Introduction to RStudio and R markdown

Kim Dill-McFarland

version January 23, 2018

Contents

Review Data Science Friday Git from last week

Today's introductory activity aims to ensure that you feel comfortable working with your local copies of your GitHub repo and updating your remote (located on the website) version of that repo. The ultimate goal is to update the ID.txt document that we created last class period with your name and student ID number.

The goal: edit ID.txt on your local machine to add your name (First, Preferred, Last) and student ID, then get that edited version on your remote repo (located on the website).

Work in groups of 3-4 to determine the commands you would use to:

1. navigate to your local version of MICB425_portfolio
2. edit ID.txt with your name and student ID

- This may differ for Mac and PC users
- You may do this in the command line or any text editor but make sure the document remains a .txt

3. get your edited ID.txt file to the local index/staging space
4. check if step 3 was successful
5. tell Git that you want to make the changes final to your local repo (on your computer)
6. update your remote repo (online) with your now modified local repo (on your computer)

Review Git answer

1. `cd ~/Documents/MICB425_portfolio`
2. `nano ID.txt`

- nano is our recommended command line text editor. It comes pre-installed on Macs and can be downloaded for PCs here
- Other options include vim, emacs, Notepad++...

3. `git add .`
4. `git status`
5. `git commit -m "Updated ID.txt"`

- Remember to use `-m` to include a message about what changes your are committing

6. `git push origin`

Introduction

Throughout this course, you will be creating a course portfolio in R markdown (Rmd). This is a text-based language ideal for making reproducible reports of all your analyses in RStudio. In fact, this is what we've made all the Data Science Friday assignments and tutorials in! It also works very well with GitHub because it is entirely text-based, so we can track and control versions.

In order to complete your portfolio in Rmd, we must first orient ourselves in RStudio.

RStudio

RStudio is a graphical user interface (GUI) for R, which is a command line-esque program. When you open RStudio, you will see several parts of the program.

On the left you have the console, which is where you will run commands. On the top-right, you have several tabs. We will us the Environment to see data that is loaded in R. On the bottom-right, you also have several tabs, which will come into play during this tutorial.

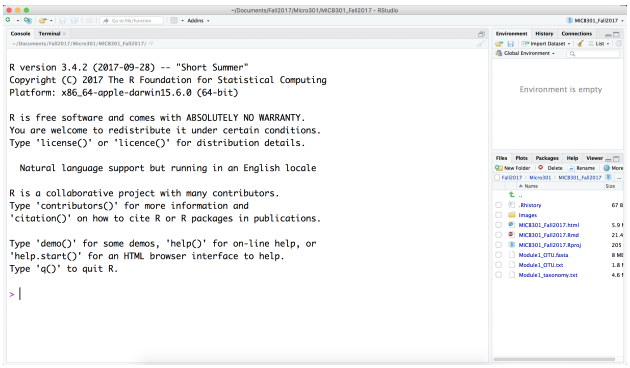


Figure 1:

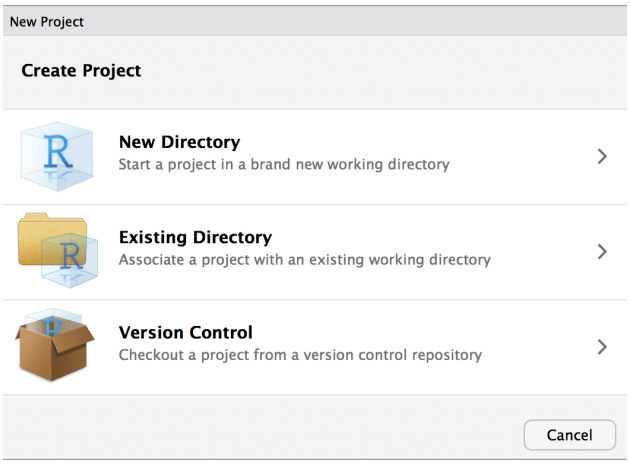


Figure 2:

To change the look of RStudio, you can go to Tools -> Global Options -> Appearance and select colors, font size, etc. If you plan to be working for longer periods, I suggest choosing a dark background color scheme to save your computer battery and your eyes.

