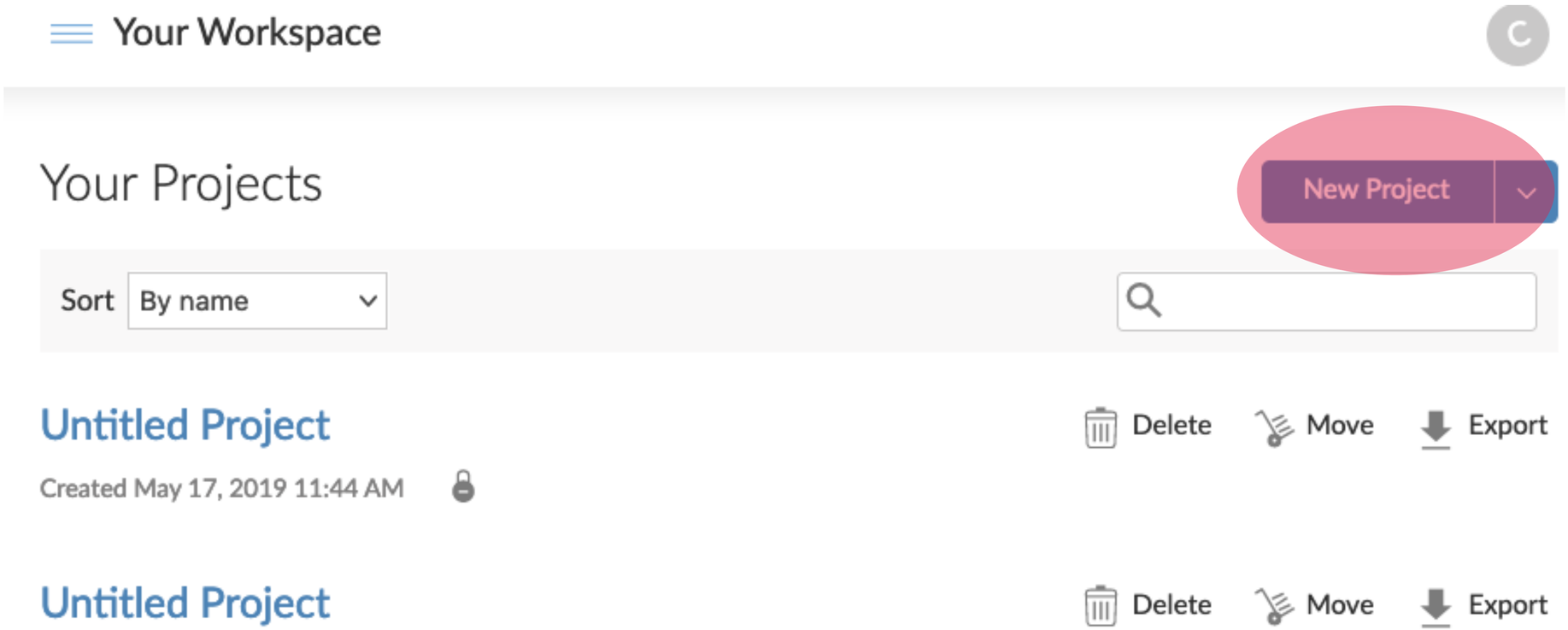
- If you just want to make a website but already have a place to host it, go straight to the RMarkdown part. Put your code wherever it needs to be for your hosting service to show it.
- If you already have a nice website (html etc), but just want a nicer .github.io address, go to the github bit.
- If you have neither, start with RMarkdown.

Make a site with RMarkdown

Get your site on github.io

Start by creating a new R project in RStudio Cloud (or locally if you prefer* – you'll have to download it eventually anyway). Start by naming it "my website" or something similar.

*In this case, you'll just do File>New Project…

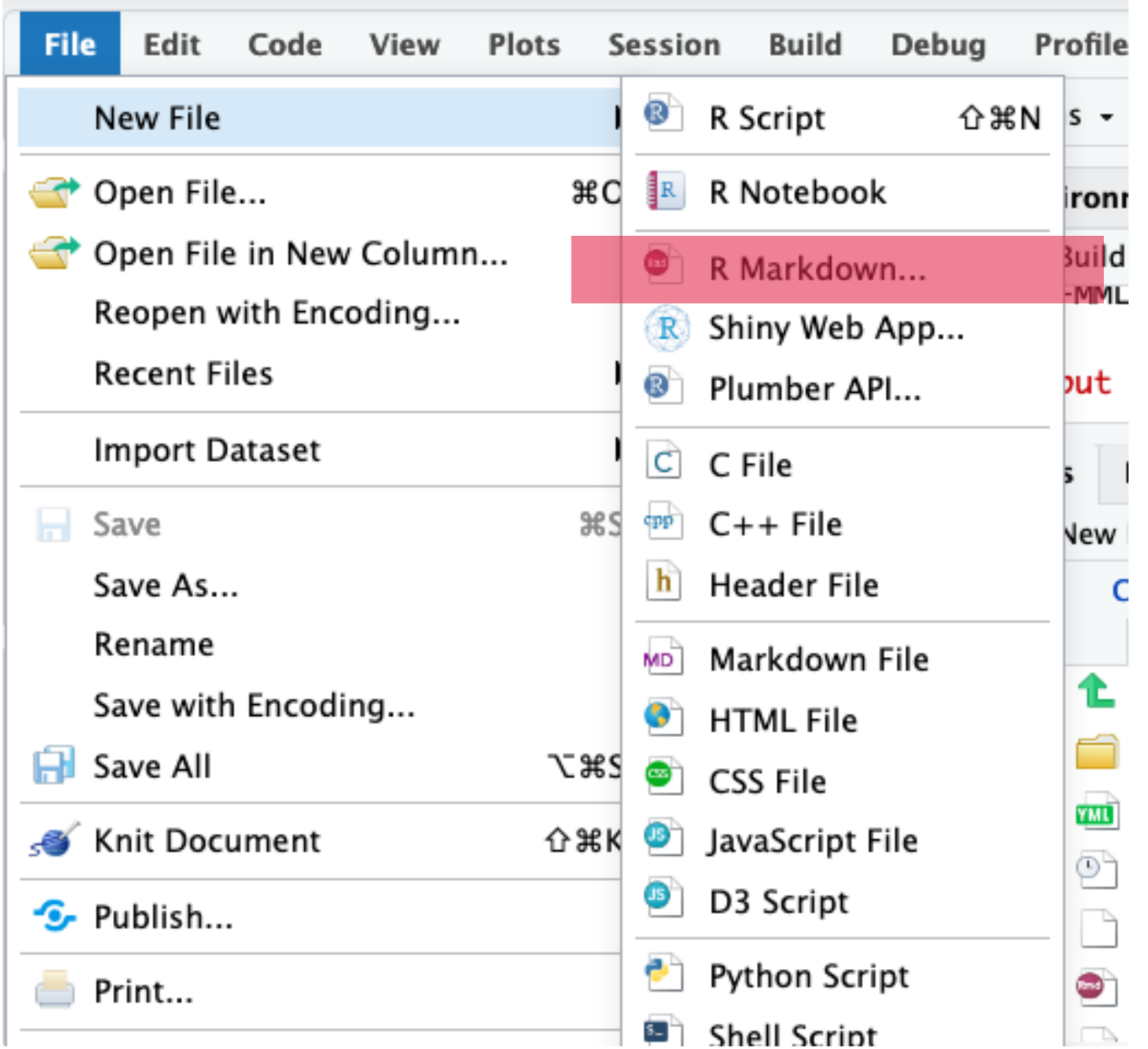


You'll need a minimum of two files for your website: an index.rmd file, which will generate the index.html page that will be the splash page for your site, and a _site.yml file, which will determine the basic styling of both the index.html page and any other pages you wish to generate.

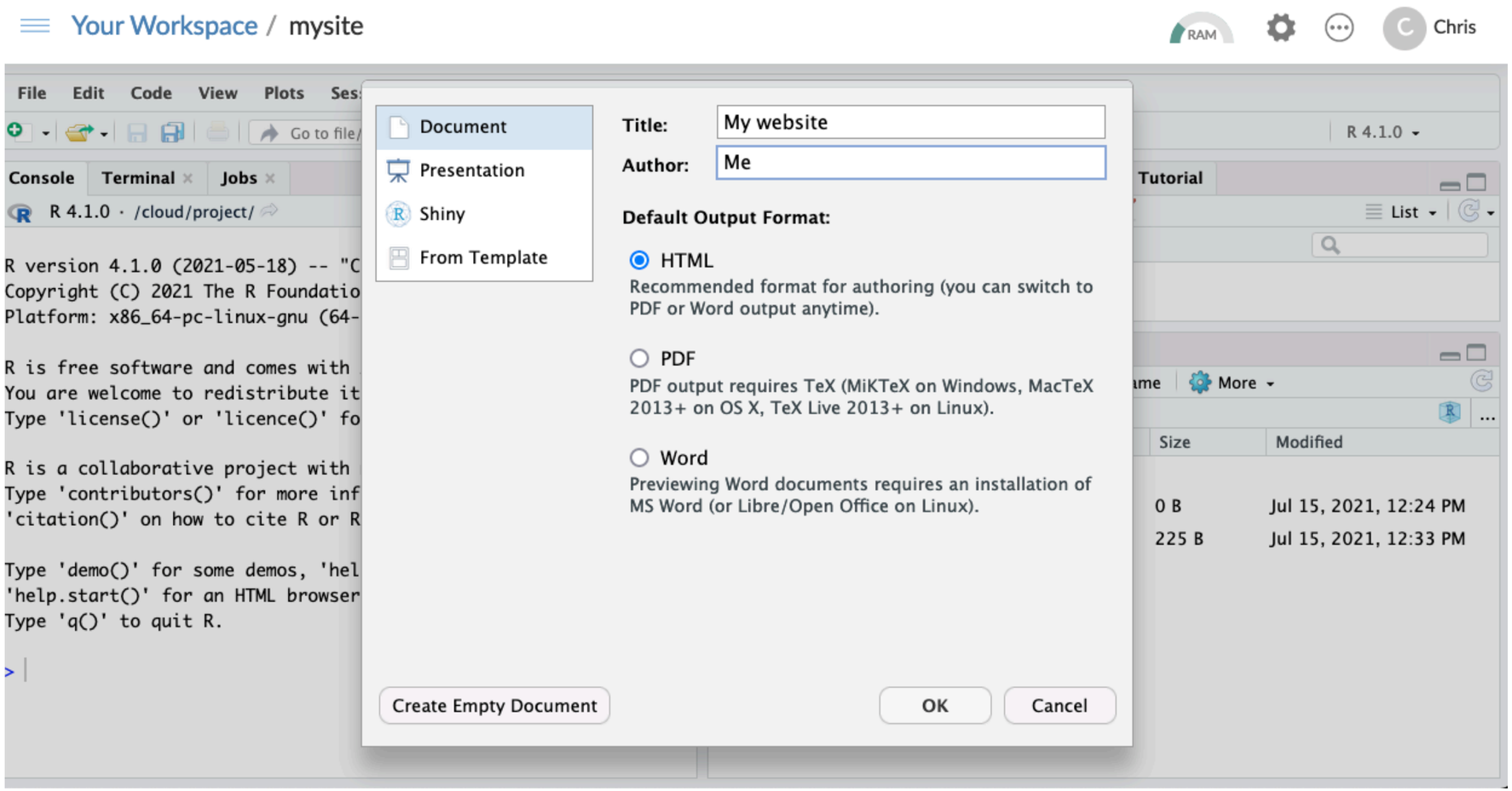
If you're just making a personal website, I recommend doing a single page site, but you can easily add additional pages that give further information about particular projects or resources you've developed – the _site.yml file will make sure these all have consistent styling.

