Module 3 – lesson 07: Adding and Using parameters in a Document Template

Script

In this lesson we are going to add a parameter to the steakArticle document template to create new articles on the fly simply by changing the region of the United States contained in the steak\_survey from the fivethirtyeight R package.

Go ahead and log back into your Github account and open your Module3 repository. If you need to, start RStudio and open the Module3 project and open your “steakArticle.Rmd” R markdown document. Let’s save this file under a new filename. Click File/SaveAs “steakArticleParams.Rmd”

To add a parameter to an R markdown document, the first thing we need to do is add the params keyword to the YAML header. You can learn more about R markdown parameters for customizing documents at <http://rmarkdown.rstudio.com/developer_parameterized_reports.html>

Add the following option to your YAML header on the last two rows of the YAML header

params:

region: "Mountain"

We’re next going to use this params keyword parameter to replace everywhere in the document we’ve previously mentioned the Mountain region of the United States. Not only are we going to change the text “Mountain” throughout the document – but we’re also going to replace the associated R code for the variable “region”. That way when we change the region parameter in the YAML header - the data, the table summaries and the clustered bar chart will all update automatically.

The first place we’re going to replace the text “Mountain” throughout the document to a short section of inline R code that puts in the region variable now defined within the params object. When the YAML header is read with the params keyword when the document is compiled, an R object called “params” is created in the environment which contains the variable or variables defined here in the YAML header. In this case we’ll have a params object with 1 variable defined for region and we’re going to set the value for that variable to “Mountain”. To get access to this value, we type in the following R code

params$region

which can be read as get params object and select (indicated by the dollar sign) the region variable.

So, we’ll type this as inline R code as

`r params$region`.

Let’s go ahead and use this inline code and make an initial change the YAML subtitle to

subtitle: "Summary Report For the `r params$region` Region"

Save the document and KNIT to HTML to make sure that the code works. You’ll notice that the subtitle is the same as before, so now the word “Mountain” is still shown but technically it was inserted based on the inline R code. You’ll also notice that since we deleted the R markdown syntax of the underscores \_ and 2 asterisk at the beginning and end of the word, the bold, italics formatting has now been removed. I used the bold, italics formatting as a formatting trick so you could do a quick look across the document and see where to insert the inline R code instead of the text for the region.

Let’s go ahead and make the other changes throughout the document to replace the work Mountain by this section of inline R code to use the params$region variable. In the RStudio editor you do have the option to do a “find and replace” by going to Edit/Replace and Fine – or in the editor window you can click on the little magnifying glass icon – both approaches open a little tiny window at the top of the editor window to do the find and replace function with various options.

However, in this document you only have a few places where the work Mountain appears – so I’m going to go through the document and change these manually – that way you can see how to change each instance. You may also refer to the read ahead supplementary materials to see all of the updates to the document. Here are all of the changes we need to make: [CHANGES SHOWN ON COMPUTER]

In the first R code chunk to define the data subset sdat;

One place each in the 2 paragraphs in the Purpose section;

In the sentence above the clustered bar chart plot;

And finally in the R code for the caption of the bar chart. It is worth noting that when we change the R code we do NOT need the backticks and `r` since this already is R code. Here we only need params$region to replace the text “Mountain” [EXPLAIN IN COMPUTER DEMO HERE]

Save the document and KNIT to HTML to make sure all the changes worked. Granted this doesn’t seem that exciting since visually nothing has really changes except that the word Mountain no longer appears in bold, italics format. That OK.

So, let’s get to the FUN part!! Since we’ve replaced a manual section of text – the word “Mountain” – with an R object params containing the region variable - All we need to do to write a completely new report for a new region of the United States from this document template based on the steak survey dataset is simply change the region in the params section of the YAML keyword. Let’s go to the YAML header and change “Mountain” to “Pacific” in the YAML header.

params:

region: "Pacific"

Save the document and KNIT to HTML and you should now see an updated document with updated text for Pacific instead of Mountain along with updated tables and an updated clustered bar chart.

AWESOME – you’ve now officially created your first honest to goodness document template where the region can be dynamically updated to customize content on the fly as desired!!

Besides manually changing the YAML header, you can also access parameters in the KNIT button – click on the down arrow next to the KNIT but instead of selecting the first option for KNIT to HTML, select KNIT with Parameters. When you choose this option, a new window opens where you can type in the region parameter instead of changing the YAML header directly. Try this and type in New England (without quotes with a space in the middle and with New and England both beginning with capital letters) for the region and click the KNIT button at the bottom right. When we type in New England it has to match exactly with what is in the steak\_survey dataset or you will get an error.

This is helpful, but can be error prone since you have to know the regions you want and you have to type in the region exactly as it was typed in the steak survey dataset. It would be better if we could create some kind of pick list or pull down list so the person updating the document template won’t make simple errors. We can do this by updating the YAML header one more time. Before we make these changes – go ahead and save your current document template “steakArticleParams.Rmd”. And let’s Save As under a new filename “steakArticleParamsList.Rmd” which will be another version of the document with a pull down list for choosing regions in the KNIT with parameters dialog box.

