Importing_Data_Crena

Andrew Crena

2023-05-19

First, we will learn how to set a working directory for an entire Rmd file, so that you don't have to worry about calling to it in later chunks. We will also learn how to specify explicit directories in specific chunks, but this method serves as an efficient way to centralize your directory for a lengthy or complex markdown file. This is actually accomplished in the first code chunk of this Rmd file, above "R Markdown". WRITE THIS ARGUMENT INTO THAT KNITR FUNCTION (look above):

```
# This chunk is meant to set my wd for the project I am planning on completing. Look to other Rmd's of
setwd("C:\\Users\\adcre\\OneDrive\\Desktop\\Desktop_RStudio")
```

R Markdown

Importing Data in the form of CSV Files

First things first, I will provide various ways of setting your working directory and knowing what directory you are pulling from and when. In the past, I have written an entire R Markdown file without telling the computer what directory I want to pull from. This became increasingly frustrating when I started changing file locations and working directories in such a way that prevented me from setting a working directory for the entire markdown file. Thus, I am creating this code chunk as a way of "ironing-out any wrinkles" when it comes to remaining hyperaware of the working directory you are interested in.

```
# First, we will learn how to set a working directory for an entire Rmd file, so that you don't have to
# [ root.dir = "C:\Users\adcre\OneDrive\Documents\R_Programming" ]

# This may seem obvious, but setting a working directory at the top of a specific code chunk will allow
 # setwd("C:\\Users\\adcre\\OneDrive\\Documents\\R_Programming")
# getwd()
```

