Jailee Foster
jailee.foster@wsu.edu
WSU ID: 11439720
Stats 419
Instructor: Monte J. Shaffer
September 7, 2020

Assignment: Datasets

1. Create the "rotate matrix" functions as described in lectures. Apply to the example "`myMatrix`".

```
myMatrix = matrix ( c (
                        1, 0, 2,
                        0, 3, 0,
                        4, 0, 5
                        ), nrow=3, byrow=T);

                        transposeMatrix = function(mat)
                                {
                                t(mat);
                                }

                        #rotateMatrix90(mat)
                        #rotateMatrix180(mat)
                        #rotateMatrix270(mat)
                        # 3x3 matrix ... ## matrix multiplication
```

The following is the R code and output for the rotateMatrix90(mat) function that I wrote. I created a transformation matrix that, when a matrix is multiplied by it, the columns of the matrix are reversed. I used both the transformation matrix and the transpose of the matrix in my function.

```
> transformationMatrix = matrix(c(0, 0, 1,
+                                 0, 1, 0,
+                                 1, 0, 0), nrow=3, byrow=T)
> rotateMatrix90 = function(mat)
+ {
+   matrix = t(mat) %*% transformationMatrix;
+   matrix;
+ }
> rotateMatrix90(myMatrix)
     [,1] [,2] [,3]
[1,]    4    0    1
[2,]    0    3    0
[3,]    5    0    2
```

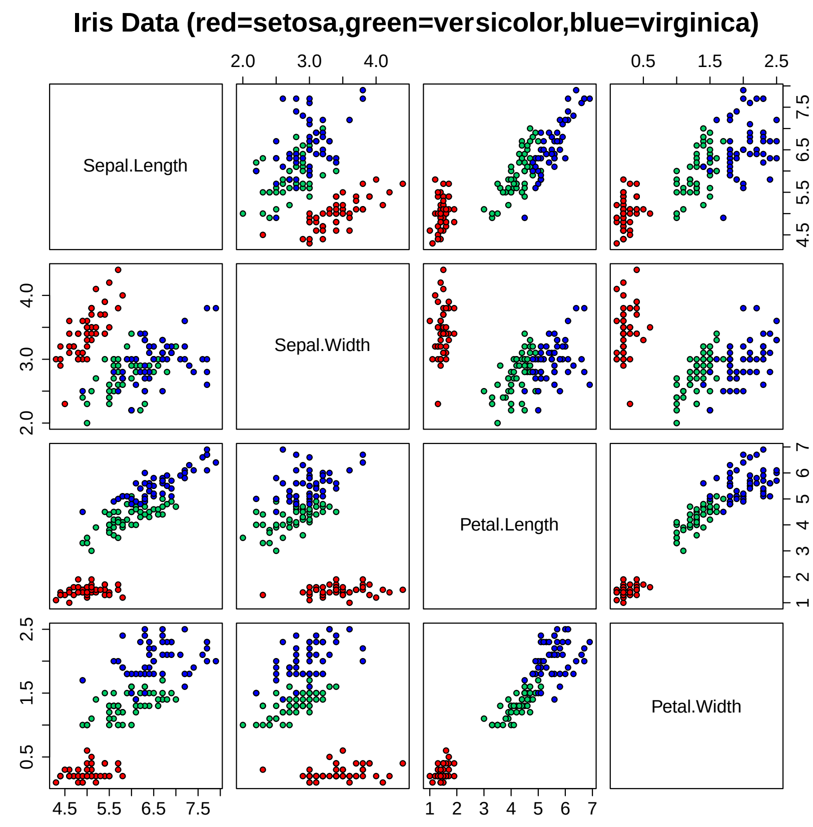
The following is the R code and output for the rotateMatrix180(mat) function that I wrote. For this function, I applied the function created above that rotates a matrix 90 degrees two times.

```
> rotateMatrix180 = function(mat)
+ {
+   matrix = rotateMatrix90(rotateMatrix90(mat));
+   matrix;
+ }
> rotateMatrix180(myMatrix)
     [,1] [,2] [,3]
[1,]    5    0    4
[2,]    0    3    0
[3,]    2    0    1
```