- **To create the Rmd file, go to File>New File>R Markdown... ***

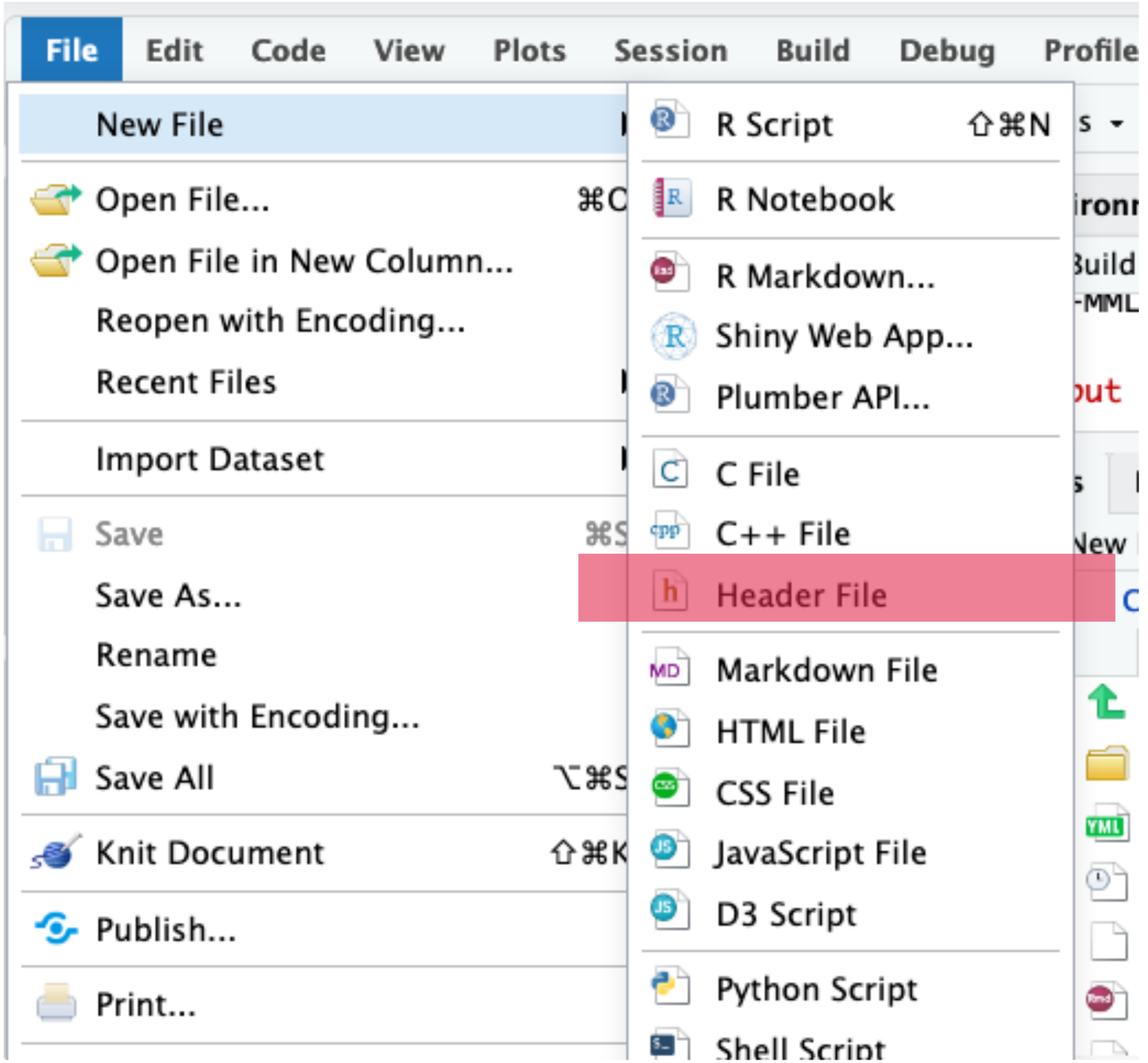
*Note that when you first try to make a new .rmd file in a new project in RStudio Cloud, it will pop up with a warning that markdown requires packages that aren't installed. Click ok to install these; once you do, you may have to do File>New File> R Markdown... again.



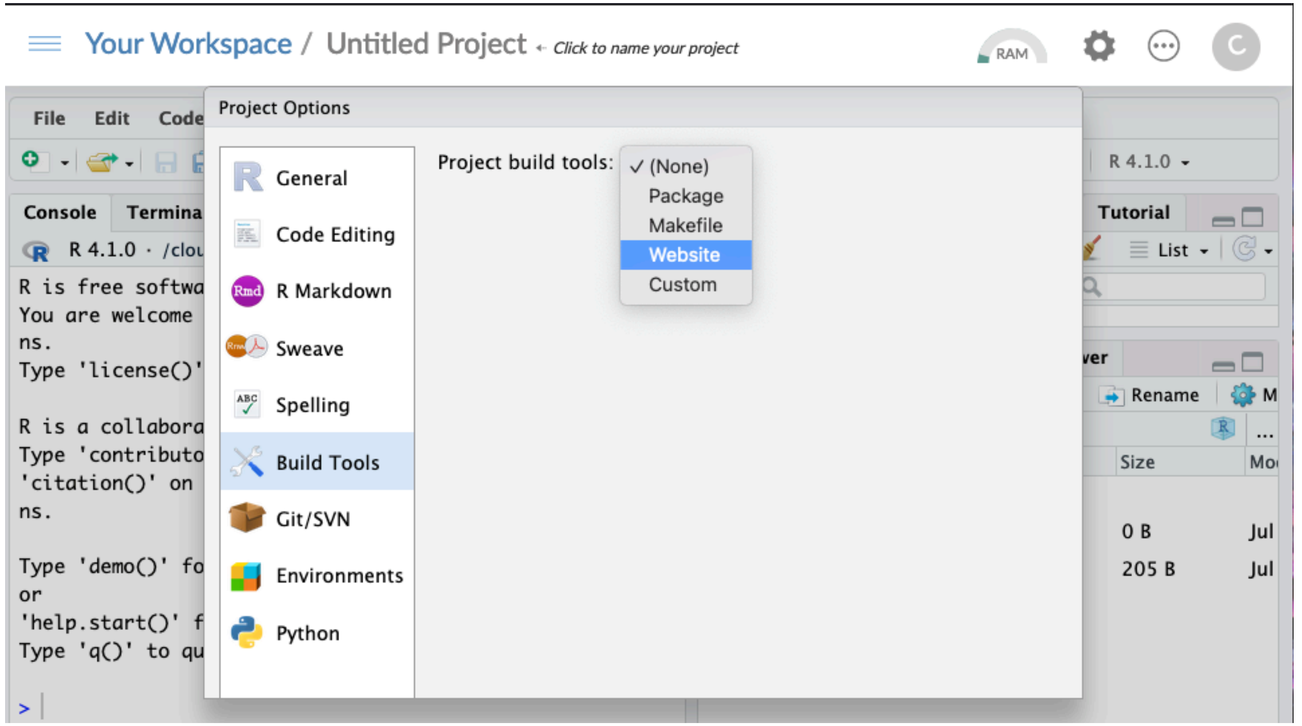
- Choose whatever title/author name you like (you can easily change this later), and make sure to choose HTML output.
- Save the file as index.rmd – it's fine that nothing is in it yet.



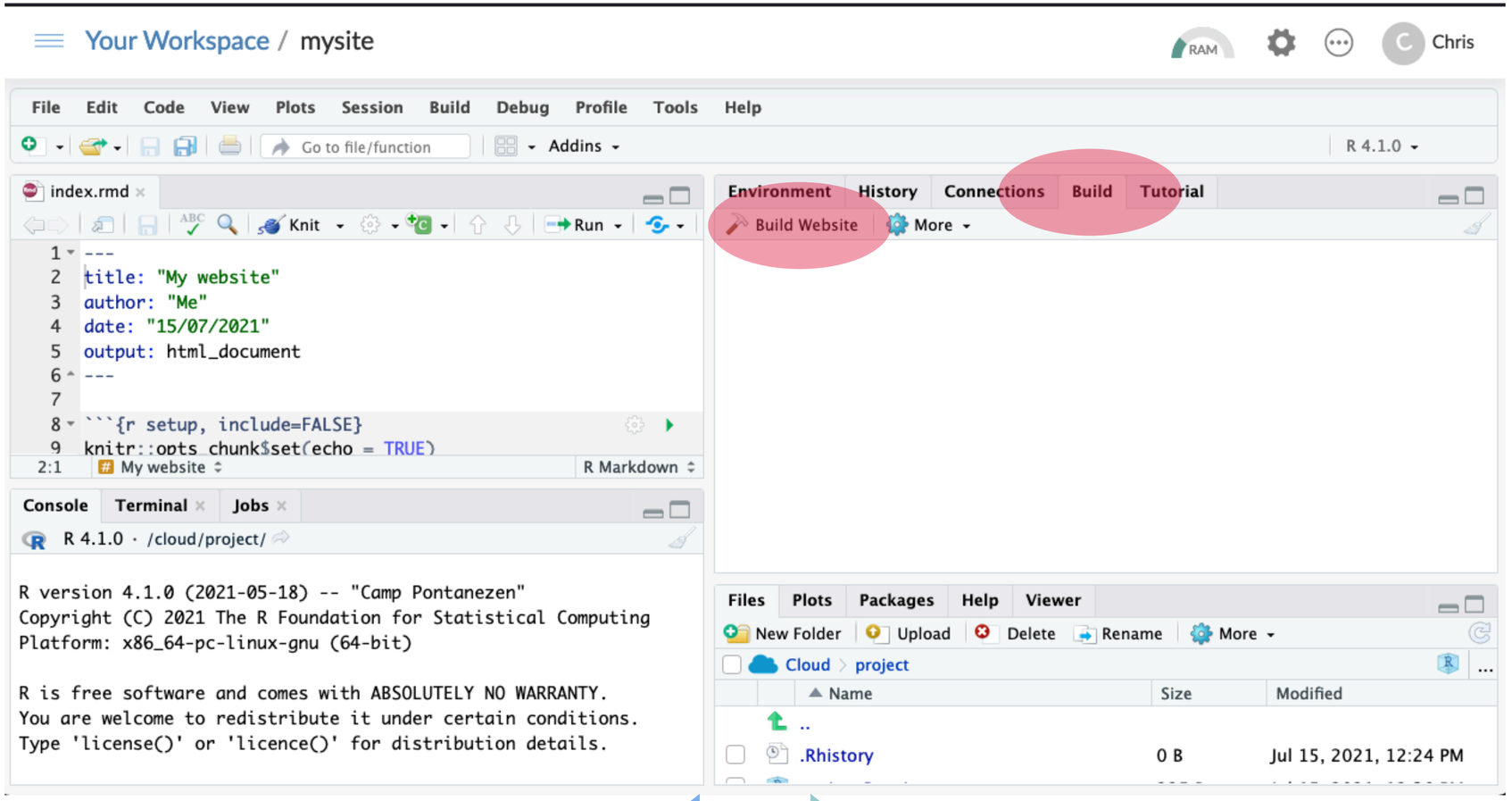
- For the .yml file, go to File>New File> Header File.
- Save this file as _site.yml (again, it will be blank at this point).



