Problem Set 5

Due March 1, 2023

Instructions

- Read all of these instructions closely.
- This problem set is due Tuesday, March 1, 2023 at 4pm.
- Submit files via Github:
    1. the .Rmd (R Markdown) file
    2. the knitted .pdf file
    3. anything else the particular problem set might require
- Use a copy of this file, perhaps with your name or initials appended to the file name, to write your answers to the questions. You'll see there is a designated space where your answers should begin.
- Knitting the .Rmd file to a .pdf file *as you work* will ensure your code runs without errors and is working how you expect. Knit early and often. You've already read the instruction that a knitted .pdf is required when you submit.
- Per the syllabus, I will not accept any late work. Keep in mind the two lowest problem set scores are dropped. Turn in what you have.
- Clarification on the expectations for problem set submissions (posted in Slack, copied here):
    - Always print the output of the code I'm requesting.
        - Ex: If I want you to create a vector x with elements 1 through 10, print x after creating it so I can see it worked.
    - Write any written answers in the space outside the code chunk, not inside with an R comment.
        - R comments are great to clarify code, but not for answering the question.
    - Make sure any code or written content is not cut off in the pdf.
        - This really should only apply to code, because if you follow item 2 in this list, the pdf will compile your written answers nicely.

Question 1–Computer Skills

This question asks you to practice navigating file directories on your computer. It also provides additional practice on writing efficient for loops.

Navigating directories can be a little funky in Rmd files. Rmd files are great for creating clean reports and problem sets, which usually don't require a lot of movement around directories. **I want to focus on the basics, so you will write your answers to Question 1 in a .R file.**. Clearly label your work with comments in the .R file (1a, 1b, . . .).

If you would benefit from a review of working directorys, I like this resource.

1a

- Write code to navigate to the course's Github folder on your computer. We want to point R *inside* this folder.
- Print the output of the `getwd()` function to demonstrate you have navigated to the right place.
- Also print the output of `list.files()` function to demonstrate you have navigated to the right place.

For example, on my computer, I use the `~` symbol as shorthand for `/Users/erossiter/` on Mac computers. Then, I point inside the "documents" folder > point inside the "Github" folder > point inside

"ProgrammingSpring2023" folder.

```
setwd("~/documents/Github/ProgrammingSpring2023")
```

```
getwd()
```

```
## [1] "/Users/erinrossiter/Documents/GitHub/ProgrammingSpring2023"
```

```
list.files()
```

```
##  [1] "Day01-Intro"                  "Day02-DataStructures"
##  [3] "Day03-Loops-Functions"        "Day04-Webscraping"
##  [5] "Day05-Webscraping2"           "FinalProjectInfo"
##  [7] "msc_files"                    "Prepare for Day01 - Email.pdf"
##  [9] "PS01"                         "PS02"
## [11] "PS03"                         "PS04"
## [13] "Syllabus.pdf"
```

1b

Since your working directory was set to point inside the course folder in 1a, using `setwd()` again will navigate relative to that location. To demonstrate this, set your working directory to the `PS01` folder.

Demonstrate this code works using `getwd()` and `list.files()` as in 1a.

Answer:

```
setwd("PS01")
getwd()
list.files()
```

1c

Again, we've changed the working directory in 1b. We're pointing inside the "PS01" folder now. To move *back outside* this folder, we use `..` (two periods). Practice moving back to the "ProgrammingSpring2023" folder.

Demonstrate this code works as expected using `getwd()` and `list.files()`.

Answer:

```
setwd("..")
getwd()
list.files()
```

1d

Like the `list.files()` function, there is a `list.dirs()` function that, as you can guess, lists all the *directories* (i.e., folders) in the current working directory.

Use this function, with the argument `full.names` set to `FALSE` to print the names of all subfolders in our course folder. You'll notice there's a lot of *hidden* folders, too. This is how Github does its magic.

Answer:

```
all_folders <- list.dirs(full.names = F)
all_folders ## lots of junk
```

1e

Now, we are going to put all the skills together.

Write a for loop that navigates into each of the folders holding class meeting materials ("Day01-XX", "Day02-XX", etc.) and prints the names of all the files in the folder. This code should be flexible, meaning it will work even when I add materials in the future (Day06, Day07, etc.).

Note there are *many* ways to execute this task. Question 1 has walked you through the skills necessary to execute the task one way, but you may have a different preferred method. Feel free to take any route to the answer as long as it is clean and efficient.

Answer:

```
all_folders <- list.dirs(, full.names = F)
day_folders <- all_folders[grepl(pattern = "Day", x = all_folders)]
day_folders

for(folder_name in day_folders){
  # navigate *into* the folder
  setwd(folder_name)
  # nicely print the contents
  cat("\n*", folder_name, "*\n")
  cat(list.files(), sep = "\n")
  #navigate out
  setwd("..")
}
```

Question 2–JSON

This question practices "functionalizing" your code, meaning turning repetitive tasks into functions. It also practices using JSON objects in R and working with errors. Please complete this question in the .Rmd file.

