Example Problem Integration

Paityn Richardson

5.2: Using R for Data Calculations

R is a programming language used for statistical analysis, data manipulation, and visualizations. When programming in R, users can work in R studio which provides an accessible platform to write and reproduce code. You will be working in R studio in what is called a Quarto Document to integrate both code and written work and best present your data.

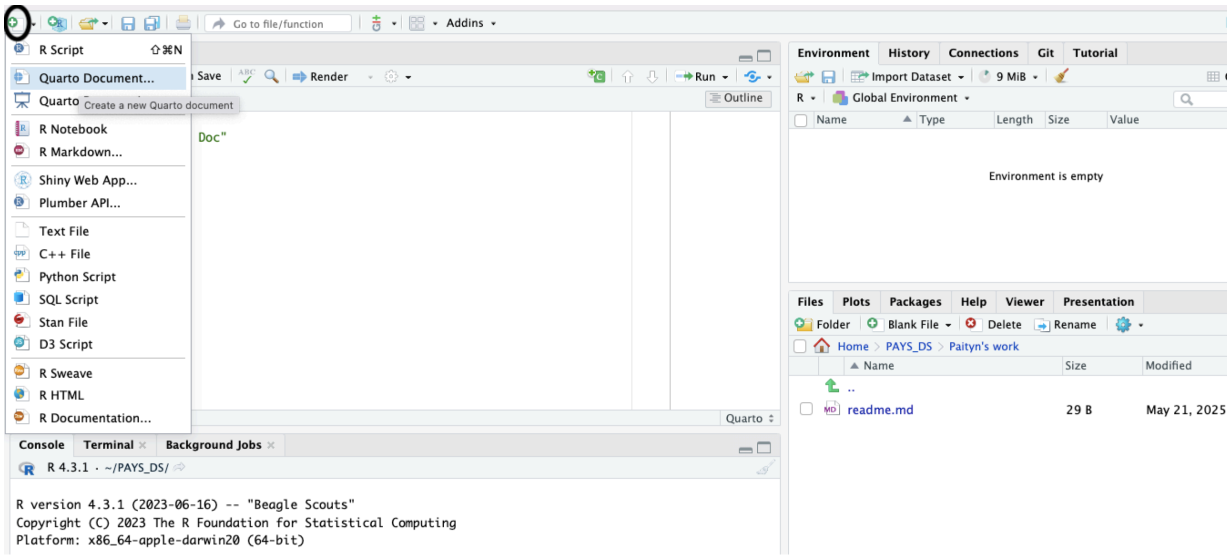
Understanding the Data

When beginning to work with a data set, it it important to understand the structure and features that define it. Let's say you're working on a science project that is dealing with the speeds of different animals.

You've measured how long your dog takes to run from the far corner of the back yard to the back door and measured that distance. Your neighbors have agreed to join join experiment and take the same measurements for their pet cat, hamster and snail. It would be same to assume that you want to compare the given speeds of each animal, meaning, in this case, animal type would be your {\bf unit of observation}. The unit of observation is the variable in your data that is described by the other observations, the unit by which you will analyze your observations.

Writing in R

Lets look at this example by creating a data frame in R. For small data sets such as this one, you can go ahead and enter the observations directly into R. First, begin by creating a new Quarto Document in R studio.



```
R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20 (64-bit)
```

To start entering data, we need to create a portion of our quarto document that will execute r code. Type “““{r}”, and a grey rectangle should appear on the line that looks something like this.

```
install.packages("tidyverse")
```

Now, anything written in this block should be code in the r programming language. Lets start entering our Data. We have 3 variables in the data we are working with. Animals, our observational unit, Distance, and Time. These are our variable names. We will create objects by assigning the values of our observations to their respective variable with a backwards arrow as shown below.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
animal <- c("Dog", "Cat", "Hamster", "Snail")
```

Here, we have assigned the observations “Dog”, “Cat”, “Hamster”, and “Snail” to the variable “animal” Now, do the same with the observations for Distance and time. Your work should now look something like this.

