Module 3 – lesson 01: Customizing HTML documents

Script

In Module 3 we’re going to focus on changing the parameters available for each document format to customize the style, appearance and layout of our documents. For this lesson, we’re going to explore the options available for customizing HTML documents. We will be referring throughout this lesson to the information contained in the Rmarkdown website for the html\_document format <http://rmarkdown.rstudio.com/html_document_format.html>

As we begin this module, let’s create a new Github repository for this Module 3. Log into your Github account and create a new repository called “Module 3”, type in a description of your choice and initialize the repository with a readme file. Click Create Repository.

Open RStudio. Click File/New Project, version control, GIT/SVN. Go back to Github and copy the Module3 repository URL to your clipboard and paste it into the RStudio GIT window to create your new project. Make sure you are creating this new project in your course folder “C:/RepTemplates” – when you finish you should have this folder created on your local computer.

C:\RepTemplates\Module3

Now that we have the project created and we’re connected to Github via GIT, let’s create a new Rmarkdown document. In RStudio, click File/New File/R Markdown. Choose Document – HTML document. Put in a title and click OK

Go ahead and save the document.

Let’s take a look first at customizing the Appearance and Style of the HTML document. Let’s look at some of the options available <http://rmarkdown.rstudio.com/html_document_format.html#appearance_and_style>

For HTML documents, there are a variety of THEMES available based on the bootswatch theme library – you can learn more at their website <https://bootswatch.com/> Not every theme is available for use with R markdown, but many of them are. For example, suppose we want our document to look like the Cerulean theme, <https://bootswatch.com/cerulean/>

To add a theme to out HTML document we need to edit the YAML header and add some options to the html\_document output keywords. Let’s edit our YAML header to change the theme. It should be noted that for each html\_document option we change, we put each option on a separate line in the YAML header with the keyword followed by a colon and the option setting. We’re going to set the theme to cerulean and we’re also going to change the highlight option for highlighting code syntax – for highlight, let’s choose kate. Each nested option is indented 2 spaces.

output:

html\_document:

theme: cerulean

highlight: kate

Go ahead and KNIT to HTML to see the result. You’ll notice that the headers are all in a blue font. Let’s try another one – change the theme to yeti and the highlight to espresso – KNIT to HTML and view the changes.

Another way to make changes to the appearance and style of your HTML document is to use Cascading Style Sheets or CSS. Let’s create a simple CSS file and then we’ll use it to apply formatting changes to the HTML document. In RStudio, click File/New File/Text File – type in the following CSS definitions

#nextsteps{

color: blue

}

.emphasized{

font-size: 1.2em

}

Save this file as “mycss.css” – be sure to include the CSS after the period in the filename so the file will be recognized as a CSS file.

Go back to your HTML document and change the YAML header options for the HTML document – remove the theme and highlight and add the option for CSS and then put in the newly created CSS filename.

output:

html\_document:

css: mycss.css

However, now we need to apply the CSS formatting where we want it. Let’s apply this formatting to the first section in the document defined by the level 2 header ##. At the end of this header title we will add the CSS formatting as follows

## R Markdown {#nextsteps .emphasized}

Click KNIT to HTML and see the changes. You’ll notice that not only is the header changed to a blue font but this whole section of text is now blue and is in a larger font as defined by the font-size: 1.2em specified in the mycss.css file. Your formatting options are pretty much limitless when applying CSS to an HTML document. Learning CSS is beyond the scope of this course, but there are hundred of helpful websites for learning CSS. A good place to start is <https://www.w3schools.com/css/>

Let’s try some other customization options. In module 2 we saw how to change the figure width and height settings to affect the size of a figure created within an R code chunk. You can also affect the sizes of your figures for the whole HTML document by setting these options in your YAML header. Let’s edit the YAML header and add options for fig\_width and fig\_height.

output:

html\_document:

fig\_width: 5

fig\_height: 5

Before we KNIT the document, let’s add another plot to our document – change the 1st code chunk from

summary(cars)

to

plot(cars)

KNIT to HTML and notice that the plots are a little smaller than they were before. Previously, the default figure width is 7 and the height is 5. These heights and widths are in inches.

Another way to access the rendering options for your document is to click on the little gear shaped icon next to the KNIT button and select Output options. Try selecting different options in here and watch how your YAML header is changed automatically. Let’s click on Figures and change the figure width and height to 6 and 6 – click OK. Notice that the YAML header now sets the figure height and width to 6 each.

We can also add a table of content to the HTML document. Before we test adding a table of contents, let’s add a few more sections to the HTML document. Let’s add the following 2 section – one with a list of bullets and one with a simple equation.

## A short list

\* apples

\* bananas

\* oranges

## A simple equation

$$ Y = \beta\_0 + \beta\_1\*X $$

We now have 4 level 2 headers so when we create a table of contents we should get a contents with 4 sections. We can add the table of contents either by editing the YAML header directly using the toc keyword – OR we can click the gear icon for Output Options and in the General Tab, click the 1st box for Include Table of Contents. Click OK – see the changes in the YAML header.