The following is the R code and output for the rotateMatrix180(mat) function that I wrote. I used both the transformation matrix created above and the transpose of the matrix in my function.

```
> rotateMatrix270 = function(mat)
+ {
+   matrix = t(mat %*% transformationMatrix);
+   matrix;
+ }
> rotateMatrix270(myMatrix)
     [,1] [,2] [,3]
[1,]    2    0    5
[2,]    0    3    0
[3,]    1    0    4
```

2. Recreate the graphic for the IRIS Data Set using r. Same titles, same scales, same colors. See: https://en.wikipedia.org/wiki/Iris_flower_data_set#/media/File:Iris_dataset_scatterplot.svg.



The following is the R code that was used to create this plot:

```
> data(iris)

> pairs(iris[, 1:4], main="Iris Data (red=setosa,green=versicolor,
      blue=virginica)", pch=21, col="black", bg=c("red",
      "springgreen3", "blue")[iris$Species], cex.labels=1.5,
      cex.axis=1.5, cex.main=1.5)
```

For this question, I referenced the following websites:

to add a title: https://www.dummies.com/programming/r/how-to-add-titles-and-axis-labels-to-a-plot-in-r/
to change the font sizes: http://www.sthda.com/english/wiki/add-titles-to-a-plot-in-r-software
to change the format of the points: https://www.datanovia.com/en/blog/pch-in-r-best-tips/

I also did a google image search for “colors in R” to find the colors that most closely resemble those in the original plot.

3. Write 2-3 sentences concisely defining the IRIS Data Set. Maybe search KAGGLE for a nice template. Be certain the final writeup are your own sentences (make certain you modify what you find, make it your own, but also cite where you got your ideas from). NOTE: Watch the video, Figure 8 has a +5 EASTER EGG.

The IRIS Data Set provides the following characteristics relating the iris flower: sepal length, sepal width, petal length, and petal width (all in centimeters). The data set also specifies which species each recording is. The data set itself is home to information about 50 samples each of 3 different species of iris flower, 150 samples in total.

The information for question 3 came from https://archive.ics.uci.edu/ml/datasets/iris.

4. Import "personality-raw.txt" into R. Remove the V00 column. Create two new columns from the current column "date_test": year and week. Stack Overflow may help: https://stackoverflow.com/questions/22439540/how-to-get-week-numbers-from-dates ... Sort the new data frame by YEAR, WEEK so the newest tests are first ... The newest tests (e.g., 2020 or 2019) are at the top of the data frame. Then remove duplicates using the unique function based on the column "md5_email". Save the data frame in the same "pipe-delimited format" (| is a pipe) with the headers. You will keep the new data frame as "personality-clean.txt" for future work (you will not upload it at this time). In the homework, for this tasks, report how many records your raw dataset had and how many records your clean dataset has.

The following is the R code used to import the data into R and remove the V00 column.

```
> personality.data = read.csv("personality-raw.txt", header=T, sep="|")
> personality.data = subset(personality.data, select=-c(V00))
```

The following is the R code used to sort the data by date order, so the newest tests are at the top of the data frame and the oldest tests are at the bottom. This code references https://stackoverflow.com/questions/6246159/how-to-sort-a-data-frame-by-date.

```
> personality.data=personality.data[rev(order(as.Date(personality.
data$date_test, format='%m/%d/%Y %H:%M'))),]
```

The following is the R code used to create two columns, “year” and “week”, from the information in the “date_test” column, and then removes the “date_test” column from the dataset. This code references the code that Dr. Shaffer posted in the discussion board.

```
> date = strptime(personality.data$date_test, format='%m/%d/%Y %H:%M');
> year = as.numeric(strftime(date, format="%Y"));
> week = as.numeric(strftime(date, format="%W"));

> personality.data$year = year;
> personality.data$week = week;

> personality.data = subset(personality.data, select=-c(date_test))
```

