Module 5 – lesson 03: Communicating to your “team” on how to use your templates and projects

Script

In the previous lesson you saw how to leverage both Rpubs and Github to publish examples of the types of documents that can be produced using your reproducible templates. We did this by providing web links in your readme file on your Github repository.

However, simply providing examples of your reports is not enough. You should also provide the following information to people using your templates

1. what kind of data and formats your report template expects – for example, the Weather article needs the weather check dataset that is built into the fivethirtyeight R package

2. the file components needed to support the template – for example the “wcBase.Rmd” template assumes the document components are contained in 3 subfolders “text” “images” and “code”

3. descriptions of each of the document components – what is in each file and its purpose.

4. and it is always good to provide instructions to people wanting to use your templates on any assumptions or dependencies they need to know. For example, can child text documents or child R scripts be used in any order? In our example for the “wcBase.Rmd” template we could easily change the order of the 2 child text documents swapping the order to have the Purpose come first followed by the Background section. Likewise, we could switch the order of the R scripts such that the code to make the clustered bar chart comes before the code to make the demographics table. However, both of these R scripts assume that the setup script has been run that loads the R packages needed and loads the weather check dataset. So, the setup script has to come first at the beginning of the document.

It would also be good to include these kind of instructions in your readme file. I leave this as an exercise to do on your own. Take some time to customize your readme file. You might also share these instructions with a collegue or a friend and see if they can follow your instructions well enough to use your Github repository and the associated files to build their own reports using your templates and instructions.

Some other ways to communicate to others how to use your templates can be accomplished by providing “vignettes” through R packages. You can learn more about creating R packages and specifically adding a vignette to your R package at Karl Broman’s tutorial at <http://kbroman.org/pkg_primer/pages/vignettes.html>

Let’s add a vignette to your “WCtemplatePackage”. Open RStudio and Open your project for “C:\RepTemplates\WCtemplatePackage”. Then open a file explorer and create a New Folder called “vignettes”. Next let’s copy over the “wcArticle.Rmd” file from your “C:\RepTemplates\WCtemplateSimple” project and place it in this new “vignettes” folder

C:\RepTemplates\WCtemplatePackage\vignettes

After placing this RMD R markdown file in your new vignettes folder, we need to also edit your DESCRIPTION file in the “WCtemplatePackage”. Open the DESCRIPTION file and add these 2 lines.

Suggests: knitr, rmarkdown

VignetteBuilder: knitr

Next we need to build the vignettes for your package using this command – type this into your Console window in RStudio

devtools::build\_vignettes()

When this finishes the final HTML file and supporting files will be in the /inst/doc folder for your package.

Now use Git Bash to sync up these changes with your repository.

git add .

git commit –m “add a vignette”

git push

git status

Then go to your Github repository and refresh to see the added vignette files.

Now let’s reinstall your package with the updated vignette. To install this off of your Github repository, type in the devtools command to install the package from github. Be sure to use your Github account name.

devtools::install\_github('melindahiggins2001/WCtemplatePackage')

In RStudio click on the Packages TAB and scroll down to find the installed WCtemplatePackage and click on it to bring up the HELP file for your package. At the top you’ll see a link for “User guides, package vignettes and other documentation”. When you click on this, it brings up a page that lists any Vignettes that come with your package. You could have more than one. If you had more than one vignette they would be listed here. Right now we just have one. Click on the link for the wcArticle to see the HTML for the wcArticle.

This is just a quick example, but you could spend some time developing R markdown files to create different Vignettes that illustrate different types of templates or layouts or options that are possible with your R package.

One example to take a look at to get ideas is the CRAN website for the dplyr package. This website lists 5 vignettes that illustrate different capabilities of the dplyr package.

<https://cran.r-project.org/web/packages/dplyr/index.html>

And in RStudio you can click on the link for the dplyr Package and view the vignettes in the HELP viewer inside RStudio also.

So, you’ve now learned 3 different ways of communicating information and instructions to others on how to use your templates and packages. You can publish examples using RPubs. You can serve HTML files using Github Pages in the docs folder of your Github repository. And if you convert your template project into an R package, you can include vignettes to illustrate how to use your template.

And through the R package you really have a 4th way of communicating through the template itself which is available to others through when they create a new R markdown template after they install your package. So, the template itself can provide the instructions and details.