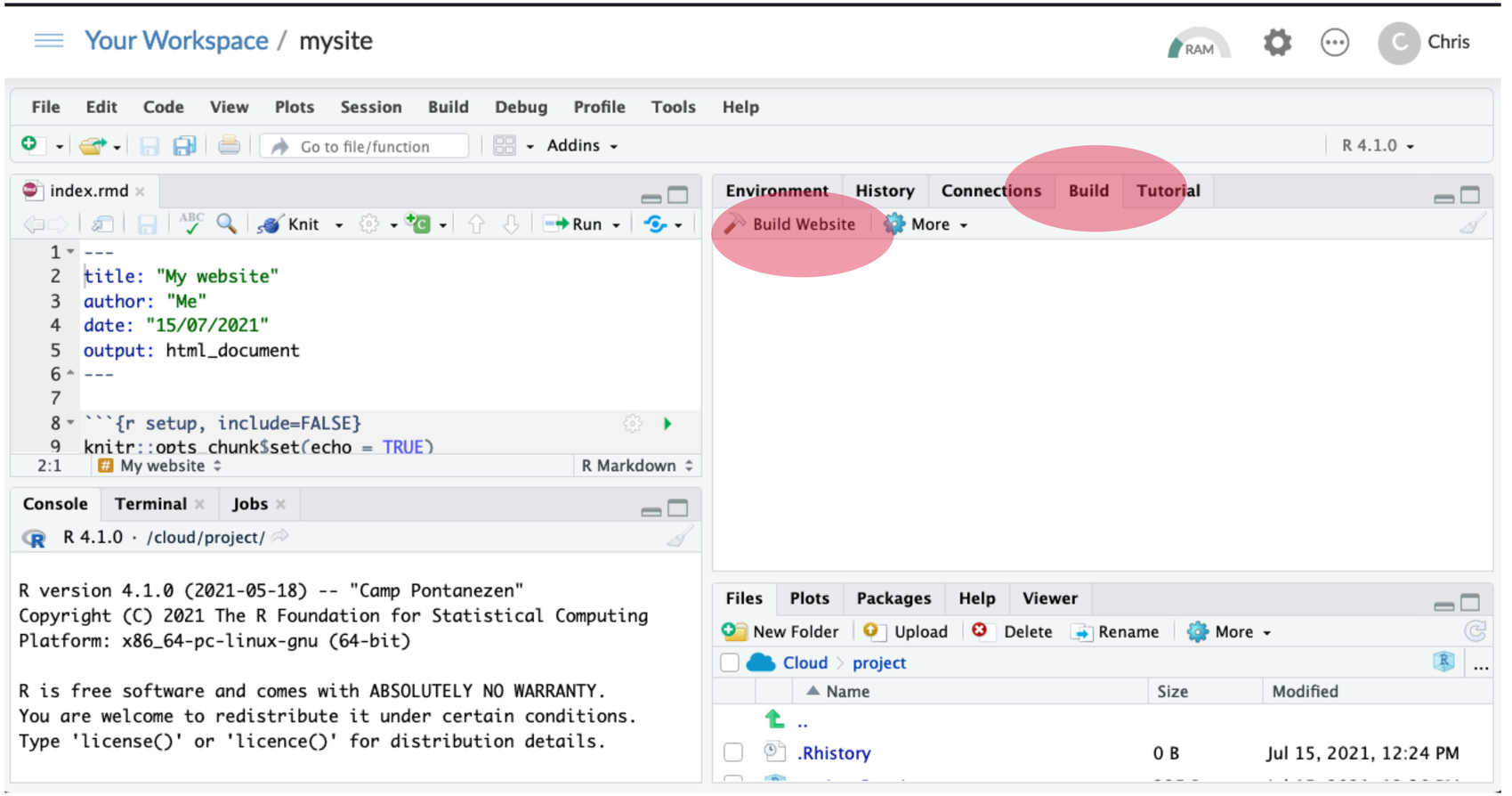
Now, we need R/RStudio to understand that this project is a website. Go to Build > Configure Build Tools. Where it says “Project build tools”, select “Website” from the dropdown menu. Set the site directory to (Project Root) and scroll down and click OK.



Now, on the upper left pane in RStudio there's a new tab called "Build" – if you click on this there's a hammer icon that says "Build Website". This button is going to take what we put in our .rmd file using the styling in the .yml file, and compile it into HTML and CSS, which is what browsers read to display content (in fact, if you are using RStudio Cloud, you are looking at HTML and CSS – and some other stuff – that is made to look like the RStudio desktop client).



Clicking the 'Build Website' button compiles your website much like the 'Knit' button compiles markdown – the main difference is that it's going to compile everything into a _site folder that will create all the code you need to put your website somewhere where others can actually see it, all in one place. Give it a try.



Your website (or what you have of it so far) should have popped up in a separate window. At this point it looks like a plain markdown document, and probably has the default RMarkdown template content in it. The next step is to change all that. Let's start with the styling. Add the text on the left to your .yml file, altering the name and title as appropriate.

Note: the colouring of the text should roughly match what's on the left; if it doesn't, the yml file won't compile properly.

```
---
name: "My name"
navbar:
  title: "A title"
  left:
    - text: "Home"
      href: index.html
  right:
    - icon: fa-twitter fa-lg
      href: https://twitter.com/<your
handle>
    - icon: fa-github fa-lg
      href: http://github.com/
<YOUR_GH_NAME>/
output:
  html_document:
    theme: readable
---
```

Let's walk through what all this means. The name is pretty self explanatory, but things get interesting with the navbar.

This will add a navbar to every page generated for your site that has the same menu items. The text displays a menu item something, and href ties a link to it links it (this stands for hypertext reference, and can be another webpage, or somewhere within your own page – more on this later). I have my page links on the left and external links on the right.

So all in all, this will generate a navbar with my name on the left, followed by a menu item that says "Home" that redirects to index.html; on the left, it will create two icons that redirect to a Twitter and Github page respectively. These use free Font Awesome icons and there are lots of possibilities – e.g., if you have an Orcid ID you'd like to link to, there's an icon for that.

```
---
name: "My name"
navbar:
  title: "A title"
  left:
    - text: "Home"
      href: index.html
  right:
    - icon: fa-twitter fa-lg
      href: https://twitter.com/<your
handle>
    - icon: fa-github fa-lg
      href: http://github.com/
<YOUR_GH_NAME>/
output:
  html_document:
    theme: readable
---
```

Finally, the `output` section tells it to build an html document with the `readable` theme. This theme will be what determines the overall look of your site – there are quite a few choices here on Bootswatch – find one you like and fiddle with the look.

Note that you can get a lot more customisation if you want to dig into something like Jekyll, but the Bootswatch themes get the job done.

```
---
name: "My name"
navbar:
  title: "A title"
  left:
    - text: "Home"
      href: index.html
  right:
    - icon: fa-twitter fa-lg
      href: https://twitter.com/<your
handle>
    - icon: fa-github fa-lg
      href: http://github.com/
<YOUR_GH_NAME>/
output:
  html_document:
    theme: readable
---
```

Now let's move onto content. What should be in a professional academic website? This depends a bit on your field and where you are in your career. Here's what I'd recommend:

- **About:** what you do generally, where you are working now; maybe a bit about where you've been. A photo of you can be useful here; you may also include things like pronouns.
- **Contact**: this should be more than implicit (i.e., just putting your email address somewhere) – e.g., you might say "I'm happy to correspond via email: myadress@theinternet" or "Get in touch with me on twitter @myhandle"
- **Interests:** This should give some additional information about what you're working on more specifically, and what your general aims are (depending on career stage). Are you looking for a job? Open to collaborations? Recruiting students? Hiring for a project?
- **Outputs:** This might be a bajillion publications, or just a few presentations, or some blogging you're particularly proud of (or all of the above). This will depend on your career stage; the more you've done, the more you'll want to curate this list. Not much to put here yet? Don't worry – even one thing (e.g., a polished dissertation) gives an impression of what you can do.

There's lots more you could have – you might have information about what you teach, or if you're at an earlier stage, what courses you've taken. You might develop resources (e.g., R Packages!) and want to have a separate section for that. Overall, the aim is to keep it relevant, and fairly clean/concise as a single page site.

Now there's a lot you'll want to do to make your content better. First, you'll want to write your custom content: just do this in Rmarkdown.

Click on each of the squares on the right for how to do a few other essential things, but there's a lot you can find out on your own. This is just rmarkdown, so google e.g., "bold font RMarkdown" and you'll get results pretty quickly.

Add sections

Add a photo

Add publications

Add tabs

Add to the navbar

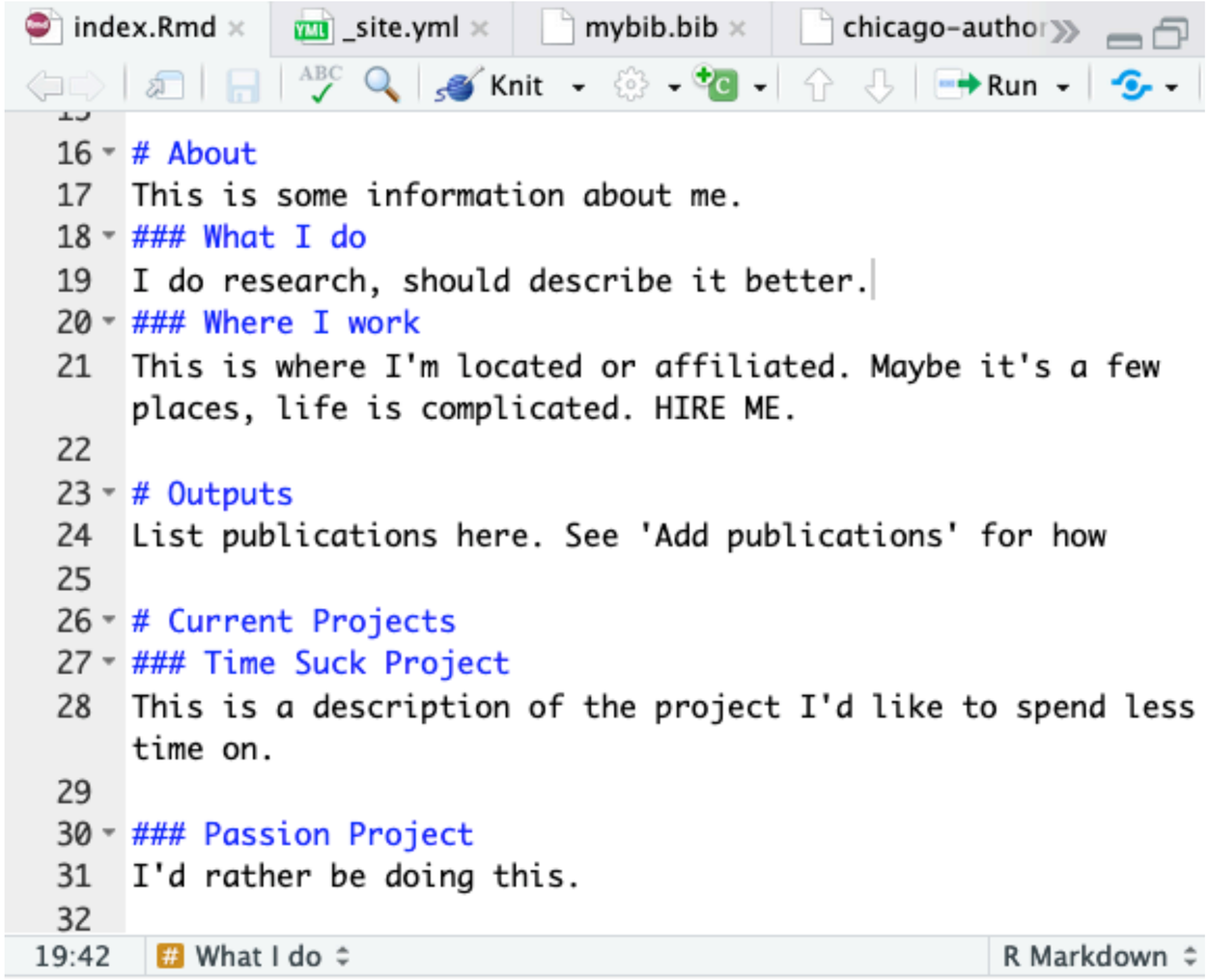
Get your site on github.io

Essentials Menu

Adding Sections

Adding new sections in RMarkdown is as simple as adding a number of hashtags. One hashtag, #, will create an HTML <h1> element, which is the largest header, ## will create an <h2> element, which is slightly smaller, and so on. It's usually not worth going below 5 ##### or <h5>, as this will start to look identical to the main text (which will be converted to an html <p> element).