```r
library(tidyverse)

animal <- c("Dog", "Cat", "Hamster", "Snail")
distance <- c(20,10,3,.1)
time <- c(2.5,.8,.6,10)
```

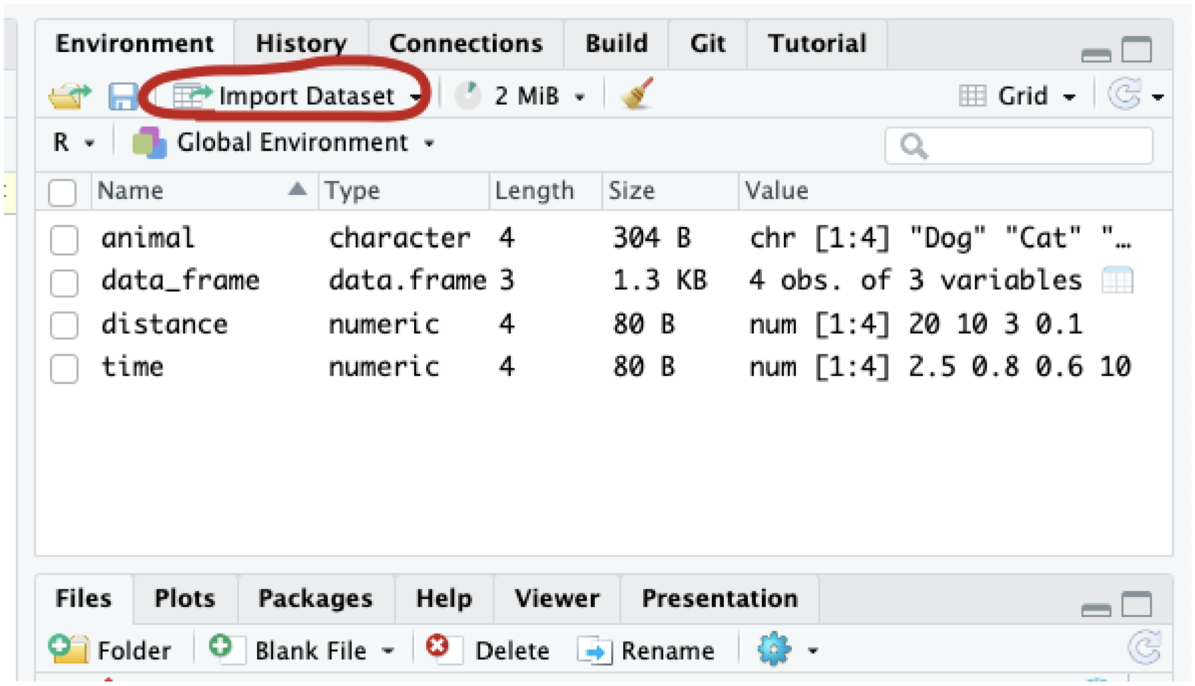
Notice how there are no quotation marks around the observations for distance and time. That is because these variable are quantitative, meaning their values represent numerical amounts, as opposed to the variable animal. Animal is a categorical variable, as its observations represent different groups or categories. Finally, we want to arrange these items, so they appear neatly in a data frame. You may have been wondering what this line of code refers to.