2a

Write a function called `scrape_state` that scrapes the county-level information for 2018 senate races like in class. Recall data is stored on this website: https://www.cnn.com/election/2018/results/senate. The function should:

- scrapes the election returns for a single state given the state's abbreviate ("IN", "IA", etc.) as the only argument to the function
- return the county-level results as a data.frame, like we practiced in class
- include an error if the user of the function provides a non-existent state code

Demonstrate the function works as intended by doing the following things:

- call the function for Indiana
- print the first few rows of the Indiana dataset
- call the function for a non-existent code to generate your custom error

Hint 1: remember R has all state abbreviations stored in a vector called `state.abbr`. Hint 2: Notice I've given the `r` chunk an extra argument (`error=TRUE`), which allows you to demonstrate an error without killing the knitting of the document.

```
# scrape_state <- function(...){
#
# }
#
```

```r
# # will generate error
# scrape_state("Indiana")
#
# # correct use of function
# indiana_df <- scrape_state("IN")
# head(indiana_df)
```

Answer:

```r
library(jsonlite)
scrape_state <- function(state_abbr){
  if(!state_abbr %in% state.abb){
    stop("Must provide valid state abbreviation.")
  }
  base_url <- "https://data.cnn.com/ELECTION/2018November6/"
  state_json_url <- paste0(base_url, state_abbr, "/county/S.json")
  state_json <- jsonlite::fromJSON(state_json_url, flatten = T)
  return(state_json$counties)
}

# will generate error
scrape_state("Indiana")
```

```
## Error in scrape_state("Indiana"): Must provide valid state abbreviation.
```

```r
# correct use of function
indiana_df <- scrape_state("IN")
head(indiana_df)
```

```
##   co_id        name countycode      race.ts race.pctsrep race.sw race.ip
## 1 18001       Adams      18001 1.545407e+12          100    TRUE       D
## 2 18003       Allen      18003 1.545407e+12          100    TRUE       D
## 3 18005 Bartholomew      18005 1.545407e+12          100    TRUE       D
## 4 18007      Benton      18007 1.545407e+12          100    TRUE       D
## 5 18009   Blackford      18009 1.545407e+12          100    TRUE       D
## 6 18011       Boone      18011 1.545407e+12          100    TRUE       D
##
## 1        19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE, FA
## 2 19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE, FALSE, ma
## 3 19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE, FALSE, ma
## 4           19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE
## 5        19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE, FA
## 6 19233, 50000, 65428, Mike, Joe, Lucy, , , , Braun, Donnelly, Brenton, , , , FALSE, FALSE, FALSE, ma
```

2b

Now write a for loop that calls your function for all 50 states. Because not every state had a senate race in 2018, use the `try()` function to essentially skip over those states. Note we will not "catch" and handle the errors or warnings in any way with `tryCatch()`.

The for loop should somehow store all the results in the same data.frame. To execute this, I recommend appending each iteration's data.frame to the same, main dataframe using the rbind.data.frame() function. I've kindly outlined the structure of the for loop below.

Print the dimensions of the resulting full dataset.

```
# full_data <- data.frame()
# for(s in state.abb){
#   # I recommend implementing this function
#   # to store results
#   rbind.data.frame()
# }
```

Answer:

```
full_data <- data.frame()
for(s in state.abb){
  print(s)
  try({
    s_df <- scrape_state(s)
    full_data <- rbind.data.frame(full_data, s_df)
  })
}
```

```
## [1] "AL"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/AL/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/AL/county/S.json'
## [1] "AK"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/AK/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/AK/county/S.json'
## [1] "AZ"
## [1] "AR"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/AR/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/AR/county/S.json'
## [1] "CA"
## [1] "CO"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/CO/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/CO/county/S.json'
## [1] "CT"
## [1] "DE"
## [1] "FL"
## [1] "GA"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/GA/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/GA/county/S.json'
## [1] "HI"
## [1] "ID"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/ID/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/ID/county/S.json'
## [1] "IL"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/IL/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/IL/county/S.json'
## [1] "IN"
## [1] "IA"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/IA/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/IA/county/S.json'
## [1] "KS"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/KS/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/KS/county/S.json'
## [1] "KY"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/KY/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/KY/county/S.json'
## [1] "LA"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/LA/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/LA/county/S.json'
## [1] "ME"
## [1] "MD"
## [1] "MA"
## [1] "MI"
## [1] "MN"
## [1] "MS"
## [1] "MO"
## [1] "MT"
## [1] "NE"
## [1] "NV"
## [1] "NH"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/NH/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/NH/county/S.json'
## [1] "NJ"
## [1] "NM"
## [1] "NY"
## [1] "NC"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/NC/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/NC/county/S.json'
## [1] "ND"
## [1] "OH"
## [1] "OK"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/OK/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/OK/county/S.json'
## [1] "OR"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/OR/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/OR/county/S.json'
## [1] "PA"
## [1] "RI"
## [1] "SC"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/SC/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/SC/county/S.json'
## [1] "SD"

## Warning in open.connection(con, "rb"): cannot open URL 'https://data.cnn.com/
## ELECTION/2018November6/SD/county/S.json': HTTP status was '503 Service
## Unavailable'

## Error in open.connection(con, "rb") :
##   cannot open the connection to 'https://data.cnn.com/ELECTION/2018November6/SD/county/S.json'
## [1] "TN"
## [1] "TX"
## [1] "UT"
## [1] "VT"
## [1] "VA"
## [1] "WA"
## [1] "WV"
## [1] "WI"
## [1] "WY"
```

```
dim(full_data)
```

```
## [1] 3123    8
```