RStudio projects

Projects in RStudio help us stay organized. When you use a project, RStudio creates a `.Rproj` file that links all of your files and outputs to the project folder. When you import data, R automatically looks for the file in the project folder instead of you having to specify a full file path like `/Users/username/Desktop/`. R also automatically saves any output to the project folder. Finally, projects allow us to save your R environment in `.RData` so that when we close RStudio and then re-open it, we can start right where we left off without re-importing any data or re-calculating any intermediate steps. We will see these project features as we begin to work with data in RStudio.

To begin and stay organized, we will start a new R project, File -> New Project

Create this project in your MICB425_portfolio repo (which we created last time).

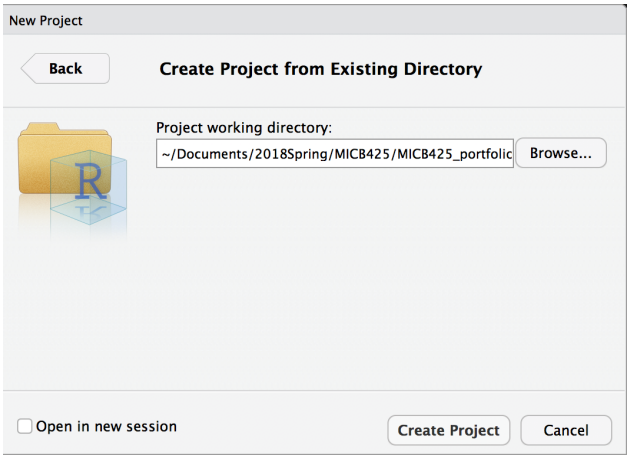


Figure 3:

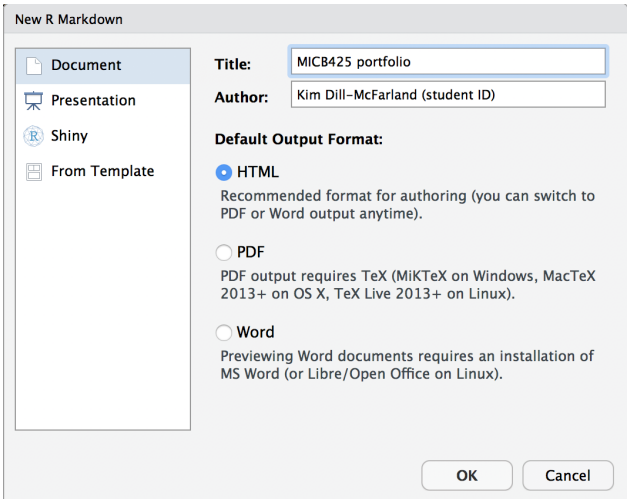


Figure 4:

Saving R code

Another important step to stay organized is R scripts. These are text files in which to save your R code. If you just type code into the Console (left side in RStudio) and run it, it will not be saved anywhere. Thus, when you come back to a project, you will not know what you did in the past.

Since we will be creating portfolios in R markdown, we will use this version of an R script to save code in addition to adding formatting like headers, tables, etc. useful in your portfolio.

Let's create a new R markdown to save your work. Go to File -> New File -> R Markdown. Input a title, your name and student ID, and select the html output format.

Now, you will see the left side split between your R markdown (top) and the Console (bottom).

Your R markdown is automatically populated with some example code. You can delete everything outside of the top text bracketed by "- - -".

Note that when we save our R markdown, it is automatically saved in our project folder!

