Beneduci Exam 1

Zachary Beneduci

2023-03-08

**Question 1.** You are looking at a collaborator’s R code on GitHub, and download the repository, and start exploring the code. The First line of the script is

setwd(“C:/Users/…”)

* What is the author of this code trying to do with the function setwd()?
* Please discuss what is wrong with this approach in terms of reproducibility.
* Where is the working directory of an **R project**?
* Explain the concept of *relative* file paths. Is the author of this code using *relative* file paths?

**Answer 1.** The author is trying to set the working directory, or the location where project-associated files are read and saved, for their R project. This author’s approach is a problem because it assumes that the reader has the same local file structure as the author, which is likely not the case. Additionally, this forces the reader to manually locate and download any project-associated files, assuming these are made available somewhere. The working directory of an R project is a file path that was user-defined at the creation of the project. It is also where the .Rproj file is saved. Relative file paths are those that are relative to the working directory of the R.proj. This can be useful as a shortened file path can be given when loading files. It is also more reproducible, as a reader could download your project to a different file location and still call the files with relative file paths. The author is not using relative file paths, as they are calling the direct file path with “C:/Users/…”

**Question 2.** What does the acronym FAIR stand for in the context of this class? Expain how R, GitHub, and other lecture concepts introduced in this course specifically help complete FAIR data principles.

**Answer 2.** The acronym FAIR stands for 1) Findable – whether your workflow can be located, 2) Accessible – whether the found workflow can be accessed by others, 3) Interpretable – whether the steps taken by the author, and which are present in the workflow, can be interpreted without the author’s outside explanation, and 4) Reusable – the data and workflow is described in sufficient detail, presented in a format usable by common software, and obtainable for use by others.

R can be used to improve accessibility, interpretability, and reusability. R presents elements of workflows in an open-source format with wide usage and support across scientific disciplines. Additionally, R code can be written and annotated in such a way as any researcher can download the script and data files and re-run the author’s analysis from raw data all the way to finished products (figures, tables, etc.). This improves the reader’s ability to understand the step-by-step analysis (interpretability) and aids with the reuse of data.

Github predominantly improves accessibility. However, Github can help to make data available, and reusable. Github is a remote repository, allowing the author to upload data and other project-associated files. This allows readers to access, download, and use the files provided (depending on licensing). Github can improve findability, for example, if the author uses their name as their Github username and formats their repository in a straightforward and easy-to-understand manner (i.e. simple/obvious naming scheme for files and paths).

Data management plans are helpful for ensuring the findability of data. A data management plan should outline the data and outputs of research, how they were generated, and where they are stored and can be accessed. Data management plans should provide sufficient documentation and explicitly describe the location and naming of files necessary for the reproduction of original research.

**Question 3.** Explain the concept of R packages. What are R packages? Who writes R packages? What is the difference between installing and loading a package? Explain two ways to install and load packages into R.

**Answer 3.** R packages are collections of code that create new functions (and sometimes come with data) to be used in R. R packages can be written by anyone. Packages only need to be installed once, but must be loaded at the start of every new R session to be used. Packages can be installed by navigating to the “Packages” tab in RStudio, clicking the “Install” icon, specifying the name of the package(s) in the resulting pop-up window, and clicking “Install.” Alternatively, one can use the function “install.packages(”insert package name”)” either in the console or an R script. To load a package, one can either navigate to the “Packages” tab, find their desired package in the list, and clicking the empty box. Packages can also be loaded by using the function “library(insert package name)”.

**Question 4.** Explain the following concepts of ggplot and give examples of each concept using code and figures generated with ggplot using the data of your choosing.

* Layering
* Scales
* Themes
* Facets

**Answer 4.**

###### Layering

Figure generation with ggplot works by adding *layers* or *geoms* to plot different components of a figure. For example, one can install and load needed packages:

#install.packages("ggplot2") # Install the package if needed.  
#install.packages("lme4")  
library(ggplot2) # Load the ggplot library.

## Warning: package 'ggplot2' was built under R version 4.2.2

library(lme4) # Load the lme4 library for the iris dataset.