The following is the R code used to remove the duplicated data based on the column "md_5 email". This code references https://www.datanovia.com/en/lessons/identify-and-remove-duplicate-data-in-r/. This code also creates a vector of unique "md_5 email"'s.

```
> unique.by.email = unique(personality.data["md5_email"])
> personality.data.clean =
personality.data[!duplicated(personality.data["md5_email"]),]
```

The following is the output produced by taking the dimension of the raw dataset, the clean dataset, and the vector of unique emails. Here we see that the raw dataset had 838 records and the clean dataset has 678 records. It is important to note that the number of unique emails found by the unique() command is also 678, implying that the duplicated() command was used properly to remove duplicate email addresses above.

```
> dim(personality.data)
[1] 838  63
> dim(unique.by.email)
[1] 678   1
> dim(personality.data.clean)
[1] 678  63
```

The following is the R code used to export the clean data frame to a csv file delimited by pipes. This new file is called "personality-clean.txt". This code references https://www.rdocumentation.org/packages/utils/versions/3.6.2/ topics/write.table

```
> write.table(personality.data.clean, "personality-clean.txt", sep="|")
```

5. Write functions for doSummary and sampleVariance and doMode ... test these functions in your homework on the "monte.shaffer@gmail.com" record from the clean dataset. Report your findings. For this "monte.shaffer@gmail.com" record, also create z-scores. Plot(x,y) where x is the raw scores for "monte.shaffer@gmail.com" and y is the z-scores from those raw scores. Include the plot in your assignment, and write 2 sentences describing what pattern you are seeing and why this pattern is present.

```
doSummary = function(x) # x is a vector of data
{
  length = length(x);
  na.count = sum(is.na(x));
  # na.rm=T does not include NAs in the calculation for mean and median
  # this allows for a numeric answer, rather than NA.
  mean = mean(x, na.rm=T)
  median = median(x, na.rm=T);
  mode = doMode(x)
  variance = sampleVariance(x, "naive");
  sd.builtin = sd(x, na.rm=T);
  sd.custom = sqrt(variance);
  data.frame(length=length, na.count=na.count, mean=mean, median=mean,
mode=mode, var=variance, sd.builtin=sd.builtin, sd.custom=sd.custom);
}
```

```
doMode = function(x)
{
  freq = table(x);
  freq.df = as.data.frame(freq);
  result = as.numeric(as.vector(freq.df[freq.df$Freq ==
max(freq.df$Freq),]$x))
  result;
}


sampleVariance = function(x, method)
{
  x=x[!is.na(x)];
  if(method=="naive")
  {
    n=0;
    sum=0;
    sum.squared=0;
    for (i in 1:length(x))
    {
      n=n+1;
      sum=sum+x[i];
      sum.squared=sum.squared+(x[i]*x[i]);
    }
    naive.variance=(sum.squared-(sum^2)/n)/(n-1);
    data.frame(sum=sum, sum.squared=sum.squared, var=naive.variance);
  }
  else
  {
    sum1=0;
    sum2=0;
    n=0;
    for (i in 1:length(x))
    {
      n=n+1;
      sum1=sum1+x[i];
    }
    mean=sum1/n;
    for (i in length(x))
    {
      sum2=sum2+((x[i]-mean)^2);
    }
    variance=sum2/(n-1);
    data.frame(sum=sum1, sum2=sum2, var=variance);
  }
}
```

I entered monte.shaffer@gmail.com into an md5 generator and was found that it converts to the hash b62c73cdaf59e0a13de495b84030734e. This correlates to the test result from week 14 in 2020 in the clean personality dataset. To access this row and create a new vector holding only the numeric values for columns V01 to V60, I used the following code.

```
> personality.vec = as.vector(personality.data.clean[1,])
> personality.vec = as.numeric(subset(personality.vec, select=-
c(md5_email, year, week)))
```