I'd recommend About, Interests (or Projects), and Outputs at a minimum (contact can be integrated into About)



Essentials Menu

Adding a photo

In the files pane, upload the picture(s) you want to display (if you're working locally, just put it in the project folder).

Add `![Caption](filename.jpg)` where you want the photo to appear.

This might put a gigantic photo that's aligned in a way you don't quite want.

Look at the bookdown docs to troubleshoot size and alignment.

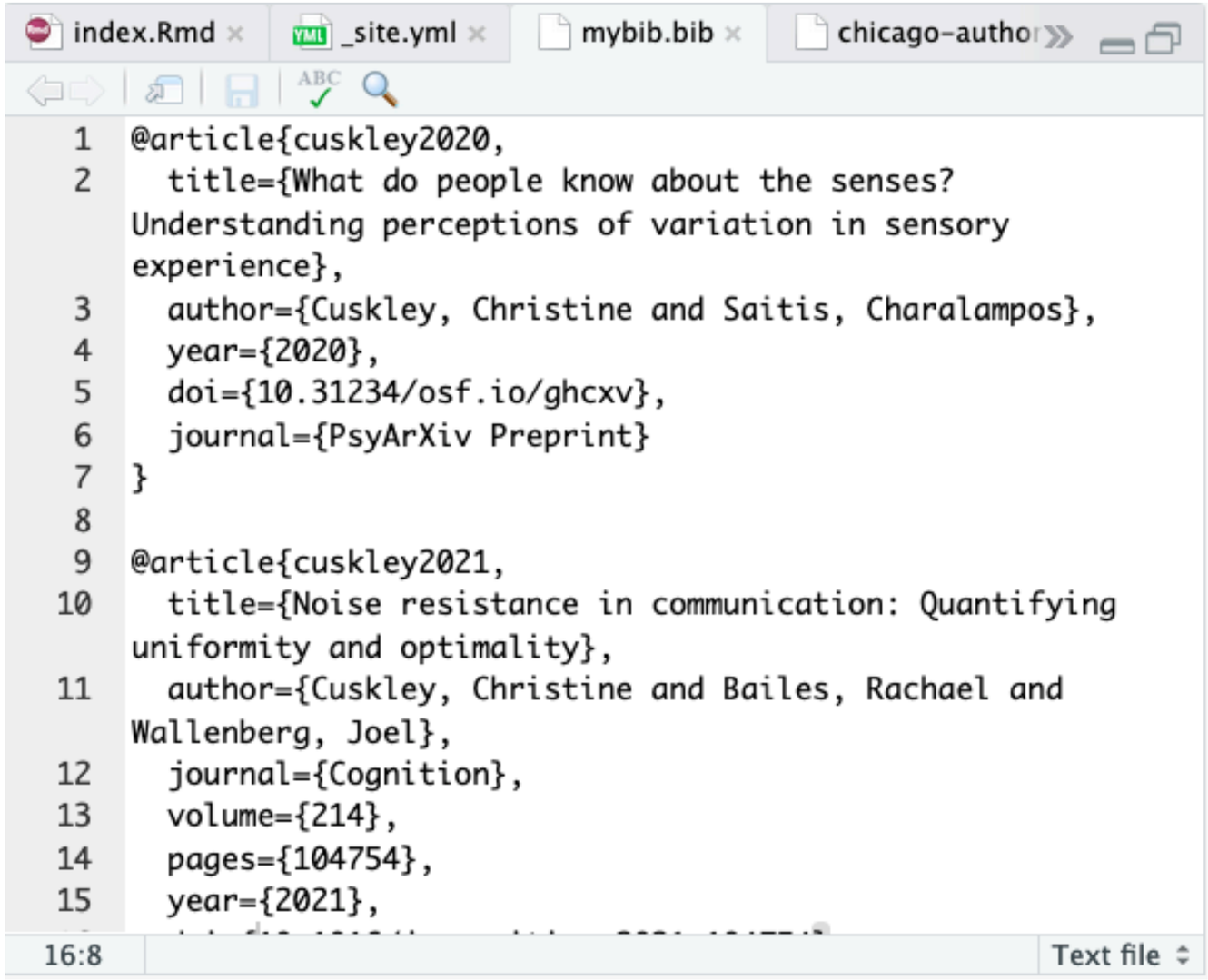
Essentials Menu

Adding publications

You can just type your publications in - you can look at the text formatting in bookdown to get this exact.

However, especially if you have a lot of stuff, this is a pain. I recommend using a .bib file of your outputs instead.

You can generate a .bib file from Zotero or another reference manager. It should look like something this image on the right. Using the files pane, upload it to your project.

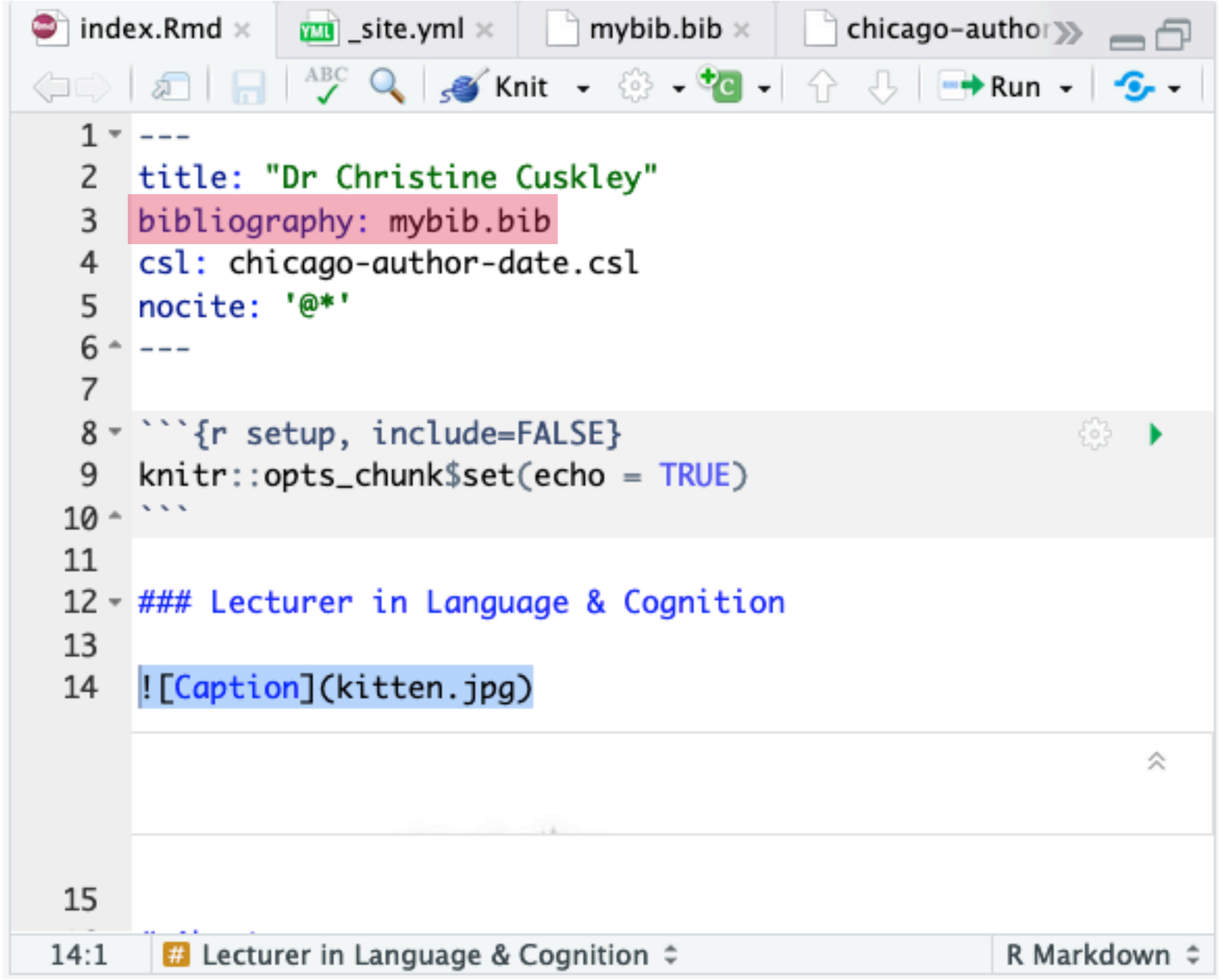


Next >

Essentials Menu

Adding publications

Once your bib file is loaded, you need to tell your markdown to use it. In the header of index.rmd, add `bibliography: mybib.bib` (but use your own bib file name)



< Back

Next >

Essentials Menu

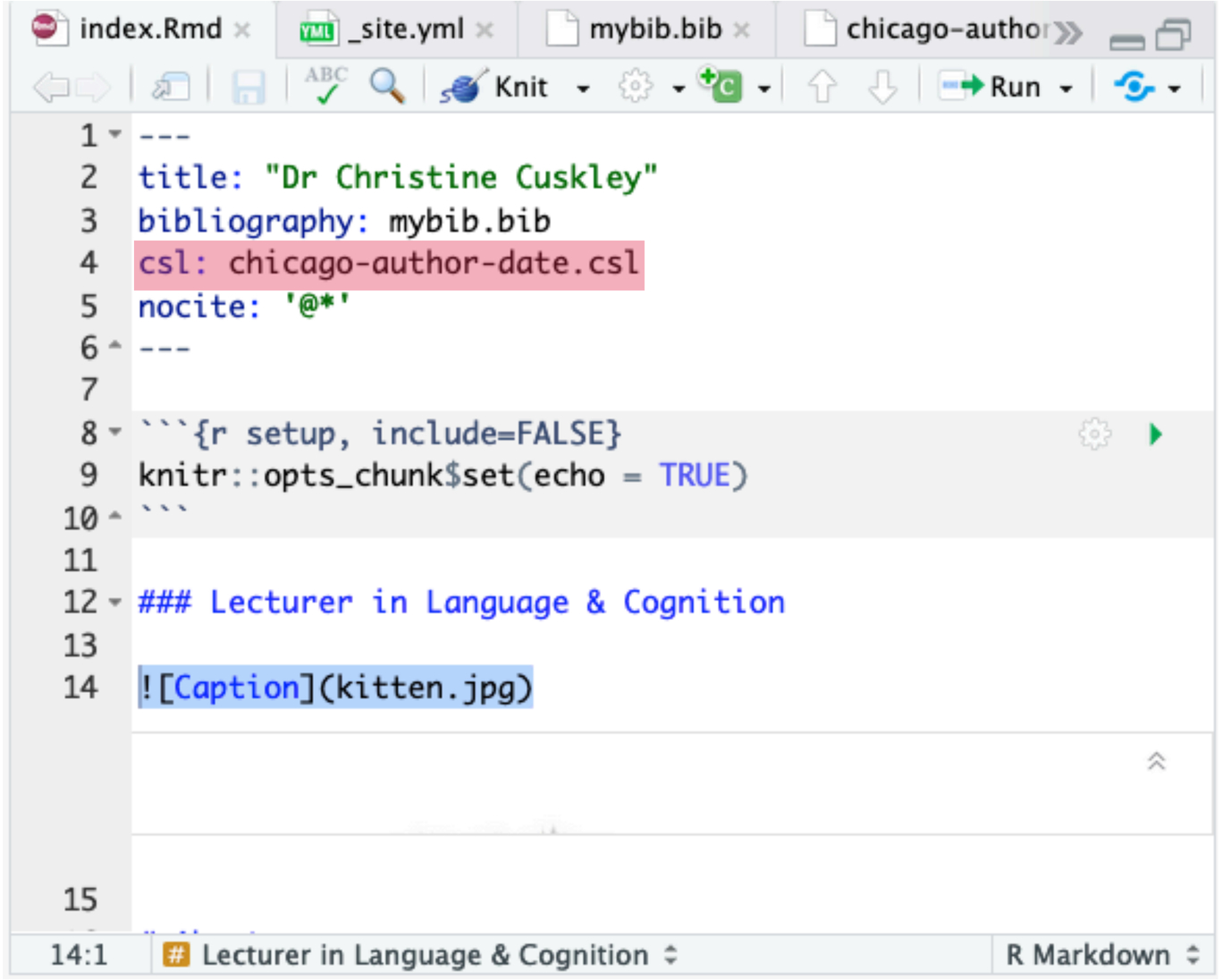
Adding publications

Next, you need to add a style file for your citations.

