Module 2 – lesson 03: Document Elements

Let's continue working with the same R markdown document “module2\_rmd1.Rmd”. Log into your Github account, open RStudio, and open the RStudio project for “Module2\_rmd1”.

So far you’ve explored the YAML header and tried out several R markdown syntax markings to change the formatting of text in your document. Next we’re going to insert other objects and elements within your document, such as:

* + - Figures;
    - Tables
    - And Equations

Figures

Take a look at the bottom of your current R markdown file. The last section of this current document contains a chunk of R code that makes a plot of the pressure dataset which is built into the base R software.

Let’s take a quick look at this built-in dataset. Go to the Console windows (bottom left) and type in

data1 <- pressure

this creates an object called “data1” that is assigned a copy of the built-in pressure dataset. Then click on the Environment TAB in the upper right window – click on the little table icon on the right – this opens the dataset in a viewer window at the top left. There are 2 columns of data and 19 rows for 19 data points of temperature and pressure. You can get more detailed information on this built-in dataset by looking it up in the HELP window. This brings up a help page on the pressure dataset that says that the pressure dataset is “Data on the relation between temperature in degrees Celsius and vapor pressure of mercury in millimeters (of mercury).”

So, in the R chunk shown here, there is only 1 line of code between backticks ```{r} ending with 3 more backticks ```

This line of code makes a scatterplot of the pressure (along the Y vertical axis) against temperature (along the X horizontal axis).

plot(pressure)

By running this R code chunk, a scatterplot is created and then inserted in the document where this code was specified.

By the way, we can control the size of the figure using code “chunk options”. You can learn more about code chunk options at Yihui Xie’s website for the knitr package (<https://yihui.name/knitr/options/> )

Let’s change the figure width and height using the fig.width and fig.height chunk options. These options come after the r in the {r} curly brackets. The current R code chunk has the following

```{r pressure, echo=FALSE}

Let’s explore each piece.

The curly brackets indicate that this is a code chunk and the r indicates that the code will be R code.

The word pressure is a label of the code chunk. This is helpful for debugging later. When you get an error the R Markdown log will show in which code chunk the error occurred. Giving them descriptive names will help you keep track.

Let’s look at the R Markdown TAB at the bottom left and review the log of when we last rendered or compiled the document – when we last ran KNIT.

After the chunk label and a comma, there is already 1 chunk option saying echo=FALSE. This tells R markdown to hide this code chunk and not “echo” it or show it in the final document. So, if you look back at the documents you’ve made so far – all you see is the plot of the pressure dataset – you do not see the R code.

At the very end of the document, let’s add another code chunk and alter this next figure slightly. Click the button in the editor window with a little green C with a + plus, click the down arrow to insert an R code chunk. When you click this, it automatically enters the 3 backticks with the r in curly brackets ```{r} followed by a blank line and then 3 more backticks ``` ending the code chunk.

Go to the blank line and type in

plot(pressure)

Now go back to the 1st line. We’ll add a new label “pressure2” since each code chunk must have a unique chunk name. We’ll also add fig.width=5 and fig.height=5 which does 2 things – it will make the horizontal and vertical dimensions of the figure the same and should render a figure approximately 5 inches by 5 inches – see the chunk options described on Yihui Xie’s website.

```{r pressure2, fig.width=5, fig.height=5}

Save the RMD file and KNIT to HTML and note the differences between the 2 plots. Not only will the plot sizes be different, in the 2nd one we left out the echo=FALSE so the R code was shown right before the 2nd plot. If you want to hide this code, go back and add echo=FALSE to the chunk options.

```{r pressure2, fig.width=5, fig.height=5, echo=FALSE}

In RStudio 1.1.x, you’ll also notice that in the editor window, within the code chunk at the far right there are a couple of icons. Click on the one that looks like a gear and another window pops up that shows various code chunk options available including

* The name of the code chunk
* What kind of output you want to show (with or without the code – setting echo-TRUE or FALSE)
* You can also turn on or off warnings and messages
* At the bottom you can also set the figure width and height – try changing this to 4 inches and watch the changes in your R markdown file.

There is also a green arrow > that you can click which will “run” the R code as if you were running R from the command line or from a standalone R script. If you click this, the plot(pressure) command will be executed and the plot will be generated in the Plots window.

Tables

Doing tables in with R markdown using Pandoc’s markdown can be done but is a tedious process for even simple tables. There are numerous examples provided on this website. ([http://rmarkdown.rstudio.com/authoring\_pandoc\_markdown.html#tables](http://rmarkdown.rstudio.com/authoring_pandoc_markdown.html) )

Typically, it’s easier to use R code to generate a table. The best function for making tables using R markdown is the kable function from the knitr package. (<https://yihui.name/knitr/>)

It may also help to install and learn more about the printr package also which improves the formatting of knitr output. (<https://yihui.name/printr/> )

Let’s try a simple table of another built-in dataset, cars. Look at the help pages for cars. The cars dataset has 50 observations or rows and 2 columns. The 1st column is data on the speed of the cars and the 2nd is the stopping distance data. We can use the head function from base R to look at the top 6 rows of the cars dataset. The cars dataset is in an R object called a “data.frame” which the kable function handles like a table. So the following R chunk will make a table of the top 6 rows of the cars dataset.

Let’s create another new header in our document

## Insert Tables

```{r}

knitr::kable(head(cars))

```

Since the kable function comes from the knitr R package, it’s good practice to list the package followed by 2 colons followed by the function so it is easy for you or anyone else reading your R code to know which package the function came from. There are tens of thousands of R packages and some have the same function name but do entirely different things. So, to avoid confusion it is always a good idea to list use the package::function() when using a function in R.

So this code says to extract the “head” or top 6 rows of the cars dataset and use the kable function from the knitr package to print out a table in the final document.

Go ahead and save your document and click KNIT to HTML to see the result. Feel free to also try KNIT to WORD and KNIT to PDF to see how the table appears in those formats.

We can add other options to the kable function – like adding a table caption. Let’s add the option caption = "Top 6 Rows of Cars Dataset"

knitr::kable(head(cars),

caption = "Top 6 Rows of Cars Dataset")

You’ll notice this added a caption or title at the top of the table in each format.

Equations

For those of you not interested in typing math equations in your documents, you can skip this section. But if you are interested, you can embed equations using LaTeX syntax. For example, suppose we wanted to write out the equation for a simple linear regression model. We would embed the LaTeX formatting inside a block beginning and ending with 2 dollar signs $$

## Insert an equation

$$ Y = \beta\_0 + \beta\_1x $$

There are hundreds of websites with information, tutorials and help online for formatting equations using LaTeX. (<https://www.sharelatex.com/learn/Mathematical_expressions>)

Finally, let’s make sure to back everything up and save your changes to your Github repository.

Open Git Bash and change to the directory for your Github repository created for “Module2\_rmd1” – so go to:

C:\RepTemplates\Module2\_rmd1

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 “inserting multiple elements to my RMD file”

git push