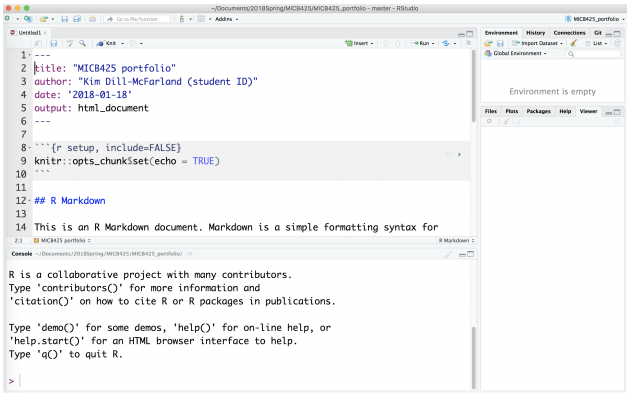


Figure 5:

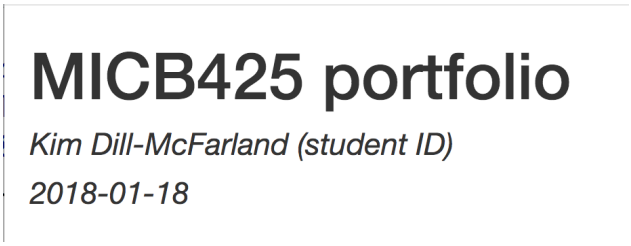


Figure 6:

Please save your R markdown as StudentID_MICB425_portfolio.Rmd

GitHub

Now that you have an `.Rproj` and `.Rmd`, be sure to add, commit, and push them!

Knitting

Knitting is the process by which R takes an R markdown, evaluates all the text and code, and creates a beautiful report in PDF, html, or Word format. Since we set our output to html when we made our R markdown, when we click 'Knit' above the markdown, it will knit to an html. Since we are in a project, this file is created and saved in your project folder.

When you knit a document for the first time, you will be asked to install some packages. Please allow all of these installs within RStudio.

Help

You can get help with any function in R by inputting `?function name` into the Console. This will open a window in the bottom right under the Help tab with information on that function, including input options and example code.

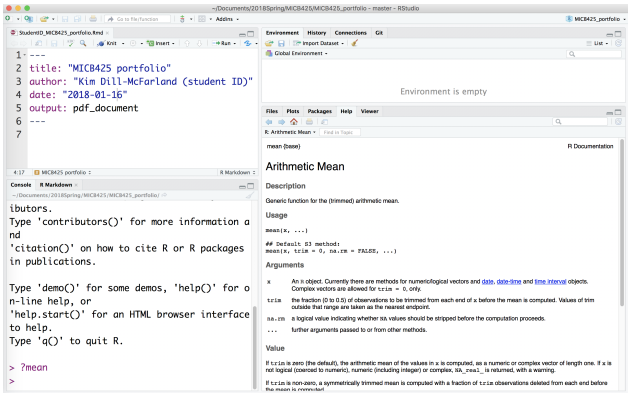


Figure 7:

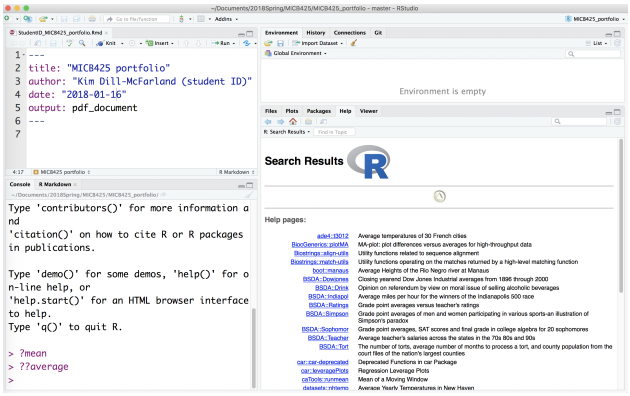


Figure 8:

```
?mean
```

If you are unsure of the exact function name, you can expand the help search to include all descriptions with `??function name`. For example, if you did not remember that R uses `mean` to calculate an average, you could use `??average`. This will bring up a list of functions with 'average' somewhere in the name or description for you to choose from.

```
??average
```

Working in R markdown

YAML header

The YAML header is the area between the "- - -" at the top of an R markdown.