It turns out when using the params keyword instead of entering a single value for region we can enter the whole list of choices. Of course, you have to know what the different choices are in the dataset you are using in the document. To see the various regions available in the steak\_survey dataset, go to the command line in the Console window at the bottom left and type in the following R code

unique(steak\_survey$region)

The unique function lists all of the possible values of the region variable in the steak survey dataset. You’ll notice that we get a list of 10 regions, one of which is NA missing. For our purposes we will not include NA as an possible region choice since it is missing. Technically, you can create a report summarizing respondents who didn’t answer the region question, but we will skip that option for this exercise.

Now that we have the list of all possible 9 regions provided in the steak\_survey dataset, let’s update our YAML header. Go ahead and look up the new YAML header in your read ahead supplementary materials. This YAML header sets up how the KNIT with Parameters interface works. These YAML settings create a pull down list where the person using the document template can choose the region they want to KNIT the document for without having to be sure they are typing the region in exactly correct.

Learn more at <http://rmarkdown.rstudio.com/developer_parameterized_reports.html#parameter_user_interfaces>

These YAML settings leverage the SHINY package for R to build the user interface dialog window with the pull down list. While learning SHINY is beyond the scope of this course, it is really nice that you have access to SHINY functionality when using the KNIT with Parameters option.

Go ahead and type in the updated YAML header.

params:

region:

label: "Region:"

value: Mountain

input: select

choices: ["East North Central",

"East South Central", "Middle Atlantic",

"Mountain", "New England", "Pacific",

"South Atlantic", "West North Central",

"West South Central"]

Save the document and KNIT with Parameters. Choose another region and see if the report works like you expect.

In addition to using the RStudio R Markdown window interface for compiling a document using KNIT to HTML or KNIT with Parameters, you can also write R code at the command line in the Console or from inside an R script using the rmarkdown render function. Here is the R code for compiling the steakArticleParams.Rmd for the Mountain region – make sure you are using the document version that has the simple params definition for a single region at a time – NOT the full params pull down list.

Let’s create a new R script to save this R code. Click File/New File/R Script

rmarkdown::render("steakArticleParams.Rmd",

params = list(region = "Mountain"))

save the R script as “steakRender.R”

In the R script window, highlight the 2 lines of R code and click Run. You’ll notice in the bottom left window for the Console the processing steps of the R markdown document scroll by. If everything works correctly, the last line will state that the output created is “steakArticleParams.html” – check your file list to see the new HTML document created. However, a preview window is NOT shown. You have to manually open the file in a browser application yourself to see the results.

We could use this R code and keep typing in different regions to generate new articles for each region. However, what if we want to make 9 articles all in one run so we have one for each of the 9 regions? We can use this basic R code to do that with a few changes.

The next section of R code will utilize the purrr R package for processing a list of values in a variable for a given function. To learn more about the purrr package see <http://purrr.tidyverse.org/> Purrr should be installed as part of tidyverse, but if you need to you can install purrr using the Tools/Install Packages in RStudio.

The first step in setting up the functionality of purrr for batch processing is to define a function. We will create a new function called render\_report. See the read ahead supplementary materials to get the code. In your steakRender.R script go ahead and type in the R code for this new function

render\_report <- function(regionvar){

template <- "steakArticleParams.Rmd"

outfile <- sprintf("steakArticle\_%s.html",regionvar)

parameters <- list(region = regionvar)

rmarkdown::render(template,

output\_file=outfile,

params=parameters)

invisible(TRUE)

}

It is worth noting that this code is based on an example demonstrated on stackoverflow at <https://stackoverflow.com/questions/41525300/how-to-generate-multiple-files-from-r-script-function-with-knitr>

This function takes 1 argument region and basically does 2 things – it creates a specific filename for the outputted HTML document based on the region name provided and runs the rmarkdown render function to compile the steakArticleParams.Rmd document with the supplied region.

Go ahead and highlight this section of R code in your R script window and click Run. This puts this new function in your local environment. Nothing happens but now we have access to this function in memory so we can test it out. Let’s run a quick test. Add another line of code in the R script window.

render\_report("Pacific")

This should run and create a new HTML output file named “steakArticle\_Pacific.html”. Take a moment and open the new file in a browser window to make sure everything worked like you expected.

This is neat, but really didn’t give us any added functionality for creating all 9 reports at once. To get the batch mode working we need to set up the rest of what we need to use the purrr functionality. Go ahead and add the rest of the R code into your steakRender.R script

library(purrr)

params\_list <- list(list("East North Central",

"East South Central", "Middle Atlantic",

"Mountain", "New England", "Pacific",

"South Atlantic", "West North Central",

"West South Central"))

purrr::pmap(params\_list,render\_report)

This code loads the purrr package into memory using the library function. The next lines of code creates a nested list (a list inside a list) of the 9 regions we want to process using our new render\_report function. To process this whole list of 9 regions through the render\_report function, we next use the pmap function from the purrr package, which is the last line of code. Go ahead and highlight these new lines of code and click Run. If everything works, this will take a few minutes to run and then in your file explorer you should see 9 new HTML files – one report for each of the 9 regions in the steak survey dataset.

Wonderful! You have now created a document template and automated it for batch processing using parameters.

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 steakArticle with parameters and all batch output”

git push

Now go to your Github repository, refresh to see your newly committed files.