## Warning: package 'lme4' was built under R version 4.2.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.2.2

Load a dataset:

data("iris")

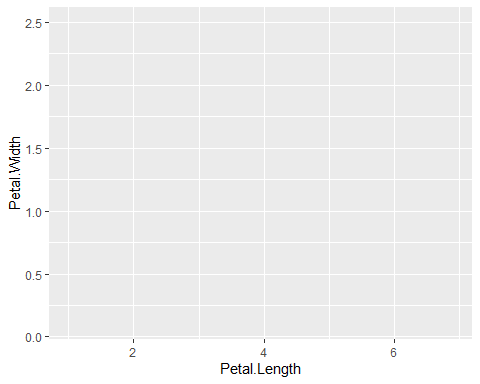
And plot a base layer:

ggplot()



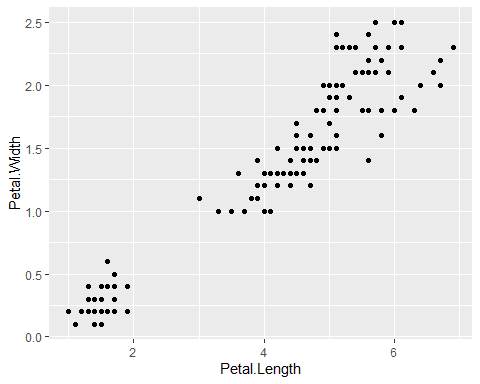
Add aesthetic mapping for the x and y variables:

ggplot(iris, aes(Petal.Length, Petal.Width))



Add a layer of individual points:

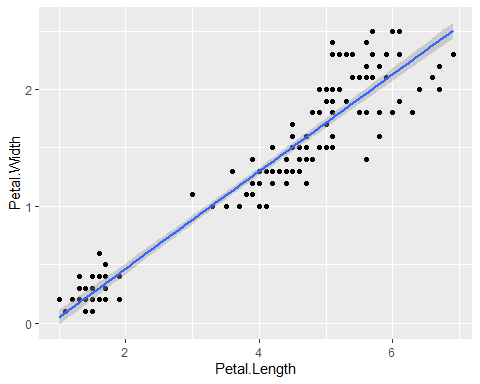
ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point()



And finally, add a trend line as a layer:

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point() + geom\_smooth(method = lm)

## `geom\_smooth()` using formula = 'y ~ x'



Order matters for the visualization of geoms. Those specified first in the code will appear in the background; those last will appear in the foreground.

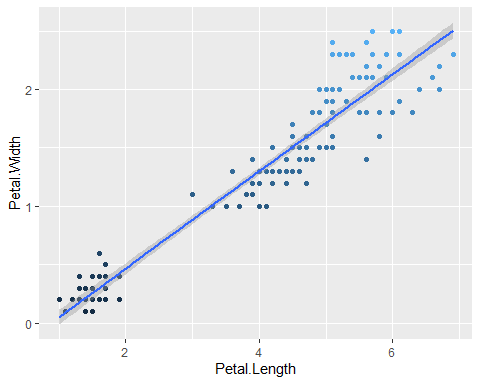
###### Scaling

*Scales* are used in ggplot to map the size, shape, color, and location of aesthetics. One can manipulate these with different functions.

Adding to the above plot, we can apply the default color gradient across values of flower petal width:

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point(aes(color = Petal.Width), show.legend = FALSE) + geom\_smooth(method = lm)

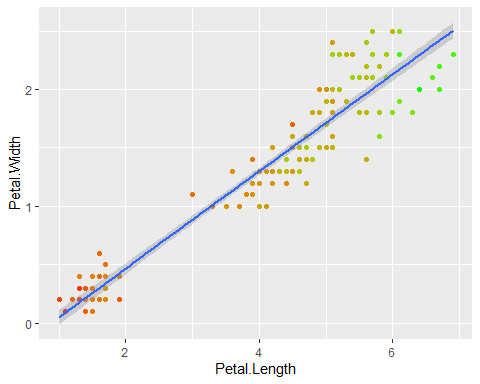
## `geom\_smooth()` using formula = 'y ~ x'



We can then change the color gradient to display low values in red and high values in green like so:

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point(aes(color = Sepal.Length), show.legend = FALSE) + geom\_smooth(method = lm) + scale\_color\_gradient(low = "Red", high = "Green")

## `geom\_smooth()` using formula = 'y ~ x'

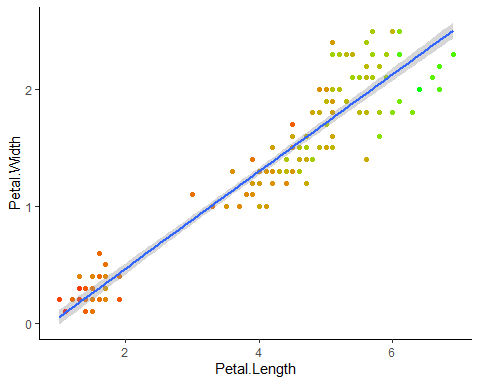


###### Themes

*Themes* can be added to ggplots to apply a set of pre-defined aesthetics without manipulating every little detail manually. For example, the classic theme is popular:

