Intro to Data Science HW 3


```
# Enter your name here: Bhavya Shah
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

Reminders of things to practice from last week:

Make a data frame: data.frame( )
Row index of max/min: which.max( ) which.min( )
Sort value or order rows: arrange( ) sort( ) order( )
Descriptive statistics: mean( ) sum( ) max( )
Conditional statement: if (condition) true stuff else false stuff

This Week:

Often, when you get a dataset, it is not in the format you want. You can (and should) use code to refine the dataset to become more useful. As Chapter 6 of Introduction to Data Science mentions, this is called ** data munging. ** In this homework, you will read in a dataset from the web and work on it (in a data frame) to improve its usefulness.

Part 1: Use read_csv( ) to read a CSV file from the web into a data frame:

A. Use R code to read directly from a URL on the web. Store the dataset into a new dataframe, called **testDF**. The URL is:
“”

Hint: use read_csv( ), not read.csv( ). This is from the **tidyverse package**. Check the help to compare them.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.1
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.3.0      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
testDF<-data.frame(read.csv("https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing.csv"))
```

B. Use View( ), head( ), and tail( ) to examine the **testDF** dataframe.
Add a block comment that briefly describes what you see.

```
#view(testDF)
#head(testDF)
tail(testDF)
```

```
##      TestDate AgeGroup PositiveCases TotalTests     AgeCategory
## 7378 1/3/2022  5 to 19          9923      38977        children
## 7379 1/3/2022 55 to 64          5739      27019 senior_citizens
## 7380 1/3/2022 65 to 74          2759      14498 senior_citizens
## 7381 1/3/2022 75 to 84          1141       6519 senior_citizens
## 7382 1/3/2022     85 +           680       4028 senior_citizens
## 7383 1/3/2022      < 1           717       2074        children
```

```
# The view function displays the entire data in a table format. The head function gives the first 6 rows of the data frame while the tail function gives the last 6 rows.
```

Part 2: Create new data frames based on a condition:

A. Use the table( ) command to summarize the contents of the **AgeCategory** variable in **testDF**.
Write a comment interpreting what you see how many age categories are there in the dataset and what is the proportion of observations in each?

```
table(testDF$AgeCategory)
```

```
##
##           children middle-aged_adults    senior_citizens       young_adults
##               2010               1345               2686               1342
```

```
# There are 4 categories in the dataset. The table command gives the values of each category and they are in the proportion as children:middle-aged_adults :senior_citizens:young_adults = 2010:1345:2686:1342
```

B. Terms like “senior citizens” can function as *othering* language which demeans the people it seeks to describe. We can use the **str_replace_all()** function from tidyverse to find all instances of **senior_citizens** in the **AgeCategory** variable and replace them with **older_adults**.

In this case, we want to search for **senior_citizens** and replace it with **older_adults** in **testDF$AgeCategory** - how can you use this information to overwrite the **AgeCategory** in the function below:

```
 testDF$AgeCategory<-str_replace_all(testDF$AgeCategory,"senior_citizens","older_adults")
table(testDF$AgeCategory)
```

```
##
##           children middle-aged_adults       older_adults       young_adults
##               2010               1345               2686               1342
```

C. Create a dataframe (called **olderAdults**) that contains only the rows (observations) for which the value in the **AgeCategory** variable (column) is **older_adults**. Hint: Use subsetting.

```
olderAdults<-testDF[testDF$AgeCategory=="older_adults",]
```

D. Use the dim() command on **olderAdults** to confirm that the data frame contains **2,686** observations and **5** columns/variables.

```
dim(olderAdults)
```

```
## [1] 2686    5
```

E. Use **subsetting** to create a new dataframe that contains only the observations for which the value in the **AgeCategory** variable is **young_adults**. The name of this new df should be **youngAdults**.

F. Create one last data frame which only contains the observations for **children** in the **AgeCategory** variable of **testDF**. Call this new df **childrenDF**.

```
childrenDF<-testDF[testDF$AgeCategory=="children",]
```

Part 3: Analyze the numeric variables in the **testDF** dataframe.

