Activity 1.2

# RStudio and R Markdown

Requirements:

* GitHub account
* RStudio Cloud account

Goals:

* Introduce you to R and RStudio
* Demonstrate the workflow to make versioning simpler

## Introduction

In this activity you are introduced to R (the statistical programming language) and RStudio (the interface) that we will be using throughout the course. You are provided with specific tasks to complete here, but I encourage you to explore beyond what is dictated. A willingness to experiment will make you much better at programming.

This activity will have you interact with the R and RStudio interface, reading in data, and basic commands. We still are not getting to collaboration with GitHub so while you work through this lab, help your team members.

## Getting Started

Activities and assignments will begin with similar steps. They will be outlined here in detail, but become more sparse as the semester progresses. You can always refer back to this activity for a detailed list of steps for getting started.

1. Click on the activity link that is posted on [Bb](https://mybb.gvsu.edu) (Documents > Activities). This will create your GitHub repository (repo) for this activity that contains a template you will build on
2. On GitHub, click on the green “Clone or download” button

* Select “Use HTTPS” (this is probably selected by default)
* Click on the clipboard icon to copy the repo URL

1. In RStudio Cloud go into the course workspace to create a “New Project from Git Repo”.

* You will need to click on the down arrow next to the “New Project” button to see this option
* Paste the URL of your assignment URL into the dialogue box
* Click “OK” and you are (almost) good to go!

### R Packages

For this activity, we will work with two R packages: - datasauRus which contains the dataset, and - tidyverse which is a collection of packages for doing data analysis in a “tidy” way.

Install these by running the following code in the *Console* (bottom left-hand pane).

install.packages("tidyverse")  
install.packages("datasauRus")

Now that the necessary packages are install, you should be able to proceed.

### Configure Communication with GitHub

Before we go crazy with R, we need to configure your Git so that RStudio can communicate with GitHub. This requires two details: the email address attached to your GitHub account and your name. To configure this, do the following steps:

* Go to the *Terminal* pane (second tab in the bottom left-hand pane - next to [*Console*](https://r4ds.had.co.nz/workflow-scripts.html))
* Type the following to lines of code, replacing the information in the quotation marks with your info (keep the quotes):

git config --global user.email "your email"  
git config --global user.name "your name"

* For example, I have done

git config --global user.email "dykesb@gvsu.edu"`  
git config --global user.name "B Dykes"

* Confirm that the changes have been implements by running the following code:

git config --global user.email  
git config --global user.name

We are now going to see how RStudio and GitHub communicate. First, we need to make some changes.

### Project Name

Currently your project is called “Untitled Project”. Update the name of your project to be “Activity 1.2 - RStudio and R Markdown” (near the top left-hand corner of your browser).

### YAML

Go to the *Files* pane (first tab in the bottom right-hand pane) and open the R Markdown (.Rmd) file in your project. R Markdown combines the core syntax of markdown with embedded R code chunks so that we can easily create output within our final documents.

The top portion of a R Markdown file contains YAML (between the three dashed lines). “YAML” is a self-referencing (it sands for *YAML Ain’t Markup Language*) human-readable data-serialization standard for all programming languages. You simply need to know that this area is called the YAML and it contains meta information about the document.

Change the author name to you name, the date to today, and knit (icon at the top of the Rmd pane) the document.

### Commit Changes

Go to the Git pane (fourth tab in the upper right-hand pane) in RStudio.

When you make changes to your Rmd file, you should see them listed here. Click on it in this list, then click on **Diff**. This shows you the *difference* between the last committed state of the document and its current state - including your changes.

If you are happy with these changes, write “Update author name” in the **Commit message** dialogue box and click on **Commit**.

You do not have to commit after every change as this would get rather annoying. Instead, you should consider committing after *meaningful* (to you) changes to help make inspection, comparison, or restoration easy. During the first few activities, I will let you know when to commit and (in some cases) what commit message to use. As we progress throughout the semester, I will gladly relinquish that power to you.