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point(aes(color = Sepal.Length), show.legend = FALSE) + geom\_smooth(method = lm) + scale\_color\_gradient(low = "Red", high = "Green") + theme\_classic()

## `geom\_smooth()` using formula = 'y ~ x'

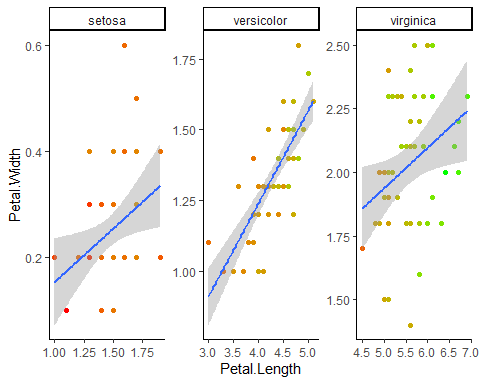


###### Facets

Finally, *facets* allow the user to subset the visualization of data without creating a separate ggplot figure. We can facet the iris dataset by species:

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom\_point(aes(color = Sepal.Length), show.legend = FALSE) + geom\_smooth(method = lm) + scale\_color\_gradient(low = "Red", high = "Green") + theme\_classic() + facet\_wrap(~Species, scales = "free") # Free the scales between species for minimized blank space.

## `geom\_smooth()` using formula = 'y ~ x'



**Question 5.** Explain the differences and similarities between a vector, matrix, and dataframe. Demonstrate you know how to subset a dataframe in two ways using the built in dataset “ToothGrowth with the prompts below:

* Subset ToothGrowth to include rows such that supp is equal to VC
* Subset ToothGrowth to include rows such that supp is equal to VC and dose is equal to 0.5
* Subset ToothGrowth to include the values of len such that supp is equal to VC

**Answer 5.**

A *vector* is an object of multiple scalar elements (i.e. numbers). It can have multiple rows but only one column. A *matrix* is an object that contains only one data class, such as numeric or character. A matrix could be considered (at least visually) as being made of multiple vectors. A matrix can have multiple rows AND columns. A *dataframe* is an object containing multiple data classes. It can be thought of as a matrix with the possibility of different data classes between columns.

Load the data for subsetting demonstration:

data("ToothGrowth")

To subset ToothGrowth to include rows such that supp is equal to VC:

ToothGrowth[ToothGrowth$supp == "VC",] # Subset with bracketting.

## len supp dose  
## 1 4.2 VC 0.5  
## 2 11.5 VC 0.5  
## 3 7.3 VC 0.5  
## 4 5.8 VC 0.5  
## 5 6.4 VC 0.5  
## 6 10.0 VC 0.5  
## 7 11.2 VC 0.5  
## 8 11.2 VC 0.5  
## 9 5.2 VC 0.5  
## 10 7.0 VC 0.5  
## 11 16.5 VC 1.0  
## 12 16.5 VC 1.0  
## 13 15.2 VC 1.0  
## 14 17.3 VC 1.0  
## 15 22.5 VC 1.0  
## 16 17.3 VC 1.0  
## 17 13.6 VC 1.0  
## 18 14.5 VC 1.0  
## 19 18.8 VC 1.0  
## 20 15.5 VC 1.0  
## 21 23.6 VC 2.0  
## 22 18.5 VC 2.0  
## 23 33.9 VC 2.0  
## 24 25.5 VC 2.0  
## 25 26.4 VC 2.0  
## 26 32.5 VC 2.0  
## 27 26.7 VC 2.0  
## 28 21.5 VC 2.0  
## 29 23.3 VC 2.0  
## 30 29.5 VC 2.0

subset(ToothGrowth, supp == "VC") # Using the subset function.

