11.5/12

Problem Set 2

Political Data Science - Spring 2020

Gencer, Alper Sukru

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.

Q1 - 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

A1 -for loops, if else, while

```
rm(list = ls())
```

Good

Write a for loop that iterates over the numbers 1 to 7 and prints the cube of each number using print().

```
for (i in 1:7){
  print(i^3)
}
```

```
## [1] 1
## [1] 8
## [1] 27
## [1] 64
## [1] 125
## [1] 216
## [1] 343
```

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?

```
set.seed(14)
for (i in 1:1000){
  how.I.roll <-sample(c(1:6), 2, replace=T)
  print(how.I.roll)
  if(i == 1 & sum(how.I.roll) >= 8){
    break
  } else if(how.I.roll[1] == 2 | how.I.roll[2] == 6) {
    print(paste("Average number of dice casts is", i))
    break
  }
}
```

```
## [1] 1 1
## [1] 3 4
## [1] 3 6
## [1] "Average number of dice casts is 3"
```

-1/2: I think you misunderstood the portion of the question regarding averages. I wanted the average number of rolls for the 1000 simulations. The question wording was ambiguous so I won't take the point. That said, there was clarification in the problem-sets slack

Q2 - PART1: Functions

Load the following data:

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

1. 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".
2. 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".

A2 - PART1: - Functions

Load the following data:

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

```
rm(list = ls())

#   Let's import the dataset:
library(readr)
GSS_data <- read_csv("http://politicaldatascience.com/PDS/Problem%20Sets/Problem%20Set%202/GSS-data.csv
```

Good.

1. 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".
2. 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".

```
#   Before we start working on our function, we should clean the dataset:
library("tidyverse")

number.voters <- GSS_data %>%
  mutate(pres16 = as.factor(pres16)) %>%
  group_by(pres16) %>%
  tally()

levels(GSS_data$pres16)
```

```
## NULL
```

```
n.voter <- c(number.voters$n[1], number.voters$n[6], number.voters$n[7])
names(n.voter) <- c("Clinton", "Other", "Trump")
n.voter
```

```
## Clinton   Other   Trump
##     764      87     577
```

```
#   Let's generate that handsome function:
vote.choice <- function(x){
  if(x == "Trump"){
    return(paste("The number of participants who voted for", x, "is", n.voter[x]))
  } else if(x == "Clinton"){
    return(paste("The number of participants who voted for", x, "is", n.voter[x]))
  } else if(x == "Other"){
    return(paste("The number of participants who voted for", x, "is", n.voter[x]))
  } else {
    stop("Please enter either 'Trump' 'Clinton' or 'Other' into the function!")
  }
}

#   Now, let's see if it works or not!
vote.choice("Trump")
```

```
## [1] "The number of participants who voted for Trump is 577"
```

```r
vote.choice("Clinton")
```

```
## [1] "The number of participants who voted for Clinton is 764"
```

```r
vote.choice("Other")
```

```
## [1] "The number of participants who voted for Other is 87"
```

```r
#   What about the error message?
vote.choice("Sanders")
```

```
## Error in vote.choice("Sanders"): Please enter either 'Trump' 'Clinton' or 'Other' into the function!
```

```r
#   YES, it workssss!
```

Q2 - PART2: Functions

Run the following code

```
# 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.).

1. 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.
2. 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
3. For simplicity, here are the number of days each president served:

$$Carter : 1461 Reagan : 2922 Bush\ 41 : 1461 Clinton : 2922 Bush\ 43 : 2922 Obama : 2922 Trump : 11051$$

A2 - PART2: Functions

Run the following code

Good.

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.).