### Pushing Changes



Push it

You have made an update and committed this change, it is now time to push these changes to your repo on GitHub. Why? So that others (namely, me, for now) can see your changes. Your repos in this course are private and so only you, me, and (when we begin group repos) your team members can see your repos.

To push your changes to GitHub, click on **Push**. This will prompt a dialogue box where you will first need to enter your user name and then your GitHub password. For now we will continue doing this method, but later I will show you how to save your password so you don’t need to enter it every time.

Now, go back to your GitHub repo (at GitHub.com) and verify that this change has been made. You may need to refresh your browser page.

Discuss with your team members: For which of the above steps do you need to have an internet connection?

## Data

The data frame that we will be working with is called datasaurus\_dozen and it is in the datasauRus package. This data frame is actually a baker’s dozen (13) of datasets, designed to show us why data visualization is important and how summary statistics alone can be misleading (you may have seen [Anscombe’s Quartet](https://en.wikipedia.org/wiki/Anscombe%27s_quartet) in previous classes).

To find out more about the dataset, type the following in your *Console* in R Studio: ?datasuarus\_dozen. A question mark before the name of an R object will bring up its help file. This command *must* be ran in the *Console*.

**Exercise 1:** Based on the help file, how many rows and how many columns does the datasaurus\_dozen file have? What are the variables included in the data frame? Add your informative responses to your activity report Rmd file. When you are done, commit your changes with the commit message, “Added answer for Ex 1”, and push.

While you work on this activity with you team, you may want to test code out in your *Console* before adding it to your Rmd document. Therefore, you should also load the packages in your *Console*. To do so, run the following in the *Console*.

library(tidyverse)  
library(datasauRus)

Note that the packages are loaded with the same commands in your Rmd document.

The original Datasaurus dataset (dino) was created by [Alberto Cario](http://www.thefunctionalart.com/2016/08/download-datasaurus-never-trust-summary.html). The other *dozen* were generated using simulated annealing and the process is described in Justin Matejka and George Fitzmaurice’s [paper](https://dl.acm.org/citation.cfm?id=3025912) where the authors simulate a variety of datasets that produce the same summary statistics as the Datasaurus, but have very different distributions.

We can peak into what these 13 datasets are with a frequency table of the dataset variable:

datasaurus\_dozen %>%  
 count(dataset) %>%  
 print(13)

## # A tibble: 13 x 2  
## dataset n  
## <chr> <int>  
## 1 away 142  
## 2 bullseye 142  
## 3 circle 142  
## 4 dino 142  
## 5 dots 142  
## 6 h\_lines 142  
## 7 high\_lines 142  
## 8 slant\_down 142  
## 9 slant\_up 142  
## 10 star 142  
## 11 v\_lines 142  
## 12 wide\_lines 142  
## 13 x\_shape 142

The print(13) line forces R to print all 13 lines.

### Data Visualization and Summary

The code you will need to complete this exercise is provided below, but you will need to include the relevant bits in your Rmd document and successfully knit and view the results.

**Exercise 2:** Plot y vs. x for the dino dataset. Then, calculate the correlation coefficient between x and y for this dataset.

Start with the datasaurus\_dozen, *then* filter it to only include observations where dataset == "dino". Assign the resulting filtered data frame as a new data frame called dino\_data.

dino\_data <- datasaurus\_dozen %>%   
 filter(dataset == "dino")

Let’s unpack this code.

First, the assignment operator <- assigns the name dino\_data to the final filtered data.

Second, the pipe operator %>% takes what comes before it and sends it as the first argument to what comes after it. This is like saying, "Take the datasaurus\_dozen data frame then filter it for observations where data == "dino".