A. How many **numeric variables** does the dataframe have? You can figure that out by looking at the output of **str(testDF)**.

```
str(testDF)
```

```
## 'data.frame':    7383 obs. of  5 variables:
##  $ TestDate     : chr  "3/2/2020" "3/3/2020" "3/3/2020" "3/3/2020" ...
##  $ AgeGroup     : chr  "45 to 54" "25 to 34" "35 to 44" "45 to 54" ...
##  $ PositiveCases: int  1 0 0 0 0 0 0 0 2 ...
##  $ TotalTests   : int  1 2 1 1 2 2 1 1 5 2 ...
##  $ AgeCategory  : chr  "middle-aged_adults" "young_adults" "middle-aged_adults" "middle-aged_
adults" ...
```

```
# The dataframe contains 2 numeric variables.
```

B. What is the average number of total daily tests? Hint: Can you think of a mathematical function we've come across before to use on the **TotalTests** variable?

```
mean(testDF$TotalTests)
```

```
## [1] 11724.96
```

C. How many tests were performed in the row with the highest number of total daily tests? What age category do they correspond to?

```
maxt<-testDF[which.max(testDF$TotalTests),]
maxtests<-select(maxt,TotalTests,AgeCategory)
maxtests
```

```
##      TotalTests  AgeCategory
## 7265      88904 young_adults
```

```
# Highest total tests are 88904 which belong to the young_adults category.
```

D. How many positive cases were registered in the row with the highest number of positive cases? What age category do they correspond to?

```
maxp<-testDF[which.max(testDF$PositiveCases),]
maxpos<-select(maxp,PositiveCases,AgeCategory)
maxpos
```

```
##      PositiveCases  AgeCategory
## 7342         17486 young_adults
```

```
# Highest positive cases are 17486 which belong to the young_adults category.
```

E. What is the total number of positive cases in **testDF**?

```
sum(testDF$PositiveCases)
```

```
## [1] 3722542
```

F. Create a new variable in **testDF** which is the ratio of **PostiveCases** to **TotalTests**. Call this variable **PositivityRate** and explain in a comment what information it gives us.

```
testDF$PositivityRate<-testDF$PositiveCases/testDF$TotalTests
# PositivityRate gives us the rate of positive cases with respect to the total recorded cases.
```

G. What is the average positivity rate in **testDF**? Hint: Use the **mean()** function on the new variable you created in F.

```
mean(testDF$PositivityRate)
```

```
## [1] 0.05363633
```

Part 4: Create a function to automate the process from F-G:

A. The following function should work most of the time. Make sure to run this code before trying to test it. That is how you make the new function known to R. **Add comments to each line explaining what it does:**

```
calculatePositivity <- function(dataset) {
  dataset$PositivityRate <- dataset$PositiveCases/dataset$TotalTests
  avePositivity <- mean(dataset$PositivityRate)
  return(avePositivity)
}
```

B. Run your new function on the **testDF** dataframe. Is the output of the function consistent with the output in Step G above? Explain.

```
calculatePositivity(testDF)
```

```
## [1] 0.05363633
```

C. Run the function on the **olderAdults** df you created earlier.

```
calculatePositivity(olderAdults)
```

```
## [1] 0.05386202
```

D. Run the function on the **youngAdults** df.

```
#calculatePositivity(youngAdults)
#[1] 0.05435627
#while knitting to pdf an error was popping which I was not able to resolve despite the code bei
ng run and the output being displayed. hence, 9ii have commented the code and the output for thi
s step.
```

E. Lastly, run the posivity function on the **childrenDF** dataframe.

```
calculatePositivity(childrenDF)
```

```
## [1] 0.05034919
```

F. In a comment, describe what you observe across these 3 datasets - which age group exhibits the highest positivity rate? How do these numbers compare to the **baseline** positivity rate in **testDF**?

```
# AS per the above observation, I conclude that the age category of young Adults has the most po
sitivity rate than other categories.
```