Module 5 – lesson 01: Organizing Components – Files, Documents, and Codes

Script

In Module 5 we’re going to focus on organization and logistics. We’ll breakdown the components of your templates and projects and discuss how to organize them. We’ll also review the logistical details of disseminating your reproducible templates, reports and documents. We’ll also discuss how these dissemination tools help you communicate your results and work collaboratively in teams.

In this first lesson for Module 5, we’re going to leverage the work you’ve already done in Modules 3 and 4 where your built the templates for the Steak Survey and Weather Check datasets from the fivethirtyeight R package. Not only were there similarities between these 2 datasets, but we made sure the 2 report templates had common sections, layout and organization.

In this lesson, we will modularize the Weather Check template by breaking it down into smaller components. Specifically, we will separate out the textual components, code and images which make up the final document.

To get started, go ahead and create a new Github repository and a new RStudio project. Log in to your Github account and create a new repository called “Module5\_project1”, type in a description and create a readme file. Then open RStudio and create a new project – click on File/New Project/ Choose Version Control with Git and then copy and paste the URL for your “Module5\_project1” Github repository and be sure to create the new project in your local file folder for this course “C:\RepTemplates”.

Once your project is created, open the new folder on your local drive using a file explorer program. In the project folder

C:\RepTemplates\Module5\_project1

Let’s create 3 new folders – click New Folder and name the folders:

/text

/code

/images

We will use these 3 new folders to hold files with each of the components of our template. For example, in the text folder we’ll put the main contents of our background and purpose sections. In the images folder, we’ll put the sunstar PNG image (that we used in Module 2) that we’ll use to add a logo type graphic to the report. In the code folder, we’ll put the r scripts for the different r code chunks we used in the templates.

We could also have created a data folder where we could put copies of the datasets needed for our project – but for the exercises we’ve done here we don’t need a data folder since the steak survey and weather check datasets are built in to the fivethirtyeight R package, which we accessed through the R code chunks.

Go back to your “WCtemplateSimple” repository created in Module 4 and copy the “wcArticle.Rmd” template over into your new “C:\RepTemplates\Module5\_project1” folder.

In RStudio, open “wcArticle.Rmd” and save it under a new filename – click File/SaveAs “wcBase.Rmd” – this filename implies that this new R markdown file will serve as the base for our report – kind of like an outline which contains only the pieces and parts we want to include in the future.

To get started, let’s first break out the 2 main textual components of the Weather Check article – specifically the background and the purpose. To do this, we’re going to use a feature of knitr that allows “child” documents to be integrated through code chunks. We will be creating child documents for the text and R code chunks that will then be called from or referenced from the parent document “wcBase.Rmd”.

Let’s first move the Background section into a separate “child” R markdown document. First create a new text file – click File/New File/Text File. Then highlight the background section and cut and paste this into this new text file. We’re going to have this new file in the /text folder. Click File/Save As and save the text file as “backgroundSection.Rmd” but put it into the “/text” subfolder. It should be noted that by putting RMD on the end, RStudio will recognize this child document as an R markdown file and not just simple text. But we didn’t need any of the YAML header information or other template sections which is why we didn’t create a new R markdown file. The child documents do not need YAML headers since that is defined in the parent document.

Now that you’ve removed the Background section, we need to replace it with an R code chunk that references the new child document in the /text folder. Type in the following R code chunk [COPY and PASTE in Computer DEMO]

```{r child="text/backgroundSection.Rmd"}

```

Now that we’ve made this change, click Save and KNIT to HTML to make sure that the parent document “wcBase.Rmd” correctly integrated this new child document “text/backgroundSection.Rmd”

Let’s do this again for the next section on the Purpose. Create a new Text File – click File/New File/text File. Cut and paste the Purpose header and 2 paragraphs into this new text file. Save the text file in the /text folder as “purposeSection.Rmd”. Then in the parent document “wcBase.Rmd” put in a R code chunk to reference this new child document. [COPY and PASTE in here Computer Demo]

```{r child="text/purposeSection.Rmd"}

```

Again, double check that the link between the parent and child document work – Save “wcBase.Rmd” and KNIT to HTML to check.

Now that we’ve got some of our textual components broken down into smaller files, let’s also break out some of the R code chunks as well. In addition to having child documents referenced from the parent document, you can also integrate other finds of files into the parent document like R code scripts. So, let’s move the R code chunk that makes the demographic tables into a separate file. Create a new R script – click on File/New File/R script. Cut and paste the R code chunk (the code only between the 3 backticks) in the demographics section into this new R script file. Save the R script file in the “/code” folder as “makeTable.R”.

Back in the parent document “wcBase.Rmd” – we need to add one small code chunk to reference the “child” R script as follows – in this code chunk we are using the readLines function from base R to read in the R code to be executed in the R code chunk from the child R script.

```{r makeTable, code=readLines("code/makeTable.R")}

```

Let’s make sure this worked. Click Save and KNIT to HTML to check.

Let’s also separate out the R code which makes the final clustered bar plot. First create a new R script – click on File/New File/R script. Cut and paste over the R code into this new R script. Save this as “makeChart.R”. Let’s make one small change to the R script. At the top let’s add a knitr reference by adding a comment at the top – we’ll use this knitr reference to “makeCharts” when we call this code from the parent document. Save the R script with this change. This comment format used in child R scripts, can be used to combine multiple code chunks together in one R script where specific sections of code can be called from this one script. To learn more, review the details provided in the read ahead supplementary materials for this lesson.