```
#install.packages('fivethrityeight')
rm(list = ls())
library(fivethirtyeight)

#   Let's import the data:
data("cabinet_turnover")
str(cabinet_turnover)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 312 obs. of  7 variables:
##  $ president: Factor w/ 7 levels "Bush 41","Bush 43",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ position : Factor w/ 28 levels "Attorney General",..: 8 24 16 19 23 1 26 14 27 11 ...
##  $ appointee: chr  "Bert Lance" "Brock Adams" "Joseph Califano Jr." "Patricia Harris" ...
##  $ start    : Date, format: "1977-01-21" "1977-01-23" ...
##  $ end      : Date, format: "1977-09-23" "1979-07-20" ...
##  $ length   : num  245 908 920 922 923 ...
##  $ days     : num  247 912 926 926 927 ...
```

1. 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.
2. 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
3. For simplicity, here are the number of days each president served:

$$Carter : 1461 Reagan : 2922 Bush\ 41 : 1461 Clinton : 2922 Bush\ 43 : 2922 Obama : 2922 Trump : 1105$$

```
library("tidyverse")
#detach(package:plyr)


my.cab.to <- cabinet_turnover %>%
  group_by(president) %>%
  summarize(mean.length = mean(length, na.rm = TRUE))

# Let's add the number of serving days as a vector:
my.cab.to
```

```
## # A tibble: 7 x 2
##   president mean.length
##   <fct>           <dbl>
## 1 Bush 41          897.
## 2 Bush 43         1135.
## 3 Carter           849.
## 4 Clinton         1236.
## 5 Obama           1252.
## 6 Reagan          1204.
```

```
## 7 Trump            470
```

```
serving.dur <- c(1461, 2922, 1461, 2922, 2922, 2922, 1105)
my.cab.to <- cbind(my.cab.to, serving.dur)

my.cab.to <- my.cab.to %>%
  mutate(prop.appoint.time = round(mean.length/serving.dur, digit = 4))

appoint <- function(x){
  return(paste("The proportion of time appointees spent serving", x, "administration is", my.cab.to$prop
}
```

```
#   Let's see if it works or not!
appoint("Trump")
```

```
## [1] "The proportion of time appointees spent serving Trump administration is 0.4253"
```

```
appoint("Bush 41")
```

```
## [1] "The proportion of time appointees spent serving Bush 41 administration is 0.6138"
```

```
appoint("Sanders")
```

```
## [1] "The proportion of time appointees spent serving Sanders administration is "
```

Q3 - PART2: Functions

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.

Good.

A3 - PART3: Functions

```
#install.packages('fivethrityeight')
rm(list = ls())
library(fivethirtyeight)
```

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.

In the beginning of the question, question clearly states that the function would take "two" arguments. However, in the rest of the question, it is clear that the question asks for a function with one argument that could take two values. Therefore, I have added both versions of the aforementioned questions.

Let's see the first case where the function takes only one argument with two meaningful input:

```
require("tidyverse")
library("dplyr")

congress_stats <- function(X){
  if (X == "congress"){
    m.age.cong <- congress_age %>%
      group_by(congress) %>%
      summarize(m.age = mean(age, na.rm=TRUE))
    print(m.age.cong)
  } else if(X == "state") {
    m.age.stat <- congress_age %>%
      group_by(state) %>%
```

```
      summarize(m.age = mean(age, na.rm=TRUE))
    print(m.age.stat)
  } else {
    print("Invalid Input! Please enter either 'congress' or 'state'.")
  }
}
```

Let's try our function:

```
congress_stats("congress")
```

```
## # A tibble: 34 x 2
##    congress m.age
##       <int> <dbl>
##  1       80  52.5
##  2       81  52.6
##  3       82  53.2
##  4       83  53.2
##  5       84  53.4
##  6       85  54.2
##  7       86  53.2
##  8       87  53.6
##  9       88  53.0
## 10       89  52.2
## # ... with 24 more rows
```