Click Save. KNIT to HTML to see the result. You notice that the top of the document now has 4 bullets – where each one is a hyperlink to that section of your document. In HTML the table of contents becomes a navigation tool.

Another neat table of contents to try out is a floating table of contents. Let’s edit the YAML header and add the toc\_float option and set it equal to true. Let’s also open the HTML file in a separate browser window to see the floating table of contents better. You’ll notice in the browser window that the table of contents is shown on the left and as you click each row in the table of contents, the document scrolls to that section of the document.

The table of contents will by default track and list level 1, level 2 and level 3 headers. Let’s go back and add a couple level 1 headers and level 3 headers. Save and KNIT to HTML to see the result.

# Module 3 – HTML document

### Plot of the Cars Dataset

### Plot of the Pressure Dataset

Let’s view the HTML document in a browser window to see the resulting table of contents. You’ll notice that lower level headers are nested within higher level headers until you click on them and then the lower level headers are shown.

Let’s look at one more option we can customize for HTML documents. In a previous lesson you saw how setting the code chunk option echo=TRUE or echo=FALSE could be used to show or hide the code in the final document. There is one more option called code folding. Let’s add that option to the YAML header – just append this option to the other options we’ve already set. Set the code\_folding option to hide, which will initially hide the code unless the reader clicks on the resulting button to show the code.

output:

html\_document:

code\_folding: hide

Save and KNIT to HTML to see the result. You’ll notice the following three things:

First, there is a button shown at the top of the document that says Code with a down arrow next to it – if you select this you can choose to Show all code or Hide all code. Toggle this back and forth and watch the changes.

Second, the only code that was shown or hidden was for the 1st code chunk for the plot of the cars dataset. This is because the ECHO option for this code chunk was set to true. The ECHO option for the plot(pressure) code chunk was set to false – so code chunks that have ECHO set to false will never be shown even when the code\_folding option is used. If you want to use code\_folding, I would suggest setting ECHO to true for all of your code chunks and then let each user show or hide the code as they wish when they read your final HTML document.

Third, for the 1st plot of the cars dataset, you can click on the Code button to show or hide this code chunk only.

Let’s add one more option to the YAML header to keep the markdown document. Technically, in RStudio you are creating an “R” markdown document. When the document is compiled, each file is first converted to a basic markdown document which Pandoc then compiles into your desired format – in this case an HTML document. You can choose to KEEP this interim plain markdown document. To do this, we need to set the option keep\_md to true.

output:

html\_document:

keep\_md: true

Save the file. KNIT to HTML and look at your list of files – notice that there is now a file with the same name as your HTML R markdown document but with the extension xxx.md. It turns out Github was built to natively display plain markdown files. I’ll demonstrate this after we upload everything to Github next.

Let’s make one more minor change. Save your current HTML document to another filename – githubmd\_document.Rmd. This time we’ll edit the YAML header and change to output to be a md\_document. We’ll set the markdown variant to be compatible with github.

output:

md\_document:

variant: markdown\_github

Save the file and KNIT to MD. This opens a viewer window but doesn’t display it the way it will look in Github, so close the viewer window. We’ll view the final github markdown file when we upload it to github in a minute.

Before we back everything up to Github, let’s create a /docs folder in your Module3 project folder. We’ll put a copy of your HTML document in this directory. That way when we push everything to Github, we can utilize the Github pages settings in the Github repository to “serve” our HTML document as a webpage. So, open your file explorer and add a new folder called docs to your Module3 project directory.

C:\RepTemplates\Module3\docs

Make a copy of your HTML document and put a copy in this new directory. Let’s also make a 2nd copy of this HTML file and rename it “index.html”. I’ll explain why we did this in a moment.

Now let’s go ahead and back everything up to your Github account.

Open Git Bash and make sure you are in the correct directory:

C:\RepTemplates\Module3

Once in that directory, type in the following 4 Git commands to check the status of your local files compared to your Github cloud repository; add or stage the modified files; commit your changes; and then push the changes to your Github cloud repository.

git status

git add .

git commit –m “add html document and files for module 3”

git push

Now go to your Github repository, refresh if necessary and see your newly committed files. Next go to the settings for the Module3 repository. Scroll down to the settings for the Github pages. Click on the button for Source and change it from NONE to master branch/docs folder. Click Save. The page will refresh. Scroll back down to Github page and you’ll see a URL provided for your website for your repository. It should look something like this:

<https://USERNAME.github.io/Module3/>

<https://melindahiggins2001.github.io/Module3/>

By default, this URL actually links to a file called “index.html” in this directory. You can link to other HTML documents directly by adding the filename on the end of the URL like this:

<https://melindahiggins2001.github.io/Module3/html_document1.html>

Congratulations – you’ve just made your first website with your HTML document.

You might try viewing this new website on a mobile device like a cell phone to get an idea of how these HTML documents display on a mobile device. The floating table of contents is especially useful on mobile devices.