This is written in YAML (YAML Ain't Markup Language), a human friendly data serialization standard for all programming languages. Basically, YAML allows us to easily read the header both within RStudio and after we knit into a report.

In general, we will not edit the header as RStudio automatically populates it with what we need.

MICB425 portfolio

Kim Dill-McFarland (student ID)

version January 18, 2018

Figure 9:

For example, your header should have your title, name, date, and output type as a result of the information you input when creating the R markdown.

The one thing we will change is the date so that R will automatically update the date every time you create a report.

Change the date section of your YAML header to be

```
date: "version `r format(Sys.time(), '%B %d, %Y')`"
```

When you knit, this looks like

Other headers

R markdown designates headers with # with each # pushing the header down one level.

```
# Example header 1
## Example header 2
## Example header 3
```

Example header 1

Example header 2

Example header 3

You should utilize the portfolio template provided in your MICB425_materials repo to see how module 01 headers should be organized. Please also follow a similar format for later modules in this course.

Table of contents

You can insert a table of contents by adding `toc: yes` under the output part of your header. This will take all of your headers and organize them into a table of contents automatically.

```
---
title: "MICB425 portfolio"
author: "Kim Dill-McFarland (student ID)"
```

```
date: "version `r format(Sys.time(), '%B %d, %Y')`"
output: pdf_document
  toc: yes
---
```

Text

You can treat a markdown like any text document outside of special formatting like # for headers. Simply type away and plain text will appear in your portfolio.

You can modify this text by surrounding it with special characters like below.

```
*italicized text*
```

These characters include

- * for *italics*
- ** for **bold**
- ^ for $^{\text{superscript}}$
- ~~ for ~~strikethrough~~

Line breaks

Ending a line of text with 2 spaces results in a line break like

line 1
line 2

while 2 hard returns (Enter) yields

line 1

line 2

Note that using 1 hard return (Enter) does not yield a line break at all.

line 1 line 2

Tables

I will say that the one thing not easy to format in R markdown is tables. A standard table looks like

```
Row names | Data 1 | Data 2
--------- | ------ | ------
Row 1     |   0    |    1
Row 2     |   1    |    0
```

to make

Row names	Data 1	Data 2
Row 1	0	1

Row names	Data 1	Data 2
Row 2	1	0

Please note that the | lines do not all have to line up to make a table, but it makes it much easier to read in the R markdown before knitting.

Feel free to explore other table options through packages like

- kable (already loaded as part of RStudio)
- xtable
- stargazer
- pander
- tables
- ascii

Links to websites

You can simply paste in the URL https://github.com/EDUCE-UBC/MICB425 or link it to a word or phrase with

```
[text](https://github.com/EDUCE-UBC/MICB425)
```

text

Links to images

If your image is online:

```
![Image title](http://)
```

If your image is on your computer:

```
![Image title](~/Documents/MICB425_portfolio/image.png)
```

If your image is in your project folder, you don't need the entire file path.

```
![Image title](image.png)
```

All of the above will insert the image within your knit document. By default, the image will be whatever size the file is or span the entire page if it is larger. You can change the size by modifying images as below.

```
![Image title](image.png){width=50%}
```

Code chunks

Arguably the most powerful aspect of working in R markdown (as opposed to other markdown formats or a simple text file) is the ability to include analyses from R in-line within the document.