The following is the ouput when the personality.vec vector is a parameter to the doSummary(), doMode() and sampleVariance() methods written above.

```
> doSummary(personality.vec)
  length na.count mean median mode var.sum var.sum.squared   var.var
1     60        0 3.48   3.48  4.2   208.8          771.04 0.7528136
  sd.builtin sd.custom.sum sd.custom.sum.squared sd.custom.var
1  0.8676483      14.44991              27.76761     0.8676483

> doMode(personality.vec)
[1] 4.2

> sampleVariance(personality.vec, "naive")
    sum sum.squared       var
1 208.8      771.04 0.7528136
> sampleVariance(personality.vec, "na")
    sum   sum2         var
1 208.8 0.5184 0.008786441
```

Based on the results from performing the above functions on the "monte.shaffer@gmail.com" record from the clean dataset, the following conclusions can be drawn:

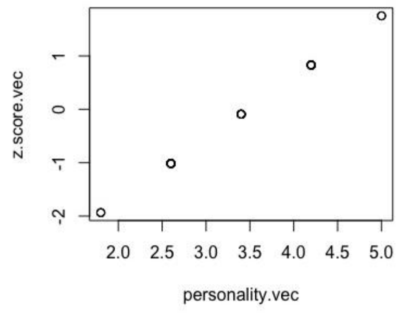
- there are 60 entries, which corresponds to the columns V01 to V60
- there are not any NA values in this data, which is expected because it is clean data
- the mean and median test results are both 3.48, this suggests that the distribution of scores is symmetrical. This also suggests that the individual taking the test answer does not possess much bias, because 3.48 is almost exactly in the middle.
- the mode tells us that the most common answer was 4.2
- the naïve variance was significantly larger than the two-pass variance, suggesting that there may have been a small amount of "outlier" answers that are significantly higher or lower than the other answers.

The following is the R code used to find the z-scores to normalize the raw data scores and then transform those scores into a vector that can be used to plot.

```
> z.scores = scale(personality.vec, center=T, scale=T)
> z.score.vec = as.vector(z.scores)
```

The following is the R code used to find the plot the raw scores (x) against the z-scores (y). Looking at the plot, I noticed that there were only 5 data points, when I thought that there should be more. Upon further inspection, it seemed that many of the scores were repeated throughtout each of the vectors.

```
> plot(personality.vec, z.score.vec)
```



To get a better idea of what was going on, I applied the unique() command to each vector and found that the personality vector only had 5 unique values, which corresponded to the 5 unique z-scores for those values. The points fall in a linear fashion, which makes sense because the same formula/ratio is used to find each z-score, there are essentially just different scalers.

```
> unique(z.score.vec)
[1] -0.09220326  0.82982933 -1.01423585  1.75186192 -1.93626843
> unique(personality.vec)
[1] 3.4 4.2 2.6 5.0 1.8
```

6. Compare Will Smith and Denzel Washington. [See 03_n greater 1-v2.txt for the necessary functions and will-vs-denzel.txt for some sample code and in DROPBOX: __student_access__\unit_01_exploratory_data_analysis\week_02\imdb-example] You will have to create a new variable $millions.2000 that converts each movie's $millions based on the $year of the movie, so all dollars are in the same time frame. You will need inflation data from about 1980-2020 to make this work.

The code for questions 6 and 7 piggybacks off of the code that Dr. Shaffer provided in the “imdb-example” folder in Dropbox. I have only included the code that I either modified or wrote from scratch.

The following is the R code that calculates the inflation rate as a decimal that goes out further than the inflation rate that was provided in the inflation table. This allows for more accurate calculations in the future. The new column full of inflation decimal numbers is added to the inflation table as the column “inflation.decimal”.

```
> inflation.decimal=c(0.1561)
> for (i in 2:101)
+ {
+   inflation.decimal[i]=result$dollar[i]/result$dollar[i-1] - 1
+ }
> result$inflation.decimal = inflation.decimal
```

