WEB RESUME – R MARKDOWN

Created by: Megs Seeley

**R Markdown, CSS, and Git**

This tutorial requires no coding experience or familiarity with R, CSS, or Git. This tutorial is a general introduction to using R Markdown.

Note: The example resume is a reference to a podcast character.

**Section 0 – OPENING and KNITTING YOUR RMD FILE**

 To open a new R Markdown (RMD) file, click on the new script icon then select ‘R Markdown’

Next, we will see what happens when we knit our document. First, we will need to save our file as ‘index’ The reason will become clear later.

 Click the knit icon and choose ‘Knit to HTML’

An html of our document with stock code will appear in our viewer. If we want to view our output in a window pane instead of the R studio viewer, we can select that option in the gear icon menu.

Now let’s dive into editing your RMD!

**Section 1 – THE HEADER**

The first section of our document is the header. The stock header is fairly bland, so we will spruce it up.

First, we will change the title. Since this is your resume, just put your name here.

---

title: Usidore the Blue

output: html\_document

---

Now we probably want to center the title and change the font family.

We can center objects using center tags: <center> TEXT </center>

Awesome! Now let’s apply that to our title:

title: <center> Usidore the Blue </center>

Next, let’s edit the author section. In most RMDs, you would simply put your name here. Here, we will take advantage of this section to add your contact information.

We will likely want to style this subheader a bit. We can do that by inserting a ‘span’ tag.

< span style = ‘ ‘>

Within the quotations, we can use css to style the text. Let’s change the ‘color:’ ‘font-size’ and ‘font-style’

Since we will include an ‘@’ in our email, let’s ensure that we escape it using a ‘\’. Without the ‘\’, we would unintentionally introduce a command into our code.

Great! We are almost done. Let’s take a (line) break <br>

If we want to add a link with our website or linkedin, we can use an ‘a’ tag.

Inside the ‘a’ tag, we can include the url of our website with ‘href=’. If we want new tab to open when someone follows our link, we use ‘target = “blank”’.

Now check out the code below to see how we put it all together.

author: <center> <span style = 'color:dimgray;font-size:18px;font-style:normal;'>puppies.supplies\@gmail.com

| (312) 867-5309 <br> <a href = "https://www.usidorerocks.com">https://www.usidorerocks.com</a></span></center>

**Themes**

To stylize our RMDs without a lot of extra code, we can use the build-in themes. To see examples, visit: <http://www.datadreaming.org/post/r-markdown-theme-gallery/>

We can also choose add-on themes, see link above to style our document.

To include a theme in our document, we include this under output.

output:

rmarkdown::html\_document:

theme: yeti

Since we will customize our body font, lets revert the output to:

output:

html\_document: default

**Section 2 – STYLIZING YOUR DOC!**

Let’s talk a bit about that code snippet automatically generated in your RMD. You can use code chunks to insert code in your document. You can choose to have your code visible or invisible.

‘include = F’ will suppress the code itself and all its output.

‘echo=F’ will result in code outputs, like plots, appearing, but it will not print out the underlying code in your final document.

Let’s just delete the stock text and code for now.

**CSS**

In RMD, we can custom our document using css. Now, you do not need to know extensive css to style your documents. We will go over a couple stylization examples to give you an idea of what you can do with css.

To include a css chunk, we will use:

```{css, echo = F}

```

Now, we can customize our entire document within this css chunk. We will start with the body. Let’s play around a bit with some of our options just to see what the different commands do. It might be useful to add some nonsense text outside the code chunk to observe how the changes affect the document.

body {

font-family:DejaVu Sans;

font-size: 30px;

background: blue;

color: white;

}

That was fun! Now let’s get serious again. Remove the last two lines (background and color).

**Section 3 – THE TOFURKY AND POTATOES**

Now let’s add our content. We will revisit styling our document once we have something to look at.

Add your main sections: Education, Experience, Presentations, Awards, Qualifications …

Now, add <h2> tags surrounding education and observe what happens when you knit your document.

Note, in markdown language, we can also use hashtags.