To do this, go to Zotero and search for the kind of style you want (this will depend on your field).

Download the .csl file (just click in the list) and upload it to your project (or put it in the local folder).

Finally, put `csl: yourstylefilename.csl` in the header of index.rmd



< Back

Next >

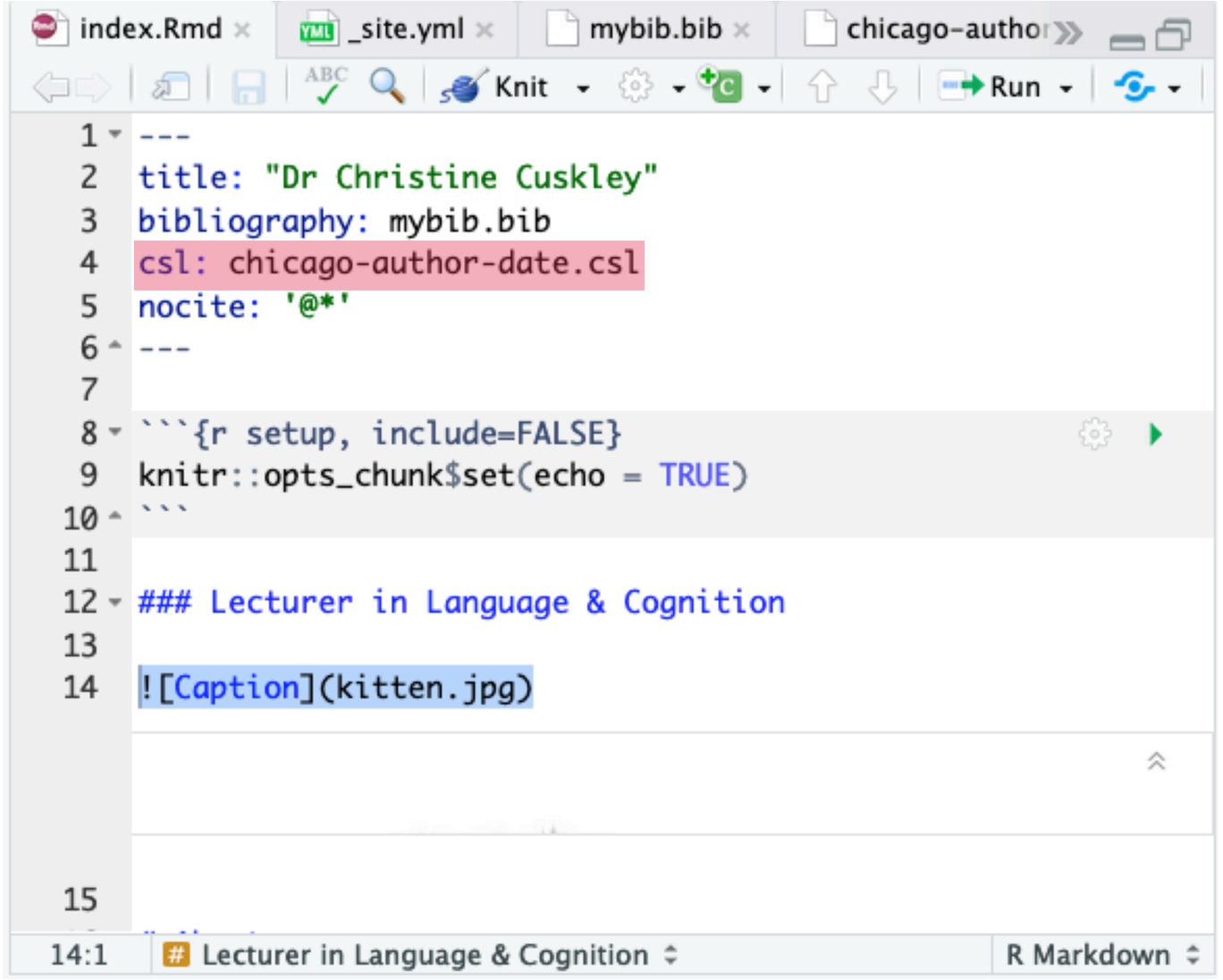
Essentials Menu

Adding publications

There's a slight issue with how reference lists normally work, vs how you want it on your website.

Reference lists generally go in reverse chronological order, showing oldest outputs first

You probably want to show your neweest work first, so we need to make a small edit to the style file.



< Back

Next >

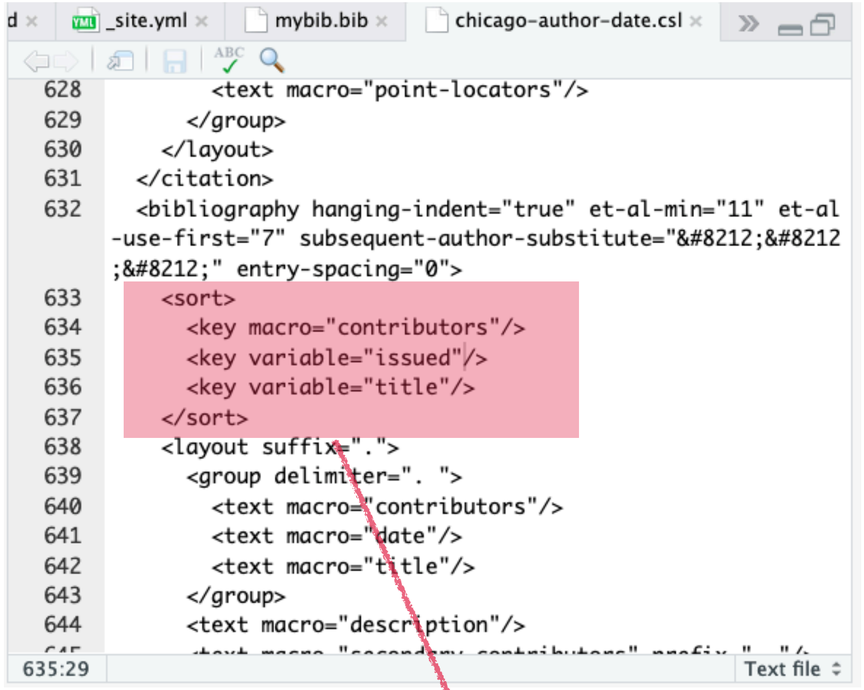
Essentials Menu

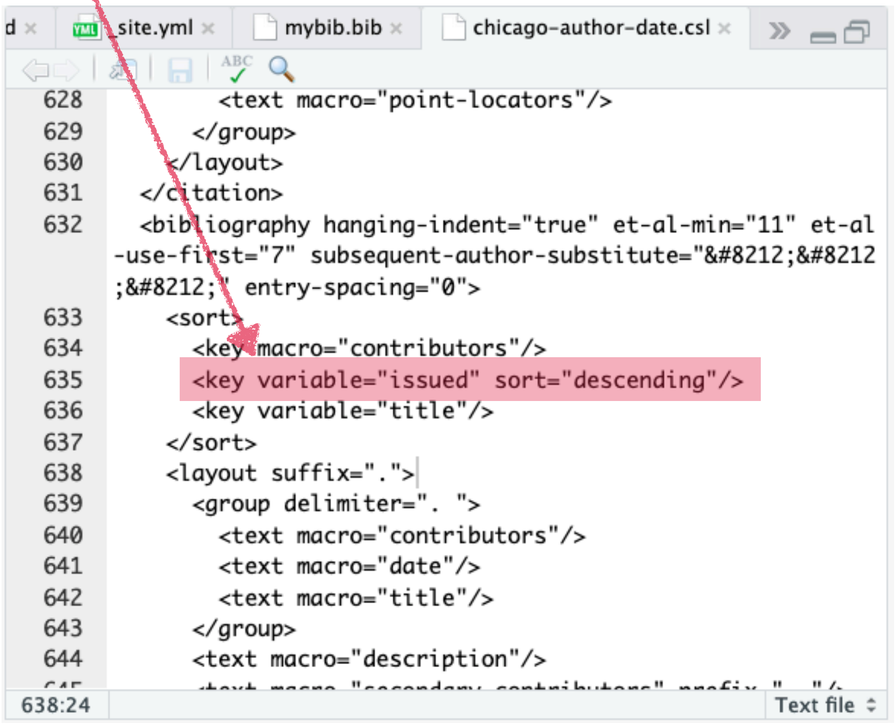
Adding publications

All style files will be slightly different and in scary looking xml. Search for the `<sort></sort>` tag (probably towards the bottom).

Once you've found it, add `sort="descending"` to `<key variable="issued">`

Save the file.