This chunk of code is specifically for importing csv files into your working directory. This is operating under the assumption that your working directory contains the csv files being pulled in this chunk. The arguments are good opportunity to tell your system about any incorrect/missing values, so make sure to memorize them (atleast the common and relevant ones)!

```
# setwd("C:\\Users\\adcre\\OneDrive\\Documents\\R_Programming")

library(readr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v purrr     1.0.1
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## -- 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 become errors
```

```
library(dplyr)

Brain_Data <- read_csv("Brain_Data.csv")
```

```
## Rows: 49 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Group
## dbl (6): ID, RSFC1, RSFC2, RSFC3, RSFC4, RSFC5
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
Bx_Data <- read_csv("Bx_Data.csv")
```

```
## Rows: 55 Columns: 32
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): SubjID, Group
## dbl (30): ID, Gender, Sex, Ethnicity, Race, Puberty, Age_at_Bx, Age_at_Scan,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str((Bx_Data))
```

```
## spc_tbl_ [55 x 32] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID                            : num [1:55] 1 2 3 4 5 6 7 8 9 10 ...
##  $ SubjID                        : chr [1:55] "01-T1" "02-T1" "03-T1" "04-T1" ...
##  $ Group                         : chr [1:55] "MDD" "MDD" "MDD" "MDD" ...
##  $ Gender                        : num [1:55] 2 1 2 1 2 2 2 2 3 1 ...
##  $ Sex                           : num [1:55] 2 1 2 1 2 2 2 2 2 1 ...
##  $ Ethnicity                     : num [1:55] 2 2 1 2 1 2 2 2 1 2 ...
##  $ Race                          : num [1:55] 5 6 7 5 6 2 5 2 1 5 ...
##  $ Puberty                       : num [1:55] 3.5 4 3.5 4.5 4 4 5 4.5 4.5 5 ...
```

```
##  $ Age_at_Bx                       : num [1:55] 15.2 13.6 15.7 15.1 14.3 ...
##  $ Age_at_Scan                     : num [1:55] 15.2 13.7 15.8 15.1 14.3 ...
##  $ Height                          : num [1:55] 63.5 67.5 59.1 70.5 64 62.1 62 67 63.8 66.9 ...
##  $ Weight                          : num [1:55] 167 107.6 91.4 125.4 179 ...
##  $ Bx_to_Scan_Days                 : num [1:55] 12 8 29 7 7 5 11 7 14 13 ...
##  $ CDRSR_total                     : num [1:55] 53 33 50 42 65 49 49 81 70 61 ...
##  $ CDRSR_tscore                    : num [1:55] 70 58 68 64 80.5 67.5 67.5 86 83 77 ...
##  $ Current_Med                     : num [1:55] 0 0 0 1 1 1 1 0 1 1 ...
##  $ Recurrent_Past_Month            : num [1:55] 1 3 1 3 3 3 3 3 3 1 ...
##  $ Recurrent_Current_Duration_Weeks: num [1:55] NA 2 NA 156 364 156 312 8 676 NA ...
##  $ Age_First_SI                    : num [1:55] NA 12 13 13 10 12 14 13 8 NA ...
##  $ SI_Group                        : num [1:55] 0 1 1 1 1 1 1 1 1 0 ...
##  $ SI_Past_Month                   : num [1:55] 1 3 2 1 3 2 1 3 3 1 ...
##  $ Hx.Attempt                      : num [1:55] 0 1 1 0 1 0 0 0 1 0 ...
##  $ Age.First.Attempt               : num [1:55] NA 13 15 NA 10 NA NA NA 10 NA ...
##  $ Number.of.Attempts              : num [1:55] 0 2 1 0 3 0 0 0 5 0 ...
##  $ RADS_DM                         : num [1:55] 26 26 25 25 27 30 27 32 32 30 ...
##  $ RADS_AN                         : num [1:55] 11 19 13 18 10 19 13 25 16 16 ...
##  $ RADS_NS                         : num [1:55] 16 32 21 25 29 25 21 29 29 19 ...
##  $ RADS_SC                         : num [1:55] 21 22 18 18 28 23 17 26 26 22 ...
##  $ RADS_total                      : num [1:55] 74 99 77 86 94 97 78 112 103 87 ...
##  $ PHQ9                            : num [1:55] 19 14 18 16 19 21 16 23 23 21 ...
##  $ PHQ9_q9                         : num [1:55] 1 3 0 1 3 2 1 2 2 2 ...
##  $ SIQ                             : num [1:55] 10 50 17 42 34 75 34 74 52 11 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   SubjID = col_character(),
##   ..   Group = col_character(),
##   ..   Gender = col_double(),
##   ..   Sex = col_double(),
##   ..   Ethnicity = col_double(),
##   ..   Race = col_double(),
##   ..   Puberty = col_double(),
##   ..   Age_at_Bx = col_double(),
##   ..   Age_at_Scan = col_double(),
##   ..   Height = col_double(),
##   ..   Weight = col_double(),
##   ..   Bx_to_Scan_Days = col_double(),
##   ..   CDRSR_total = col_double(),
##   ..   CDRSR_tscore = col_double(),
##   ..   Current_Med = col_double(),
##   ..   Recurrent_Past_Month = col_double(),
##   ..   Recurrent_Current_Duration_Weeks = col_double(),
##   ..   Age_First_SI = col_double(),
##   ..   SI_Group = col_double(),
##   ..   SI_Past_Month = col_double(),
##   ..   Hx.Attempt = col_double(),
##   ..   Age.First.Attempt = col_double(),
##   ..   Number.of.Attempts = col_double(),
##   ..   RADS_DM = col_double(),
##   ..   RADS_AN = col_double(),
##   ..   RADS_NS = col_double(),
##   ..   RADS_SC = col_double(),
```

```
##   ..   RADS_total = col_double(),
##   ..   PHQ9 = col_double(),
##   ..   PHQ9_q9 = col_double(),
##   ..   SIQ = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

It's important to remember that th data you are importing could have missing values, incorrect values, and other pieces of data that need to be attended to by the software. Using the arguments in this chunk is a good example of weeding out the missing values and other data that isn't wanted.

What if we are downloading csv files from the internet to do some exploratory data analysis? For example, what if i found an interesting data set on Kaggle?

This next code chunk is an example of downloading a data set from the internet so that it is expressed as a data frame in your working directory. It is a depression data set randomly chosen on Kaggle, and I want to write the code that will grab it from the specific URL that refers to the specific data set, or some code that will achieve the same goal but in different ways. Using Kaggle data usually requires installing the Kaggle package, and having permission or an API token to use the data for exploratory analysis. the data set can be found with this link: https://www.kaggle.com/datasets/arashnic/the-depression-dataset ALWAYS REMEMBER DIRECTORIES!!!! Where's your data coming from/going to?