# == <h1></h1>

## == <h2></h2>

<h2>Education</h2>

## Education

The text is much larger!

Now, go back to the css code chunk. We will customize both our ‘h1’ and ‘h2’ tags.

h1 {

color: darkblue;

}

h2 {

color: black;

text-align:center;

font-size: 30px;

}

Awesome! Now we can fill in our resume info!

**Section 4 – FINAL TOUCHES**

Now that we have our content, we can finish styling our resume.

*The Flair*

\*\*\* == horizontal line

\*italics\*

\*\*bold\*\*

<br> == line break (this can also be achieved by adding two spaces to the end of the previous line)

. \* make

. \* a

. \* bulleted list

. \* using

. \* asterisks

# Note: List needs an empty line above list and space between asterisk and text

Special characters can be achieved using HTML entities (<https://www.w3schools.com/html/html_entities.asp>)

*Columns*

Rather than allowing our bulleted lists to run down the entire page, we can create columns.

Returning to the css code chunk, we will add id classes for a left and right column.

#left {

}

#right {

}

We will adjust a couple properties to create the columns.

Float – positioning text

Width – width of our text container (imagine we are creating text boxes using our code)

#left {

float:left

width:48%

}

#right {

float:right

width:48%

}

Now that we created our ‘text boxes’ we can apply them in our code using ‘div’ tags.

<div id = ‘left’>TEXT</div>

<div id = ‘right>TEXT</div>

Because this can jumble our text a bit, we can add a couple break tags to separate our columns from the next line.

<br>

<br>

*Fancy Headers*

The last styling option we will cover is creating professional headers for your title. We will again return to the css code chunk.

Let’s first define two headers:

div.head1 {

}

div.head2 {

}

Here we will again create ‘text boxes.’ We can define our text box using:

padding – how much space we have surrounding our text

border-radius – roundness of our edges

background-color – color fill of the ‘text box’

div.head1 {

background-color:#4682B4;

border-radius: 3px;

padding: 3px;

}

div.head2 {

background-color:#B0C4DE;

border-radius: 3px;

padding: 5px;

}

Now we can apply these headers to our main titles and sub titles:

<div class = 'head1'>

<h3> Quests</h3>

</div>

<div class = 'head2'>

Defeat the Dark Lord | <span style = 'color:dimgray'>Hogsface, Foon</span>

</div>

**Section 5 – PUBLISHING**

Once you are satisfied with your resume, you can publish it on git as a webpage. In this section, we will learn not only how to push code to a github repository, but how to push it as a webpage.

\*\*Important note: Any code you publish on github either needs to go through a software release or needs to be created on your own time. This includes your RMD files. This tutorial hopefully provides you with the tools necessary to create a web resume, but please do not use DEVELOP time to code and publish your resume.

\*\* Unless you are viewing this tutorial on your own time and have your final resume (again, created on your own time), please just sit back and watch/read the rest of the tutorial.

*Git*

What is git? Git is simply version control for code. It’s like track changes on word. When we initialize git in a directory, we create a .git repository that tracks changes we commit through the cmd.

*Github*

Github is a git in a cloud. It allows us to share code, track changes, and access other peoples’ code. Github uses git as a version control, so every time you or another user updates a file in a github repository, it tracks those changes. There are other platforms that provide the same service, but Github is the most widely-used platform.

You will need to install git on your computer.