```r
library(tidyverse)
```

The tidyverse is package in R with tools designed for data wrangling and exploration. We need to install this package and load it into our library in order to use its functions in our code. it is good to note that, once you have installed a package on your device, you only must call call the package from your library to use it again. To combine these objects into a data frame, we will use the tidyverse function “data.frame” and set each object equal to the name of its column. We also want to make sure we assign a name to the object that is created when using the function.

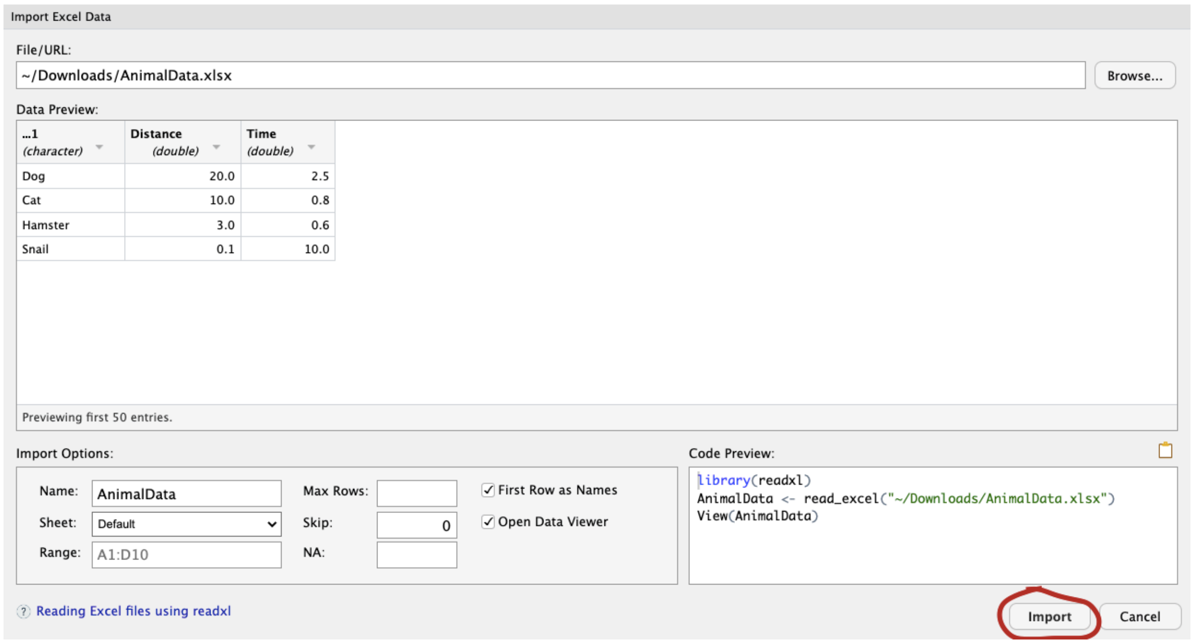
```r
library(tidyverse)

animal <- c("Dog", "Cat", "Hamster", "Snail")
distance <- c(20,10,3,.1)
time <- c(2.5,.8,.6,10)

data_frame <- data.frame(Animal = animal, Distance = distance, Time = time)
```

Now we have an organized data frame, with all of our variables saves as objects to use for later calculations. If the data you are working with is any larger than a few observations, it is much more practical to type the data into excel and import that file into r. Animal is the observational unit, so this variable should define your rows. Each of the other variables should be labeled at the top of their column. Export your spreadsheet as a csv file and import it into r using this button in the environment tab.



Name	Type	Length	Size	Value
animal	character	4	304 B	chr [1:4] "Dog" "Cat" "…
data_frame	data.frame	3	1.3 KB	4 obs. of 3 variables
distance	numeric	4	80 B	num [1:4] 20 10 3 0.1
time	numeric	4	80 B	num [1:4] 2.5 0.8 0.6 10

From the drop down menu, you should import from excel, and upload your downloaded file. After you import from this screen, your data set will be saved as an object in your environment.



File/URL: ~/Downloads/AnimalData.xlsx

Data Preview:

...1 (character)	Distance (double)	Time (double)
Dog	20.0	2.5
Cat	10.0	0.8
Hamster	3.0	0.6
Snail	0.1	10.0

Previewing first 50 entries.

Import Options:

Name: AnimalData
Sheet: Default
Range: A1:D10
Max Rows:
Skip: 0
NA:
✓ First Row as Names
✓ Open Data Viewer

Code Preview:

```
library(readxl)
AnimalData <- read_excel("~/Downloads/AnimalData.xlsx")
View(AnimalData)
```

Doing Calculations

Here is where r can really start doing the heavy lifting for you Remember, in this example we wanted to compare the given speed of each animal. To do this we must calculate the speed by computing the equation distance/time for each animal. In r, however, you cn write one simple block off code to compute this value for each animal, and also add it as a variable in your data frame. We spoke about the tidyverse earlier and here we will use another one of its data wrangling functions. "Mutate" is a function that allows you create a new column in your data frame based on calculations with other existing variables. In this case we are creating a variable "speed" that is defined by the equation "distance/time".

```r
new_data <- data_frame |>
  mutate(speed = Distance / Time)

new_data
```

```
   Animal Distance Time speed
1     Dog     20.0  2.5  8.00
2     Cat     10.0  0.8 12.50
3 Hamster      3.0  0.6  5.00
4   Snail      0.1 10.0  0.01
```

When using tidyverse functions such as mutate, it is important to use what we call the pipe operator, in order to connect the information from your data set to the next line of code. Think of this symbol "|>" as a link between lines that carries data information. If we were to add additional lines, we would need to include the pip operator at the end of each line.

Basic Graphing

Curve Fitting