From STA 216, this is like the following in SAS:

DATA dino\_data;  
SET datasaurus\_dozen;  
WHERE dataset = "dino";  
RUN;

Now we need to visualize the data! We will use the ggplot function for this. Its first argument is the data that you are visualizing. Next, we define the aesthetic mappings (aes). In other words, the columns of the data that get mapped to certain aesthetic features of the plot (e.g., the x and y axis will be called x and y, respectively). Then, we add (+) another layer to this plot where we define which geometric shapes (geom) we want to use to represent the observations. In this case we want to use points (geom\_point).

ggplot(data = dino\_data, mapping = aes(x = x, y = y)) +  
 geom\_point()

We are going to go into great detail of this and the philosophy of building data visualizations in layers in the next learning module. For now, just follow along with the provided code.

To do the second part of this exercise, we need to calculate a summary statistic: - the correlation coefficient. Recall that this measures the strength and direction of the linear association between two quantitative variables. You will see that some of the pairs of variables we plot do not have a linear relationship between them. This is why we want to visualize first!

* Visualize to assess the form of the relationship, and
* Calculate only if relevant.

In this case, calculating a correlation coefficient doesn’t make sense because the relationship is between x and y is not linear. How would you describe its shape?

For illustrative purposes, we will now calculate between x and y.

dino\_data %>%   
 summarize(r = cor(x, y))

This code says, “Start with the dino\_data, then calculate a summary statistic (summarize) that we will call as the correlation (cor) between x and y.”

*Verify your results with your team members, then commit changes with the commit message, “Added answer for Ex 2”, and push.*

**Exercise 3:** Plot y vs. x for the star dataset. Then, calculate the correlation coefficient between x and y for this dataset. How does this value compare to the of dino?

You can (and should) reuse the code we introduced above, just replace the information with the desired names.

*Verify your results with your team members, then commit changes with the commit message, “Added answer for Ex 3”, and push.*

**Exercise 4:** Plot y vs. x for the circle dataset. Then, calculate the correlation coefficient between x and y for this dataset. How does this value compare to the of dino?

You can (and should) reuse the code we introduced above, just replace the information with the desired names.

*Verify your results with your team members, then commit changes with the commit message, “Added answer for Ex 4”, and push.*

**Exercise 5:** Now we will plot all datasets at once. To do this, we will make use of faceting.

ggplot(data = datasaurus\_dozen, aes(x = x, y = y, color = dataset)) +  
 geom\_point() +  
 facet\_wrap(~ dataset, ncol = 3) +  
 theme(legend.position = "none")

As a team, discuss the following parts of this code: What did the color = dataset do? What did the ncol = 3 option in the facet\_wrap do? What did the theme(legend.position = "none") do?

We will discuss these as a class after everyone has done the final commit at the end of this activity.

Now we will use the group\_by function to generate all the summary correlation coefficients.

datasaurus\_dozen %>%  
 group\_by(dataset) %>%  
 summarize(r = cor(x, y)) %>%  
 print(13)

You’re done with the data analysis exercises, but I want you to spend some time fancifying up your Rmd document.

### Re-size Your Figures

* Click on the gear icon in on top of the R Markdown document, and select “Output Options…” in the drop-down menu.
* In the pop up dialogue box go to the *Figures* tab and change the height and width of the figures.
* Click on “OK” when done.
* Knit your document and see how you like the new sizes.
* Change and knit again and again until you’re happy with the figure sizes.

Note that these values get saved in the YAML (another way that you can make these changes).

### Change the Look of Your Report

* Again, click on the gear icon in on top of the R Markdown document, and select “Output Options…” in the drop-down menu.
* In the *General* tab of the pop up dialogue box try out different Syntax highlighting and theme options.
* Click on “OK” and knit your document to see how it looks.
* Play around with these until you’re happy with the look.

**You are done!** *Commit all remaining changes, use the commit message, “Done with Activity 1.2! 😎”*

Double check that your GitHub repo has been updated.

## Reflection

As a team, discuss your responses to these questions.

What concept, if any, remains fuzziest for your team? How could what we learned in this activity be applied in your current/future careers?