```
# setwd("C:\\Users\\adcre\\OneDrive\\Documents\\R_Programming")

# Assuming the csv file has been extracted and is in your working directory....
# We won't focus on additional arguments, so that we can focus on one thing at a time

library(readr)
scores <- read_csv("scores.csv")
```

In this example, we will show how to get the data set into a df after downloading it and extracting the zip file (I find this to be easiest at my current skill level)

```
## Rows: 55 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): number, age, edu
## dbl (9): days, gender, afftype, melanch, inpatient, marriage, work, madrs1, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
View(scores)

# Next, view your assigned variable to make sure that your data frame was made properly
View(scores)
```

```r
# Instead of using the view function, you can also use the str or "structure" function, which will give
str(scores)
```

```
## spc_tbl_ [55 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ number   : chr [1:55] "condition_1" "condition_2" "condition_3" "condition_4" ...
##  $ days     : num [1:55] 11 18 13 13 13 7 11 5 13 9 ...
##  $ gender   : num [1:55] 2 2 1 2 2 1 1 2 2 2 ...
##  $ age      : chr [1:55] "35-39" "40-44" "45-49" "25-29" ...
##  $ afftype  : num [1:55] 2 1 2 2 2 2 1 2 1 2 ...
##  $ melanch  : num [1:55] 2 2 2 2 2 2 NA NA NA 2 ...
##  $ inpatient: num [1:55] 2 2 2 2 2 2 2 2 2 2 ...
##  $ edu      : chr [1:55] "6-10" "6-10" "6-10" "11-15" ...
##  $ marriage : num [1:55] 1 2 2 1 2 1 2 1 1 1 ...
##  $ work     : num [1:55] 2 2 2 1 2 2 1 2 2 2 ...
##  $ madrs1   : num [1:55] 19 24 24 20 26 18 24 20 26 28 ...
##  $ madrs2   : num [1:55] 19 11 25 16 26 15 25 16 26 21 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   number = col_character(),
##   ..   days = col_double(),
##   ..   gender = col_double(),
##   ..   age = col_character(),
##   ..   afftype = col_double(),
##   ..   melanch = col_double(),
##   ..   inpatient = col_double(),
##   ..   edu = col_character(),
##   ..   marriage = col_double(),
##   ..   work = col_double(),
##   ..   madrs1 = col_double(),
##   ..   madrs2 = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Remember that the bottom right quadrant is excellent for doing these same tasks, just not programatically. However, let's practice unzipping a file obtained from the internet!

```
## Warning in unzip("DEP_DATA_KAGGLE.zip"): error 1 in extracting from zip file
```

```
## Rows: 55 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): number, age, edu
## dbl (9): days, gender, afftype, melanch, inpatient, marriage, work, madrs1, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## spc_tbl_ [55 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ number   : chr [1:55] "condition_1" "condition_2" "condition_3" "condition_4" ...
##  $ days     : num [1:55] 11 18 13 13 13 7 11 5 13 9 ...
##  $ gender   : num [1:55] 2 2 1 2 2 1 1 2 2 2 ...
##  $ age      : chr [1:55] "35-39" "40-44" "45-49" "25-29" ...
```

```
##  $ afftype  : num [1:55] 2 1 2 2 2 2 1 2 1 2 ...
##  $ melanch  : num [1:55] 2 2 2 2 2 2 NA NA NA 2 ...
##  $ inpatient: num [1:55] 2 2 2 2 2 2 2 2 2 2 ...
##  $ edu      : chr [1:55] "6-10" "6-10" "6-10" "11-15" ...
##  $ marriage : num [1:55] 1 2 2 1 2 1 2 1 1 1 ...
##  $ work     : num [1:55] 2 2 2 1 2 2 1 2 2 2 ...
##  $ madrs1   : num [1:55] 19 24 24 20 26 18 24 20 26 28 ...
##  $ madrs2   : num [1:55] 19 11 25 16 26 15 25 16 26 21 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   number = col_character(),
##   ..   days = col_double(),
##   ..   gender = col_double(),
##   ..   age = col_character(),
##   ..   afftype = col_double(),
##   ..   melanch = col_double(),
##   ..   inpatient = col_double(),
##   ..   edu = col_character(),
##   ..   marriage = col_double(),
##   ..   work = col_double(),
##   ..   madrs1 = col_double(),
##   ..   madrs2 = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

REFLECTION

I first want to make note of a mistake I made when setting working directories within an R Markdown file, and running the code through the console expecting the computer to know where the data is at all times. After speaking with GPT, it appears that I failed to establish a reliable method of knowing what directory I am pulling from (or at least, attempting to pull from).

Since this is the first Rmd project I will be using in the future, I want to purposely leave it alone as much as possible with regard to the way I solve the problems and the aesthetic nature of the code. I think looking at this Rmd, I can see so many ways in which I have already improved. This includes decision-making, structure, purpose, and many other things that I can reflect on by appreciating how far I have come, and using these mistakes as learning lessons.