Follow this documentation: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

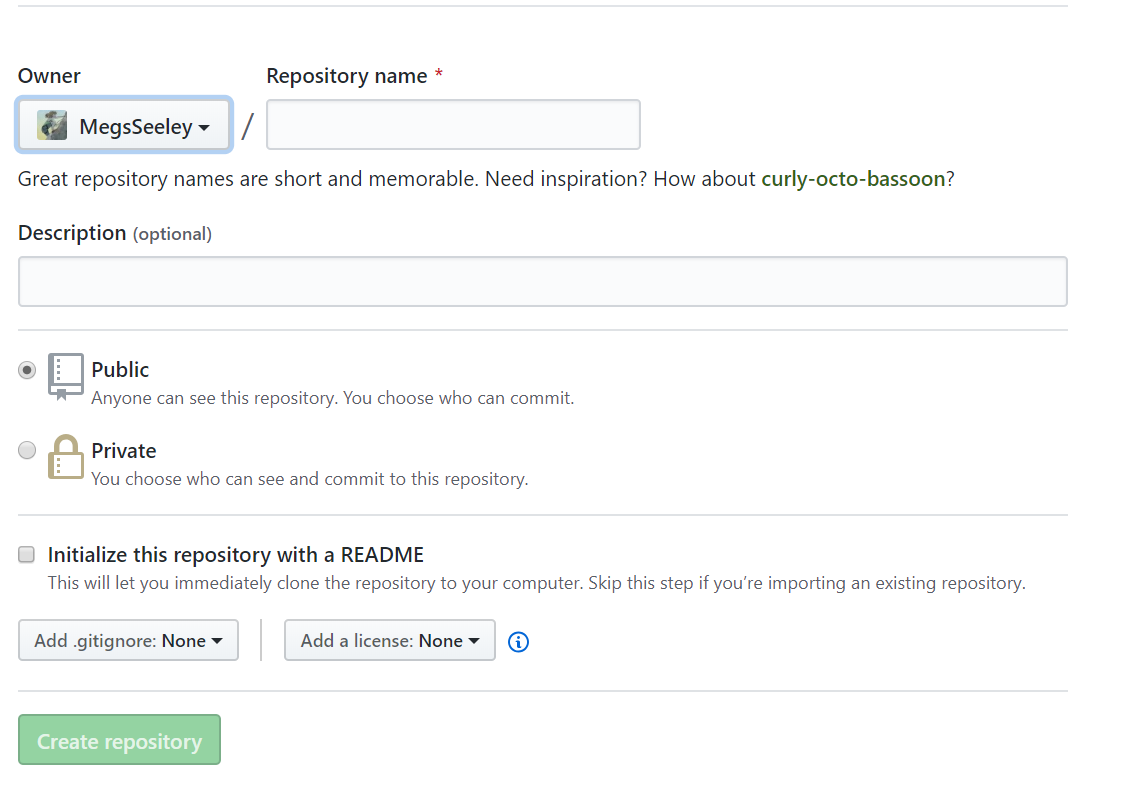
Github allows every user to create one published website with the url: <https://username.github.io>

Github recognizes the html file named ‘index’ as the main webpage. Because we named our RMD ‘index.rmd’ the html output will be ‘index.html’ If this is not the case, rename your html file.

Next, we will create a repository on github. If you do not have a github account, create one at: <https://github.com/>

Navigate to your repositories: https://github.com/username?tab=repositories

 Create a new repository here.



Enter the repository name: username.github.io

And create your repository

Next, we will open our command prompt or git bash and navigate to our folder

cd FILEPATH

*Connect to your account*

If you do not have git set up on your computer, connect to your git account

git config --global user.name "username"

git config --global user.email "email"

Check your settings

git config --list

*Make current directory a repository*

git init

Now you have a .git sub-directory that tracks all your commits. Check that it is present using:

ls -a

*Pulling*

We can pull code from github to our local files. To set up our website, we need to clone our web repository to our local file.

git clone <https://github.com/username/username.github.io>

Since this action will add an index.html file to our folder, we may want to delete the new index.html and reknit our RMD.

*Committing*

Pushing our code uploads it to github. First, we will tell git to track our files.

git add .

Next, we commit our files with a comment. Keep these comments concise, but include what you updates (e.g. “added DEVELOP spring term”). When you upload your code, everyone will be able to see your commit comments

git commit -m “Initial commit”

Check the status of your git

git status

*Pushing*

Now we will push your local files to github.

First, add the url of the repository to which you want to commit

git remote add origin <https://github.com/username/username.github.io>

In the future, you do not have to add the url. You can check that your code will be going to the desired repository with:

git remote -v

Finally, push your code

git push origin master

*Check out your webpage!*

<https://username.github.io>