< Back

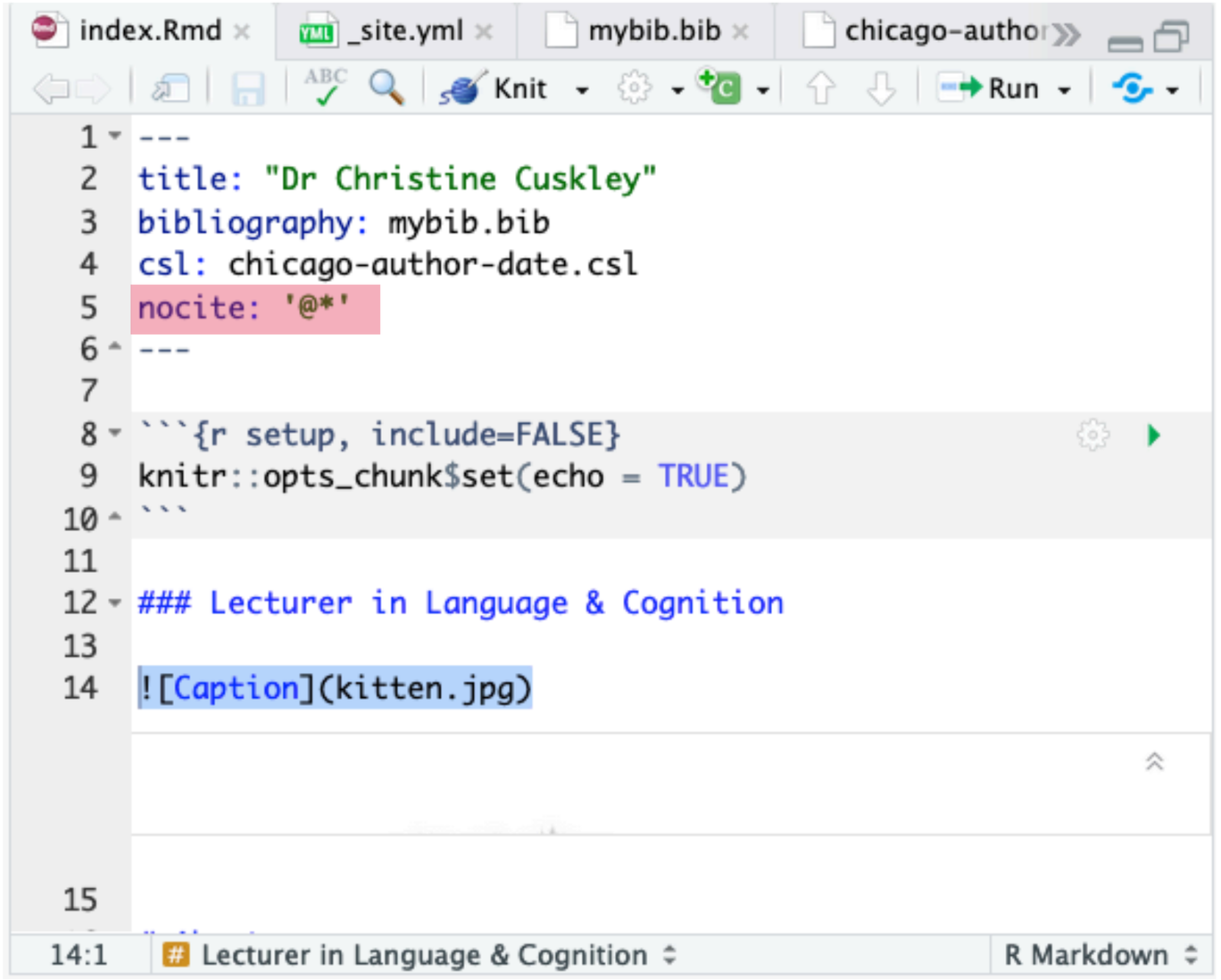
Next >

Essentials Menu

Adding publications

Normally, we'd cite things in text when writing (e.g., Jones, 2020 argues that...), and then reference management software would add this to a reference list.

You want a reference list of everything in your bibfile. So, add `nocite: '@*'` to your header



< Back

Next >

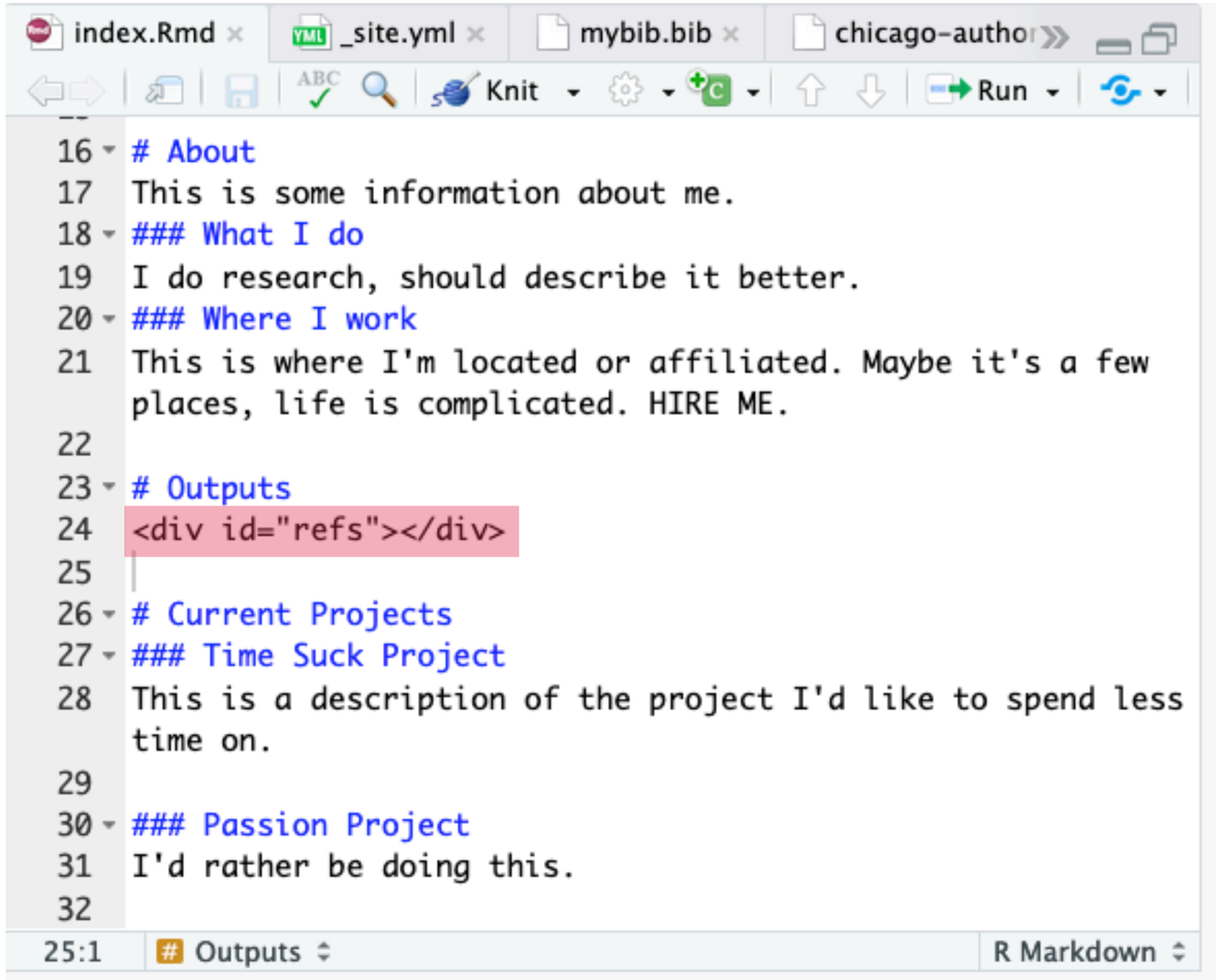
Essentials Menu

Adding publications

Almost there! Now we have to tell RMarkdown where to put the reference list.

Normally, RMarkdown defaults to putting it at the end, but we don't want that here. We have to put a bit of HTML in to tell it to do otherwise. Add `<div id="refs"></div>`

to your outputs section.



< Back

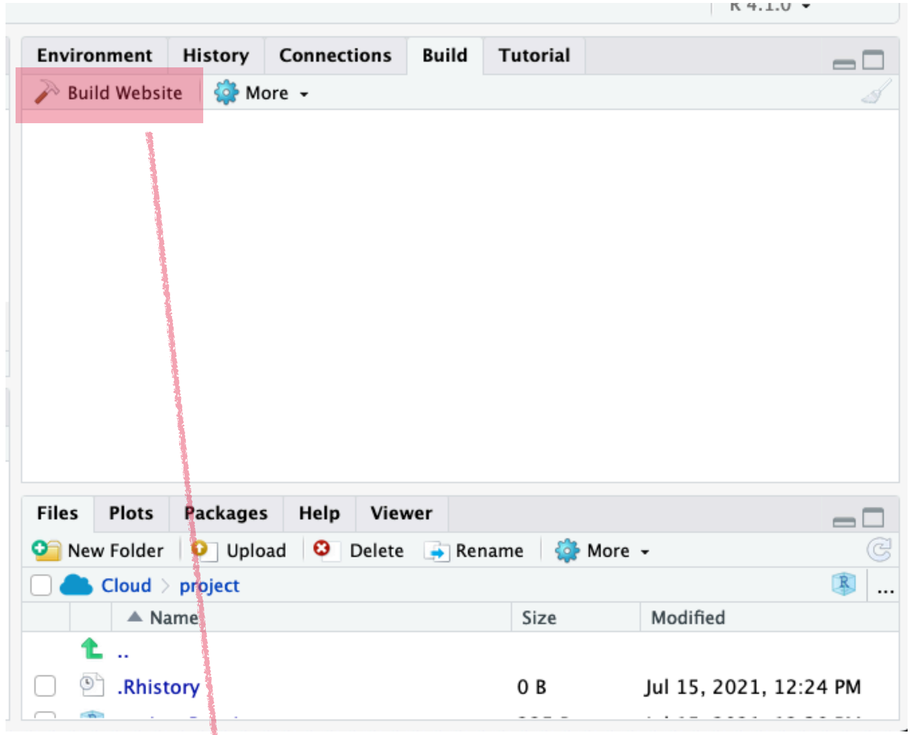
Next >

Essentials Menu

Adding publications

That's it! Now go to the Build Tab on the top right pane and click "Build Site"

You should now have an automagically formatted list of your outputs, complete with URLs if you've used DOIs (you can also use the url field in .bib entries).



Outputs

Cuskley, Christine. 2017. "A Sensory Theory of Protolanguage Emergence." *Protolang Conference*.

Cuskley, Christine, Rachael Bailes, and Joel Wallenberg. 2021. "Noise Resistance in Communication: Quantifying Uniformity and Optimality." *Cognition* 214: 104754. https://doi.org/10.1016/j.cognition.2021.104754.

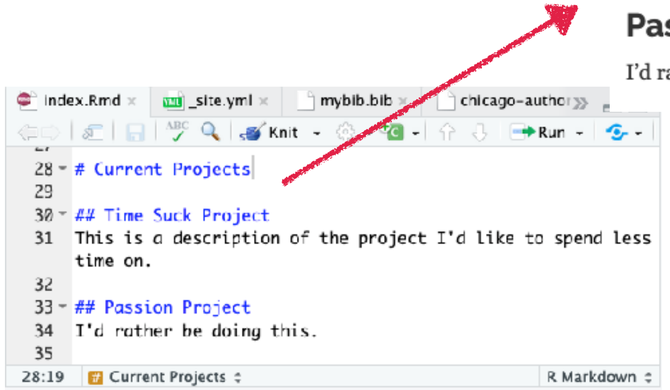
Cuskley, Christine, and Charalampos Saitis. 2020. "What Do People Know about the Senses? Understanding Perceptions of Variation in Sensory Experience." *PsyArXiv Preprint*. https://doi.org/10.31234/osf.io/ghcxv.

< Back

Essentials Menu

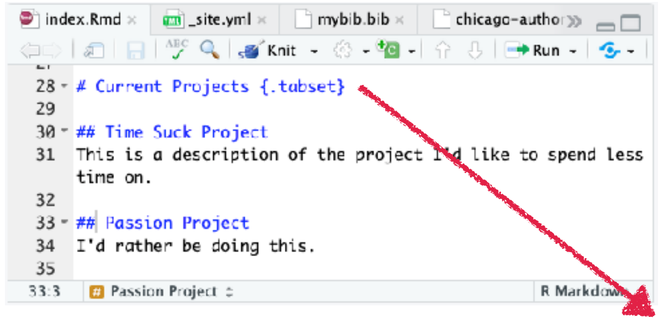
Adding Tabs

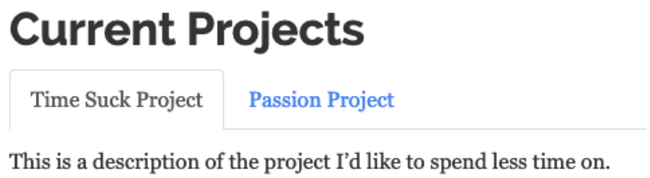
You might not want *everything* in your website to be linear. For example, maybe you want someone to be able to click through your various projects one at a time without leaving the page.