## @knitr makeCharts

With this header added, there is an alternative way to reference this child R Script in the parent document. Type in the following R code chunk in the “wcBase.Rmd” parent document [COPY and PASTE – computer demo].

```{r}

knitr::read\_chunk("code/makeChart.R")

```

```{r makeCharts}

```

These 2 R code chunks bring in the code in 2 steps. The first R code chunk uses the command read\_chunk to read in the R script and then the next R code chunk specifically calls the code using the comment reference ## @knitr makeCharts from the “makeChart.R” R script.

Click Save and KNIT to HTML to make sure everything worked.

At this point we have created a very small parent document “wcBase.Rmd” with 2 child documents containing text and 2 child R scripts with R code for making a table and a chart for the article. While we haven’t changed any content for this particular article, we have improved the modularity of the process and we’ve improved our organization of all of the supporting materials that feed into our article. We’ll test using these modules in other ways shortly.

Let’s add one more customization to the document. Go back to your “Module2\_rmd1” github repository and project and make a copy of the sunstar.PNG image and place a copy of that file here in the “Module5\_project1” directory for “/images”.

Let’s add a few lines of R markdown and HTML syntax at the top of the document – right before the R chunk to insert the Background section. This R markdown syntax inserts the sunstar.PNG image and controls the size of the image. We will insert the image 5 times. We are also adding a little HTML syntax to center these images in the middle of the document. Refer to the read ahead supplementary materials to get the syntax. But I will insert it here [COMPUTER DEMO]

<center><hr>

![](images/sunstar.png){width=50}

![](images/sunstar.png){width=50}

![](images/sunstar.png){width=50}

![](images/sunstar.png){width=50}

![](images/sunstar.png){width=50}

<hr></center>

So, while this is probably not how you would use a company logo or icon, it should give you an idea of what is possible for inserting images in your document. And we are using a folder “images” to organize all of the images we may want to use in our document.

After inserting this R markdown and HTML syntax, click Save and KNIT to HTML to see the result.

And to completely modularize everything, let’s also move the initial setup Code chunk into a separate child R script. In RStudio, click File/New File/R script – cut and paste over the R code in the 1st code chunk. Then we’ll add 2 knitr reference markers. At the top of the script put in

## @knitr setup

Above the initial set up of the knitr options. Then right above the data command, put in this knitr reference

## @knitr loadData

Save the R script in the /code folder and name it “setup.R”.

Then back in the parent document “wcBase.Rmd” put in the following 3 R code chunks

```{r, echo=FALSE, message=FALSE, warning=FALSE }

knitr::read\_chunk("code/setup.R")

```

```{r setup, echo=FALSE, message=FALSE, warning=FALSE}

```

```{r loadData}

```

It is worth noting that since we are loading the R code from an external source inside these R scripts, until the actual knitr options get executed the R code, messages and any warnings generated by the R code will show up in your document which you do not want. So, in these R code chunks we have to include the options for echo, message and warning and set each one to false to prevent the printing of these things in our final document. By the loadData R code chunk, the knitr options have been loaded so we can skip adding these options in this and all subsequential R code chunks.

Save your document and KNIT to HTML to see the result. Let’s also do one more quick check – remember that this document was designed to work with parameters – so also click KNIT with parameters and choose a different region like Pacific and make sure that the KNIT with parameters also gives you the result you want.

Now that we’ve got everything broken down into modularized pieces – let’s use these pieces to quickly build a presentation document. This will be based on our Weather Check article using the set of text child documents and child R scripts and the sunstar image we just put together. In RStudio, click File/New File/R Markdown, choose Presentation Ioslides. Put in a title “Weather Information Preferences”.

In the YAML header, go ahead and copy and paste over your subtitle, author and date from the “wcBase.Rmd”. We also need to copy and paste over the parameters options.

After the YAML header, delete everything else in the template. Click Save and name the document “wcPresentation.Rmd”. Right after the YAML header, copy and paste over everything else from the “wcBase.Rmd” file except for the inserted sunstar images. We’re going to add the sunstar image in a different way for this presentation format.

```{r, echo=FALSE, message=FALSE, warning=FALSE}

knitr::read\_chunk("code/setup.R")

```

```{r setup, echo=FALSE, message=FALSE, warning=FALSE}

```

```{r loadData}

```

```{r child="text/backgroundSection.Rmd"}

```

```{r child="text/purposeSection.Rmd"}

```

## Demographics of Survey Respondents

This section provides demographic summary tables for the `r nrow(sdat)` survey respondents.

```{r makeTable, code=readLines("code/makeTable.R")}

```

## Weather Source Preference by Gender

Finally, here is a breakdown of the `r nrow(sdat)` survey respondents for the `r params$region` region of the US on how they check their daily weather by gender.

```{r}

knitr::read\_chunk("code/makeChart.R")

```

```{r makeCharts}

```

And let’s make more minor change to the YAML header to add a “logo” to our title slide using the sunstar image. Modify the output option to:

output:

ioslides\_presentation:

logo: images/sunstar.png

Click Save and KNIT to ioslides(HTML) to see the result. And also click KNIT with parameters and choose a different region like Pacific and make sure that the KNIT with parameters also gives you the result you want for this presentation format.

You have now successfully modularized your document content to easily build new documents in new formats and layouts leveraging these components.

Before we end this lesson, let’s back everything up to your Github repository. Open Git Bash and make sure you are in the correct directory:

C:\RepTemplates\Module5\_project1

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 template components and files”

git push

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