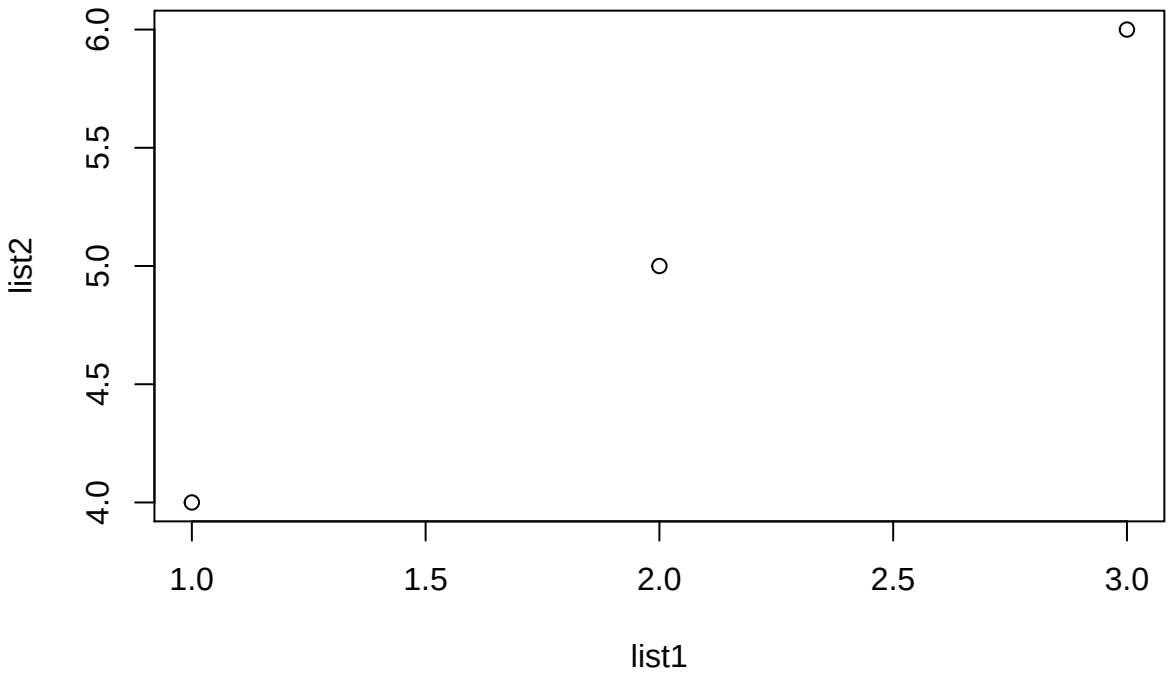
This code is designated with

```{r}
Code, code, code
```

When code chunks are knit, R evaluates the code and prints both the code and the output into your report. *Note that within a code chunk, comments (i.e. text, notes, etc.) are designated by the #.*

```
#Create 2 lists of numbers
list1 = c(1,2,3)
list2 = c(4,5,6)

#Plot list1 vs. list2
plot(x=list1, y=list2)
```