The following is the R code that restructures the inflation table so that it is in $1,000,000 in the year 2000. I broke the problem into 2 for loops depending on if the year fell before or after 2000. The new column of values in relation to the value of $1,000,000 in 2000 is added to the inflation table as the column “dollars.2000”.

```
> dollars.2000=c()
> dollars.2000[81] = 1000000
>
> for (i in 82:101)
+ {
+   dollars.2000[i] = dollars.2000[i-1]*
                      (1+result$inflation.decimal[i])
+ }
>
> for (i in 1:80)
+ {
+   dollars.2000[81-i] = dollars.2000[82-i]/
                         (1+result$inflation.decimal[82-i])
+ }
>
> result$dollars.2000 = dollars.2000
```

In order to check the results of the work performed on the inflation table, the command head() was used to view the first 6 rows.

```
> head(result)
  year  dollar inflation inflation.decimal dollars.2000
1 1920 1000000     15.61        0.15610000    116144.02
2 1921  895000    -10.50       -0.10500000    103948.90
3 1922  840000     -6.15       -0.06145251     97560.98
4 1923  855000      1.79        0.01785714     99303.14
5 1924  855000      0.00        0.00000000     99303.14
6 1925  875000      2.34        0.02339181    101626.02
```

The following is the R code that converts the “movies.50.millions” column into values that are all the same scale. The scale used is what $1,000,000 was worth in the year 2000. I first created data frames from the information from grabFilmsForPerson() for each of the actors. This may have not been the most efficient way to do this, but it still worked. I then iterated through each value in the “movies.50.millions” column for each actor and expressed those numbers in dollars in relation to what $1,000,000 was worth in 2000. Those values then got added to different vectors for each actor and added as new columns on the data frames called “will.millions.2000” and “denzel.millions.2000”.

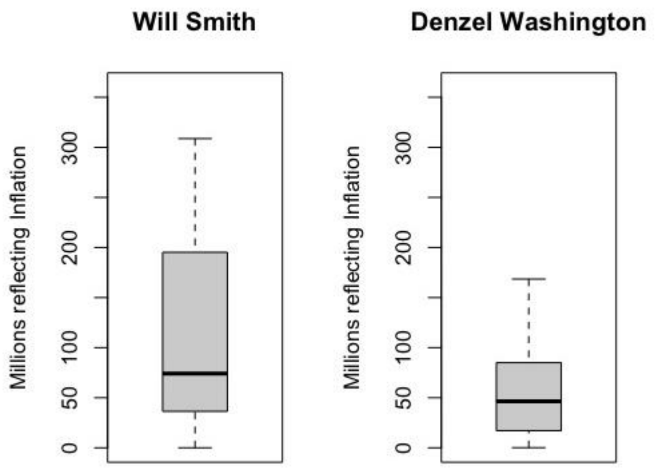
```
> will.df = as.data.frame(will)
> denzel.df = as.data.frame(denzel)
>
> will.millions.2000 = c()
>
> for (i in 1:50)
+ {
+   line = as.numeric(will.df$movies.50.year) - 1919
+   will.millions.2000[i] = will.df$movies.50.millions[i] *
                           (result$dollars.2000[line[i]])/1000000
+ }
```

```
> will.df$millions.2000 = will.millions.2000
> will$millions.2000=will.millions.2000
>
> denzel.millions.2000 = c()
>
> for (i in 1:50)
+ {
+   line = as.numeric(denzel.df$movies.50.year) - 1919
+   denzel.millions.2000[i] = denzel.df$movies.50.millions[i] *
                          (result$dollars.2000[line[i]])/1000000
+ }
>
> denzel.df$millions.2000 = denzel.millions.2000
> denzel$millions.2000=denzel.millions.2000
```

7. Build side-by-side box plots on several of the variables (including #6) to compare the two movie stars. After each box plot, write 2+ sentence describing what you are seeing, and what conclusions you can logically make. You will need to review what the box plot is showing with the box portion, the divider in the box, and the whiskers.

The first box plots that I built compare the dollars brought in by each of the actor with regard to inflation. Below is the R code and the plots.

