Problem Set 2

Due February 13, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered within an R script. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded.
2. Work on git. Fork the repository found at https://github.com/domlockett/PDS-PS2 and add your code, committing and pushing frequently. Use meaningful commit messages – these may affect your grade.
3. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. If you have any questions regarding the Problem Set, contact the TAs or use their office hours.
5. For students new to programming, this may take a while. Get started.

for loops, if else, while

1. Write a for loop that iterates over the numbers 1 to 7 and prints the cube of each number using print().
2. Write a for loop that does 1000 simulations of where two fair dice are rolled. Use the function `set.seed(14)` so that we all have the same values when using the sample() function.
   - Write the loop such that if the two dice total to values 8,9,10,11,12 the game ends immediately
   - If the first roll does not equal one of those five values continue to roll the dice until you roll either a 2 or a 6
   - What is the average number of dice casts per game
3. Run the code below. The `game1` object includes the results of five different rounds among 2 players. Write a for loop which returns "Win!" if Player 1 wins the game and returns "Lose :(" if Player two wins.

```
game1 <- list("Game 1" =cbind(3    ,2 ),"Game 2" = cbind(1,2),
"Game 3" =cbind(8,4), "Game 4"= cbind(2, 1), "Game 5" = cbind(4, 6))
colname <- c("Player 1", "Player 2")
for (i in seq_along(game1)){colnames(game1[[i]]) <- colname}
```

   - Now, run this new code to create `game2` which has 6 rounds and add to your for loop a function that returns "Draw!" if player 1 and player 2 have a tie and also include an argument that returns the statement "Warning, there were not enough values in this game" if there is an NA in the either players' values.

```
game2 <- list("Game 1" =cbind(3    ,3 ),"Game 2" = cbind(NA,2), "Game 3"
=cbind(8,4), "Game 4"= cbind(2, NA), "Game 5" = cbind(4, 4), "Game 6" = cbind(3, 4))
colname <- c("Player 1", "Player 2")
for (i in seq_along(game2)){colnames(game2[[i]]) <- colname}
```

Functions

4. Load the following data: http://politicaldatascience.com/PDS/Problem%20Sets/Problem%20Set%202/GSS-data.csv.

Now create a function called `vote.choice` which can take one of three arguments: "Trump", "Clinton", or "Other". The function should return the number of participants who voted for Trump when you input "Trump" into the function; the number of participants who voted for Clinton when you input "Clinton" into the function; and the number of participants that voted for neither when you input "Other".

- Now edit this function so that if a pre-defined object, numeric value or misspelled word is entered, the function returns the message "Please enter either 'Trump' 'Clinton' or 'Other' into the function to return a valid response".

5. Run the following code:

```r
install.packages('fivethrityeight')
library(fivethirtyeight)
```

Now review the data in the `cabinet_turnover` object (this is loaded into your space when you load the library even though you cannot see it in the global space. You can also assign it to your own object if you'd like.).

Create a function named `appoint` which allows you to type in the name of a president as an argument (i.e appoint("Trump")) and returns the proportion of time appointees spent serving each adminstration i.e the number of days appointees served for each administration, on average, divided by the number of days the particular president served.

To illustrate the average number of days all appointees served in the Reagan administration was 2140.959. Below you can see that Reagan served 2922 days. So appointees served 73% of Reagan's administration, on average (2140.959/2922).

For simplicity, here are the number of days each president served:

Carter: 1461

Reagan: 2922

Bush 41: 1461

Clinton: 2922

Bush 43: 2922

Obama: 2922

Trump: 1105[1]

6. Now you will use the `congress_age` data set from the `fivethirtyfive` package. Create a function called `congress_stats` that takes two arguments "congress" and "state".

When you enter "congress" into the function it should return the average age of congressmembers for each congressional era. Your function should return 34 results which display the average age of congressmembers of an era as well as the congress. For example the most recent congress is the 113 Congress so one of the 34 results will be `57.6 113`.

Similarly, when you input "state" into the function, it should return the average age of congressmembers by state. The function will then return 50 results an example of one of the 50 is `53.4 TX`.

[1]As of January 30, 2020