```
congress_stats("state")
```

```
## # A tibble: 50 x 2
##    state m.age
##    <chr> <dbl>
##  1 AK     57.6
##  2 AL     53.0
##  3 AR     52.8
##  4 AZ     54.7
##  5 CA     53.6
##  6 CO     52.3
##  7 CT     49.7
##  8 DE     53.1
##  9 FL     53.4
## 10 GA     53.4
## # ... with 40 more rows
```

```
congress_stats("gibrish")
```

```
## [1] "Invalid Input! Please enter either 'congress' or 'state'."
```

```
congress_stats(TRUE)
```

```
## [1] "Invalid Input! Please enter either 'congress' or 'state'."
```

```
congress_stats(100)
```

```
## [1] "Invalid Input! Please enter either 'congress' or 'state'."
```

Awesome!

Now, let's reconfigurate the same function to support two arguments:

```
congress_stats.v2 <- function(X = ".", Y="."){
  if (X == "congress"){
    m.age.cong <- congress_age %>%
      group_by(congress) %>%
      summarize(m.age = mean(age, na.rm=TRUE))
    print(m.age.cong)
    if (Y == "state") {
    m.age.stat <- congress_age %>%
      group_by(state) %>%
      summarize(m.age = mean(age, na.rm=TRUE))
    print(m.age.stat)
    }
  } else {
    if (Y == "state") {
    m.age.stat <- congress_age %>%
      group_by(state) %>%
      summarize(m.age = mean(age, na.rm=TRUE))
    print(m.age.stat)
    } else {
      print(paste("Invalid Input! Please enter either"))
      print(paste("'congress' as the first argument or "))
      print(paste("'state' as your second argument, or"))
      print(paste("both."))
    }
  }
}
```

Let's try our function:

```
congress_stats.v2("congress")
```

```
## # A tibble: 34 x 2
##    congress m.age
##       <int> <dbl>
##  1       80  52.5
##  2       81  52.6
##  3       82  53.2
##  4       83  53.2
##  5       84  53.4
##  6       85  54.2
##  7       86  53.2
##  8       87  53.6
##  9       88  53.0
## 10       89  52.2
## # ... with 24 more rows
```

```
congress_stats.v2(, "state")
```

```
## # A tibble: 50 x 2
##    state m.age
##    <chr> <dbl>
##  1 AK     57.6
```

```
##  2 AL     53.0
##  3 AR     52.8
##  4 AZ     54.7
##  5 CA     53.6
##  6 CO     52.3
##  7 CT     49.7
##  8 DE     53.1
##  9 FL     53.4
## 10 GA     53.4
## # ... with 40 more rows
```

```
congress_stats.v2("congress", "state")
```

```
## # A tibble: 34 x 2
##    congress m.age
##       <int> <dbl>
##  1       80  52.5
##  2       81  52.6
##  3       82  53.2
##  4       83  53.2
##  5       84  53.4
##  6       85  54.2
##  7       86  53.2
##  8       87  53.6
##  9       88  53.0
## 10       89  52.2
## # ... with 24 more rows
## # A tibble: 50 x 2
##    state m.age
##    <chr> <dbl>
##  1 AK     57.6
##  2 AL     53.0
##  3 AR     52.8
##  4 AZ     54.7
##  5 CA     53.6
##  6 CO     52.3
##  7 CT     49.7
##  8 DE     53.1
##  9 FL     53.4
## 10 GA     53.4
## # ... with 40 more rows
```

```
congress_stats.v2("gibrish")
```

```
## [1] "Invalid Input! Please enter either"
## [1] "'congress' as the first argument or "
## [1] "'state' as your second argument, or"
## [1] "both."
```

```
congress_stats.v2(TRUE)
```

```
## [1] "Invalid Input! Please enter either"
## [1] "'congress' as the first argument or "
## [1] "'state' as your second argument, or"
## [1] "both."
```

```
congress_stats.v2(100)
```

```
## [1] "Invalid Input! Please enter either"
## [1] "'congress' as the first argument or "
## [1] "'state' as your second argument, or"
## [1] "both."
```

Awesome!