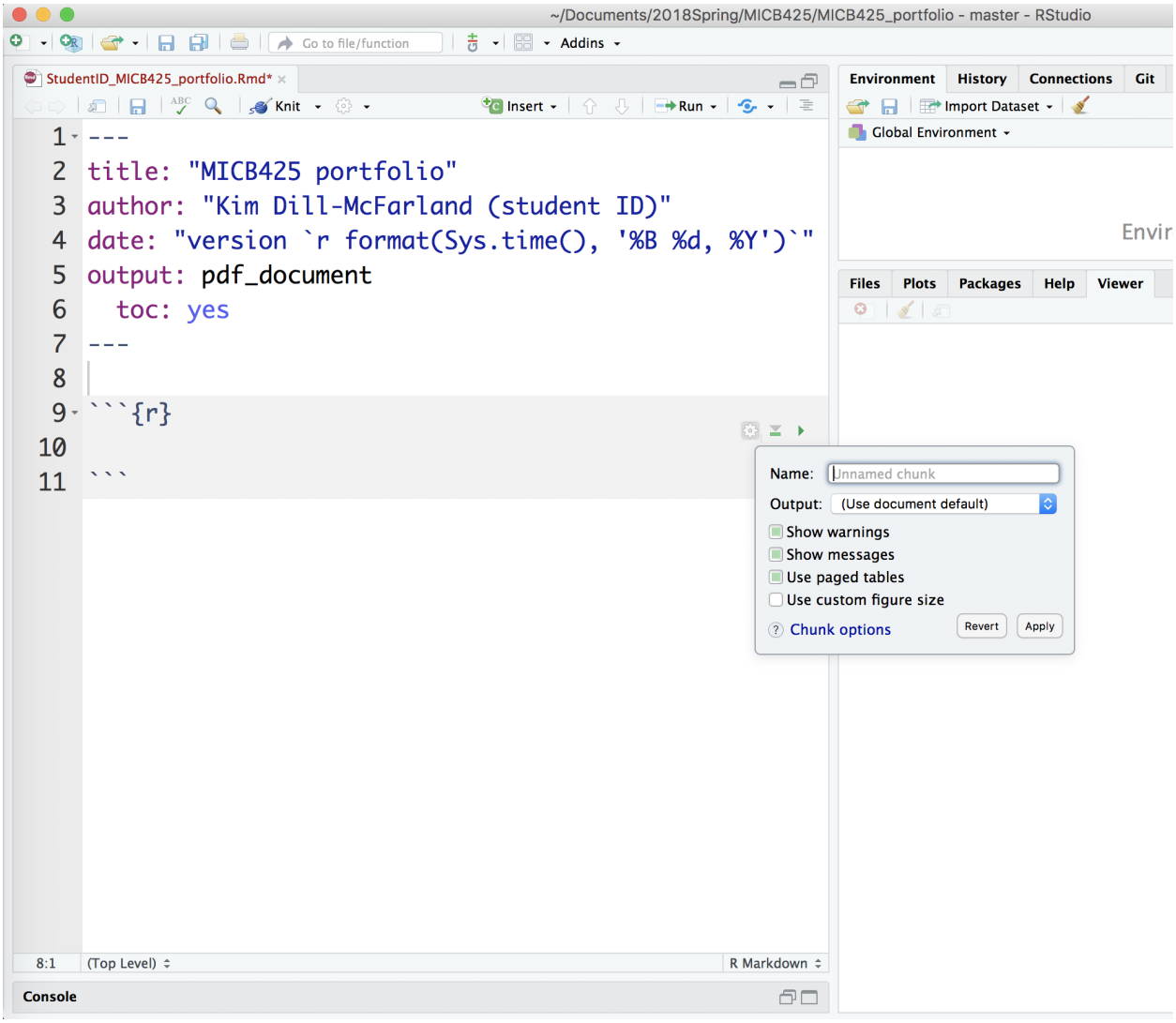


These code chunks can also be customized in what they output (code only, no warnings, figure size, etc.).

When you modify a chunk, you add parameters within `{r }` such as

```{r title, message=TRUE, warning=FALSE, fig.width=4}
```

For a more complete list of options, see here.


~/Documents/2018Spring/MICB425/MICB425_portfolio - master - RStudio
Go to file/function
Addins
StudentID_MICB425_portfolio.Rmd*
Knit
Insert
Run
---
title: "MICB425 portfolio"
author: "Kim Dill-McFarland (student ID)"
date: "version `r format(Sys.time(), '%B %d, %Y')`"
output: pdf_document
toc: yes

```{r}
```
Environment
History
Connections
Git
Import Dataset
Global Environment
Files
Plots
Packages
Help
Viewer
Name:
Unnamed chunk
Output:
(Use document default)
Show warnings
Show messages
Use paged tables
Use custom figure size
Chunk options
Revert
Apply
8:1
(Top Level)
R Markdown
Console

Figure 10:

Running code

As you work in RStudio, you can check that your code works by

- Clicking "Run" in the upper right of the R markdown window. This will run all of the code in your markdown.
- Selecting an option under "Run" to only run some of the code.
- Placing your cursor on a line of code (do not need to select the entire line) and hitting ctrl+Enter or cmd+Enter

More resources

This document demonstrates just some of the functions in an R markdown. We will continue to learn options as your portfolio progresses in this course and you are encouraged to seek out code that works best for your portfolio.

- R markdown tour
- More R markdown examples
- R markdown cheatsheet
- Other R cheatsheets