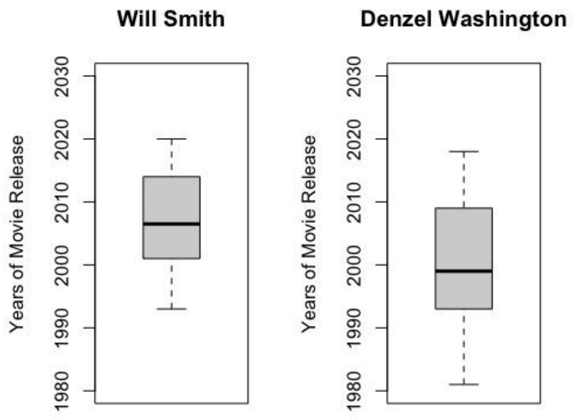
```
> par(mfrow=c(1,2));
> boxplot(will$millions.2000, main=will$name, ylim=c(0,360),
      ylab="Millions reflecting Inflation" );
> boxplot(denzel$millions.2000, main=denzel$name, ylim=c(0,360),
      ylab="Millions reflecting Inflation" );
>
> par(mfrow=c(1,1))
```



Based on these boxplots, I think that it is reasonable to say that Will Smith has brought in more money in the industry. Although it looks like the median for each of the actors is only about 25 million different, the maximum value (the top whisker), along with the third quartile are significantly higher for Will than they are for Denzel. The spread on the profit of movies that Will has been in is much higher than that of Denzel.

The next set of boxplots built show the distribution of release years of the top 50 movies that each actor has been in. Below is the R code and the plots.

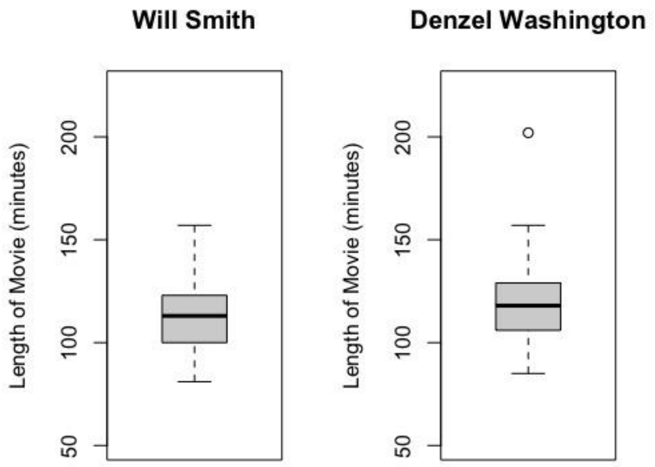
```
> par(mfrow=c(1,2));
> boxplot(will$movies.50$year, main=will$name, ylim=c(1980, 2030),
      ylab="Years of Movie Release");
> boxplot(denzel$movies.50$year, main=denzel$name, ylim=c(1980, 2030),
      ylab="Years of Movie Release");
>
> par(mfrow=c(1,1))
```



Based on these boxplots, I concluded that Will Smith's popular movies are generally more recent than Denzel Washington's. Denzel Washington also has much more of spread in years than Will Smith does based on the minimum and maximum shown by the whiskers. The median release year for Will Smith is somewhere around 2005, whereas the median for Denzel Washington looks to be around 1999. Both box plots are relatively symmetrical, signifying that the distribution is not overly skewed.

The next set of boxplots built show the distribution of the length of the top 50 movies (in minutes) that each actor has been in. Below is the R code and the plots.

```
> par(mfrow=c(1,2));
> boxplot(will$movies.50$minutes, main=will$name, ylim=c(50, 225),
      ylab="Length of Movie (minutes)");
> boxplot(denzel$movies.50$minutes, main=denzel$name, ylim=c(50, 225),
      ylab="Length of Movie (minutes)");
>
> par(mfrow=c(1,1))
```



These boxplots are relatively similar. The main difference that can be seen is the outlier in Denzel Washington's plot around the 200-minute mark. Other than the one outlier, it appears that the movies that both actors have been in range from about 80 minutes to 160 minutes, based on the spread of the whiskers. The median number of minutes for both actors looks to be between 100 and 125 minutes, and 50% of all movies for both actors look to be between 100 and 130 minutes based on the "box" part of the boxplot.