You can use tabs for this! Simply add `{.tabset}` next to the section that you want to appear tabbed. Build the website and see the difference.

Note that tabsets won't work unless you have adjacent levels, e.g., if you have tabset on a single hashtag title and then ### just below it, it won't appear tabbed.





This can be especially useful if you have lots of subsections, but still want the page to stay lean.

Essentials Menu

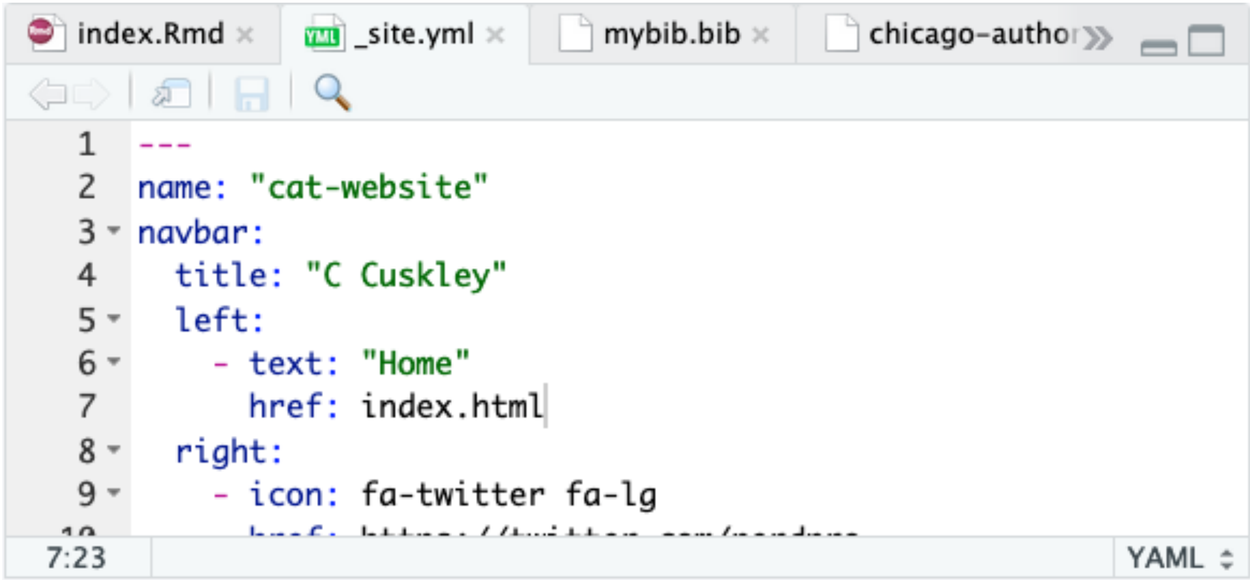
Adding to the navbar

The navbar we have is a little weird for a single page site; look at the _site.yml file.

Both the `title` and "Home" go back to index.html - this is the default behaviour for the title according to the theme, and we've specified this for the "Home" nav item.

This is a bit odd because *everything* is on index.html for our single page site - so if you're clicking these links, you're already there.

Instead, we can make our menu items jump to particular sections on the page.



Next >

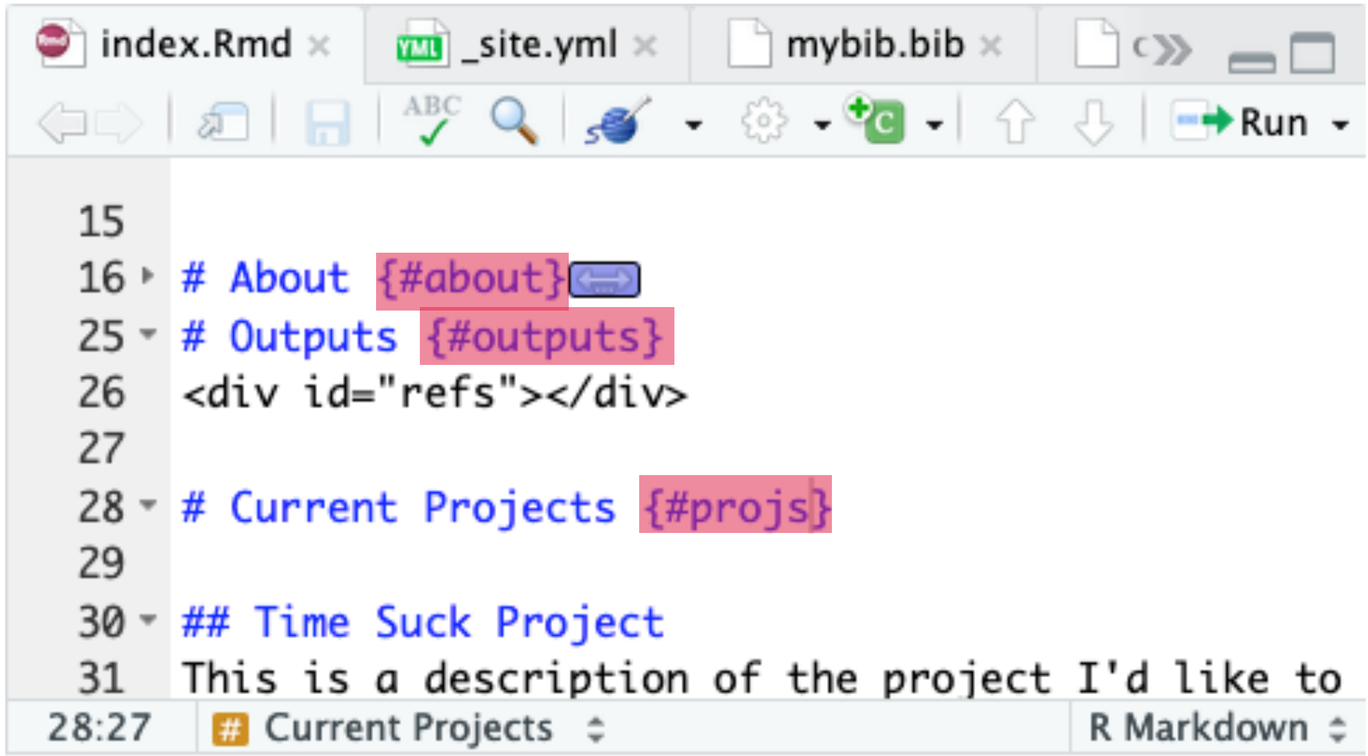
Essentials Menu

Adding to the navbar

First, we need to give each of our sections IDs in our index.rmd file.

Do this by adding a memorable and unique reference to each section you want to be able to jump to.

These can’t contain spaces and some other characters; best just to make them letters.



< Back

Next >

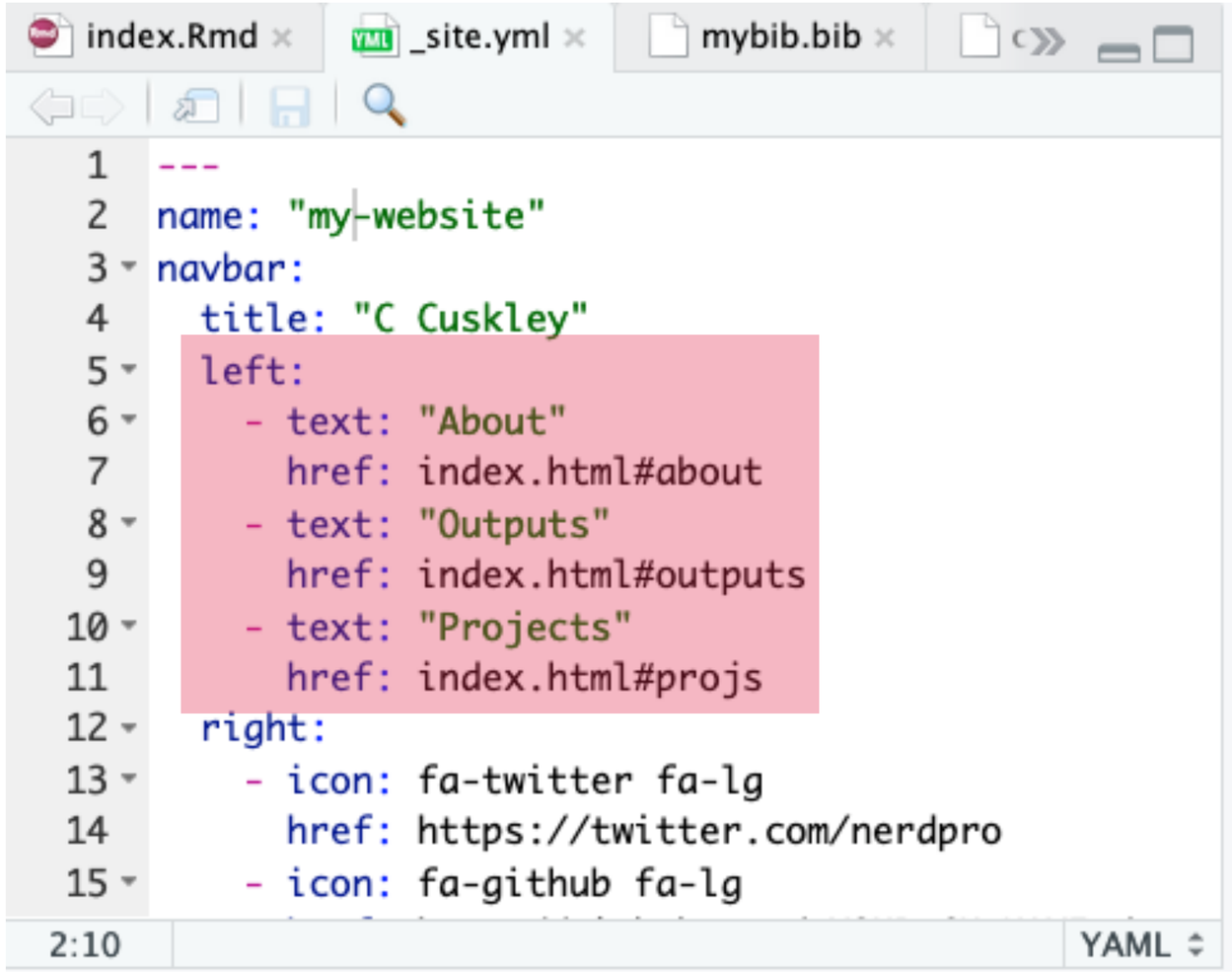
Essentials Menu

Adding to the navbar

Next, we need to go back to the .yml file and tell it to make menu items that point to these.

Add a text and href field for each section you want to point to. The text should be what you want to appear in the menu.

The href field just has to be `index.html#` the ID you assigned in the .rmd file.