## len supp dose  
## 1 4.2 VC 0.5  
## 2 11.5 VC 0.5  
## 3 7.3 VC 0.5  
## 4 5.8 VC 0.5  
## 5 6.4 VC 0.5  
## 6 10.0 VC 0.5  
## 7 11.2 VC 0.5  
## 8 11.2 VC 0.5  
## 9 5.2 VC 0.5  
## 10 7.0 VC 0.5  
## 11 16.5 VC 1.0  
## 12 16.5 VC 1.0  
## 13 15.2 VC 1.0  
## 14 17.3 VC 1.0  
## 15 22.5 VC 1.0  
## 16 17.3 VC 1.0  
## 17 13.6 VC 1.0  
## 18 14.5 VC 1.0  
## 19 18.8 VC 1.0  
## 20 15.5 VC 1.0  
## 21 23.6 VC 2.0  
## 22 18.5 VC 2.0  
## 23 33.9 VC 2.0  
## 24 25.5 VC 2.0  
## 25 26.4 VC 2.0  
## 26 32.5 VC 2.0  
## 27 26.7 VC 2.0  
## 28 21.5 VC 2.0  
## 29 23.3 VC 2.0  
## 30 29.5 VC 2.0

To subset ToothGrowth to include rows such that supp is equal to VC and dose is equal to 0.5:

ToothGrowth[(ToothGrowth$supp == "VC") & (ToothGrowth$dose = 0.5),] # Subset with brackets.

## len supp dose  
## 1 4.2 VC 0.5  
## 2 11.5 VC 0.5  
## 3 7.3 VC 0.5  
## 4 5.8 VC 0.5  
## 5 6.4 VC 0.5  
## 6 10.0 VC 0.5  
## 7 11.2 VC 0.5  
## 8 11.2 VC 0.5  
## 9 5.2 VC 0.5  
## 10 7.0 VC 0.5  
## 11 16.5 VC 1.0  
## 12 16.5 VC 1.0  
## 13 15.2 VC 1.0  
## 14 17.3 VC 1.0  
## 15 22.5 VC 1.0  
## 16 17.3 VC 1.0  
## 17 13.6 VC 1.0  
## 18 14.5 VC 1.0  
## 19 18.8 VC 1.0  
## 20 15.5 VC 1.0  
## 21 23.6 VC 2.0  
## 22 18.5 VC 2.0  
## 23 33.9 VC 2.0  
## 24 25.5 VC 2.0  
## 25 26.4 VC 2.0  
## 26 32.5 VC 2.0  
## 27 26.7 VC 2.0  
## 28 21.5 VC 2.0  
## 29 23.3 VC 2.0  
## 30 29.5 VC 2.0

subset(ToothGrowth, supp == "VC" & dose == 0.5,) # Using the subset function.

## len supp dose  
## 1 4.2 VC 0.5  
## 2 11.5 VC 0.5  
## 3 7.3 VC 0.5  
## 4 5.8 VC 0.5  
## 5 6.4 VC 0.5  
## 6 10.0 VC 0.5  
## 7 11.2 VC 0.5  
## 8 11.2 VC 0.5  
## 9 5.2 VC 0.5  
## 10 7.0 VC 0.5  
## 11 16.5 VC 0.5  
## 12 16.5 VC 0.5  
## 13 15.2 VC 0.5  
## 14 17.3 VC 0.5  
## 15 22.5 VC 0.5  
## 16 17.3 VC 0.5  
## 17 13.6 VC 0.5  
## 18 14.5 VC 0.5  
## 19 18.8 VC 0.5  
## 20 15.5 VC 0.5  
## 21 23.6 VC 0.5  
## 22 18.5 VC 0.5  
## 23 33.9 VC 0.5  
## 24 25.5 VC 0.5  
## 25 26.4 VC 0.5  
## 26 32.5 VC 0.5  
## 27 26.7 VC 0.5  
## 28 21.5 VC 0.5  
## 29 23.3 VC 0.5  
## 30 29.5 VC 0.5

Subset ToothGrowth to include values of len such that supp is equal to VC and dose is equal to 0.5:

ToothGrowth$len[(ToothGrowth$supp == "VC") & (ToothGrowth$dose == 0.5)] # Using bracketting.

## [1] 4.2 11.5 7.3 5.8 6.4 10.0 11.2 11.2 5.2 7.0 16.5 16.5 15.2 17.3 22.5  
## [16] 17.3 13.6 14.5 18.8 15.5 23.6 18.5 33.9 25.5 26.4 32.5 26.7 21.5 23.3 29.5

subset(ToothGrowth$len, (ToothGrowth$supp == "VC") & (ToothGrowth$dose == 0.5)) # Using the subset function.