< Back

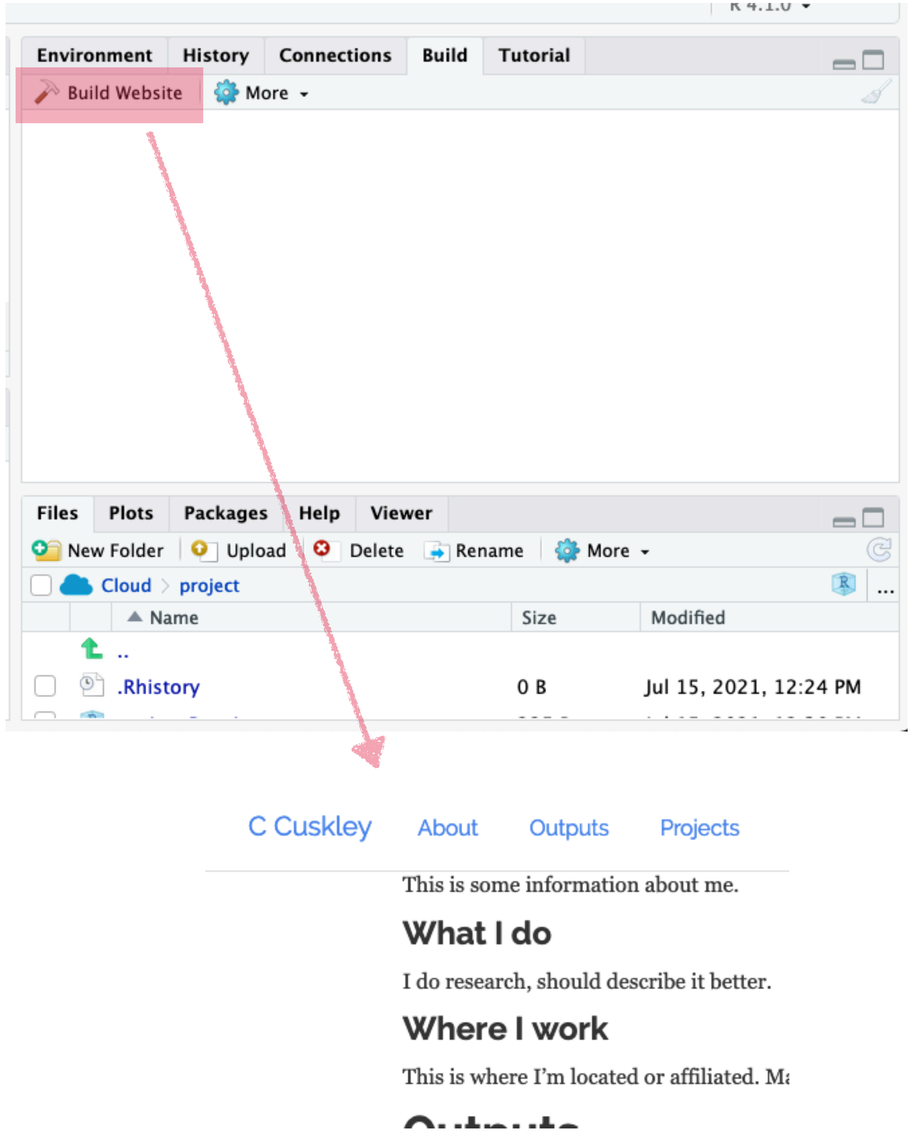
Next >

Essentials Menu

Adding to the navbar

Now build - your navbar should show the new entries, and clicking them should jump to that bit of the page.

Note, however, that unless you have enough content to go beyond the height of your browser window, it won't have anywhere to jump, so won't look like it's doing much.



< Back

Essentials Menu

Make a site with RMarkdown

You now have a beautiful website.

But you're the only one that can see it :/

We need to put it on the internet. We'll do this with Github.

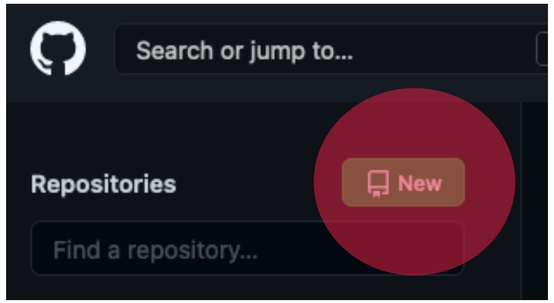
Create an account on github, if you don't have one already.

The URL for your website will eventually be [YourGithubUsername].github.io*

If your username is e.g., kewlHacker1985, you might want to consider opening another github account with a different email.

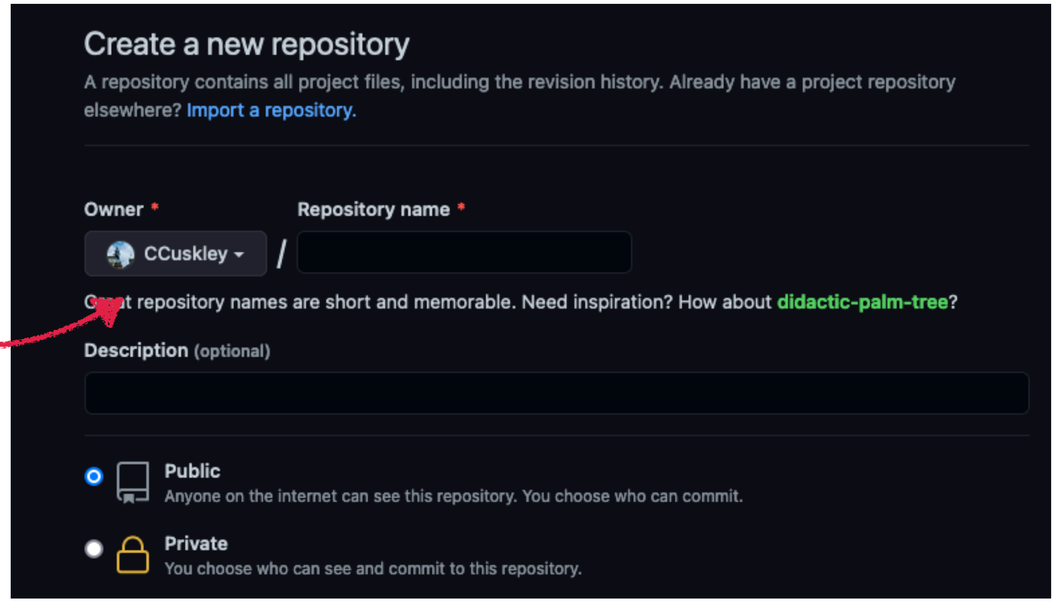
*Unless you want to get into a custom domain - this is a bit complicated, so we won't go over this here.

- **Create a repository named [YourGithubUsername].github.io**
- **e.g., my site is ccuskley.github.io, so this is also the name of the repository**
- **You can do this any way you like - using the command line, the desktop client, or in the browser.**
- **If you're new to git/hub, I recommend the browser - I've modeled this to the right.**

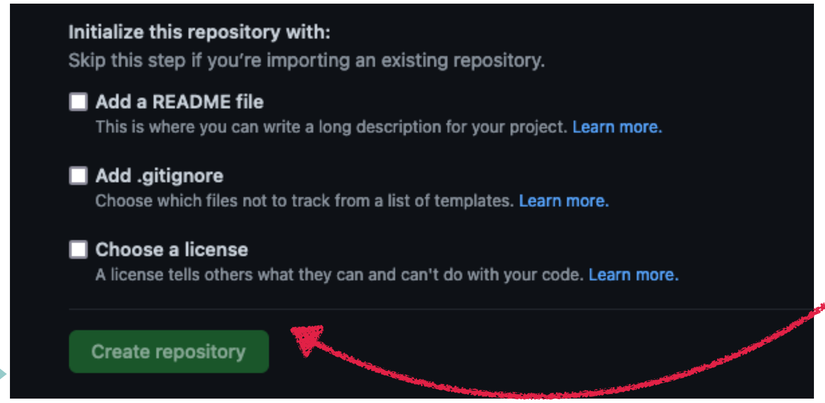


1. Make a new repository

2. Name it whatever this says .github.io



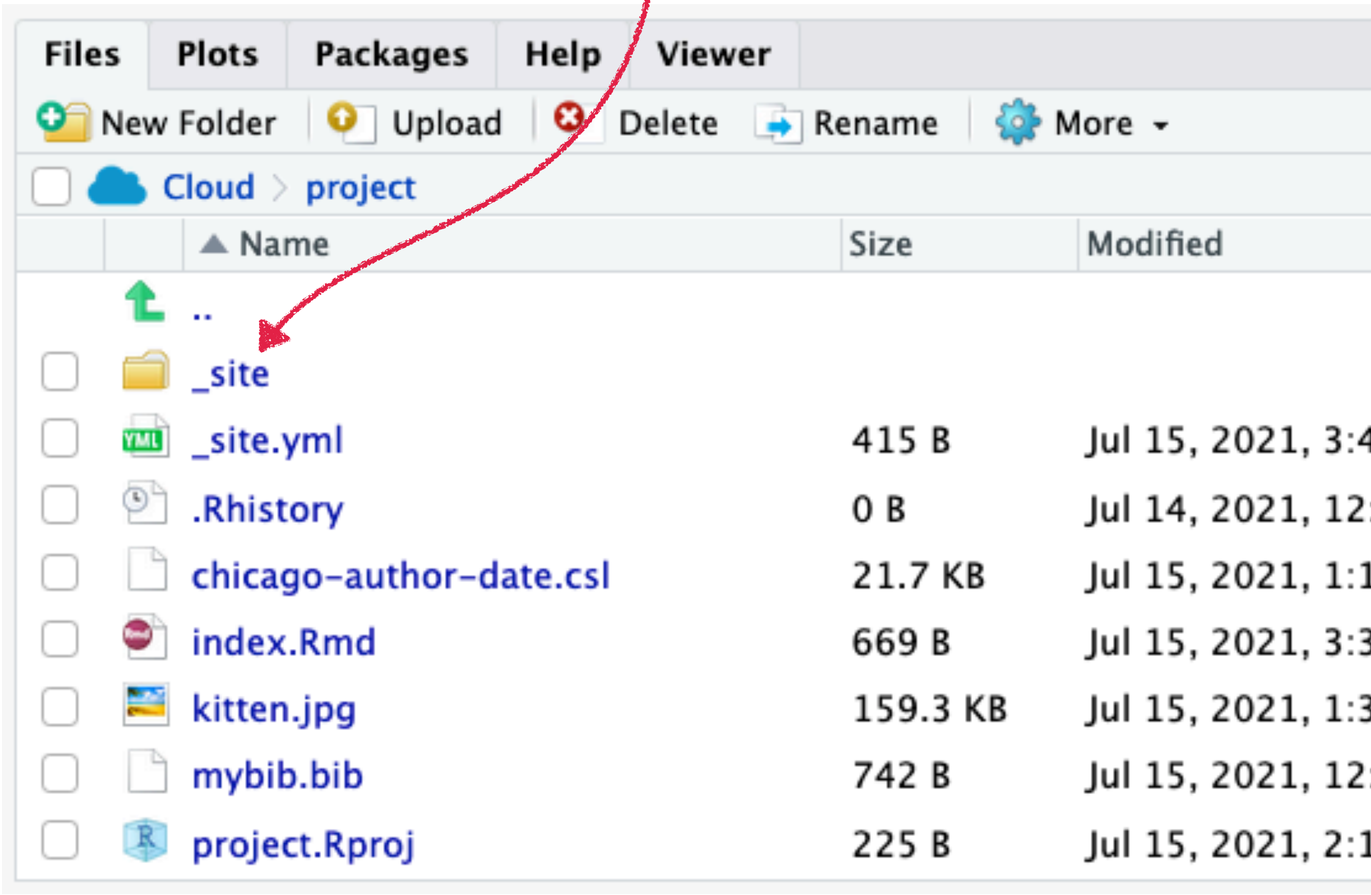
3. Make sure it's public



4. No need to add readme etc. Just Create.

- Now you want to upload your files to github.
- You only need the files generated by R when it builds the website, in other words, everything in the _site folder
- Upload the files in _site to the top level of your repository (or add/commit/push, if you git down)
- Make sure to add the files *in* the folder, and not the folder itself (although you *do* want the folder site_libs that's inside this)

You want to upload what's in here - **make sure you Build Website** in the top pane with your final files first!



Once your files upload, YourGithubUsername.github.io should automatically redirect to your index.html file

Spam your friends and family with your new website.

That’s it!

I hope it was helpful, do feel free to be in touch with questions!

Start again