## [1] 4.2 11.5 7.3 5.8 6.4 10.0 11.2 11.2 5.2 7.0 16.5 16.5 15.2 17.3 22.5  
## [16] 17.3 13.6 14.5 18.8 15.5 23.6 18.5 33.9 25.5 26.4 32.5 26.7 21.5 23.3 29.5

**Question 6.** Create an R markdown version of your answer to questions 4 and 5. Save the .Rmd file to your computer and render it as a word document (.docx), .html, and a .md file. Push these files to your GitHub and paste your GitHub url here

**Answer 6.**

**Question 7.** What is the correct order of events to get your code on GitHub through RStudio? Explain each step from creation of a repository to pushing.

**Answer 7.**

* In RStudio, go to file and select “New Project”.
* Select “Version Control” under “Create a Project”.
* Select “Git” as the version control type.
* If you do not have a Git repository, navigate to your browser and login to GitHub.
* Navigate to the “Repositories” tab and click the “New” icon.
* Give the repository a name and select your desired license.
* Inside your new repository, select the green “Code” icon and copy the url of your repository.
* Paste this url into the Git project wizard, name the local copy of your repository, and specify the file path where you would like it to be locally stored.

Your R project should now be linked with Git. To upload a file within your project to GitHub:

* Navigate to File then select Save.
* Navigate to the Git tab at the top right of RStudio.
* Click the box of the file you just saved, which should place a checkmark in that box.
* Select Commit just above.
* Write a message for this commit and press Commit at the bottom of the window.
* Close the resulting window, and now press Push (the green arrow).

Your files should now be findable on GitHub!

**Question 8.** After you have worked on a project for a while, you mistakenly delete a file on your GitHub, while it still exists in your local repository (on your computer). Now when you try to push your code to GitHub the push is rejected and gives the following error, “***Updates were rejected because the remote contains work that you do not have locally***.” How do you solve this?

**Answer 8.**

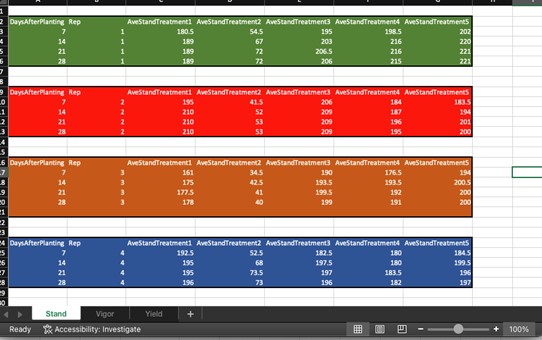
It looks like you’d receive that error for deleting a file on GitHub rather than locally. If this is the case, one can simply pull the project from GitHub.

**Question 9.** Explain the purpose of a Data Management Plan.

**Answer 9.**

The purpose of a Data Management Plan is to describe what you will do with your collected data. It should outline the expected types of data generated, the data formats, consistency with standards of the discipline or those set by the funding agency, the storage and preservation of the data, the plans to share, protect, and provide public access to the data, the roles and responsibilities of those people and parties involved with the project, as well as the budget (if needed) to maintain this information. The data management plan should detailed clearly and concisely.

**Question 10.** A colleague gives you data in an .xlsx file that looks like this:

 Please discuss at least five things wrong with how these data are formatted that make it not reproducibility friendly. Then describe/show your colleague how the data should be formatted.

**Answer 10.**

1. These data are saved as a .xlsx. Saving as a .csv file is considered more reproducible, as the comma separated file format is recognizable by most programs involving data analysis.
2. The data should not be separated by empty rows. A program like R will not know how to interpret those blank spaces correctly.
3. Ideally, data should be organized in long format. The treatment columns should be replaced by two columns: one titled “Treatment” with the cells containing the name of the treatment and another titled “StandAverage” with cells containing the measurement.
4. The data should not be separated into different sheets. When reading data into R, only the active sheet will be recognized. Instead, my colleague should use a single datasheet and create a new column titled “Characteristic” containing either “Stand”, “Vigor”, or “Yield”.
5. It is unclear what the different treatments are. The colors are additionally meaningless without other supporting information. The treatment names should be human readable, such as abbreviations that most people will understand.
6. Each individual measurement should be given a unique ID so as to keep track of total observations and avoid duplicates.
7. A column containing the date of the sample should be included, as it’s unclear if observations sharing the same number of DaysAfterPlanting share the same sampling date.

The data should be saved as a .csv and presented in long format. Columns of DaysAfterPlanting, Date, Replicate (1, 2, 3, …), Treatment (1-5, given relevant names), Characteristic (StandAverage, Vigor, or Yield) and Value (containing the numerical measurement) should be included. This format should allow all measurements to be contained on one sheet.