DSC 520 Final Project Step 2

Taylor Imhof

11/11/2021

Data Import and Cleaning Efforts

Fortunately, all of my data sets came in .csv format, so reading in the data to data frames was straightforward using the built-in read.csv() function.

```
# read in data to data frames
happiness_2018_df <- read.csv('data/happiness_2018.csv', stringsAsFactors = FALSE)
lifestyle_df <- read.csv('data/lifestyle.csv', stringsAsFactors = FALSE)
human_df <- read.csv('data/human_development.csv', stringsAsFactors = FALSE)
```

To get a basic understanding of how the data is designed, I ran str() on each of them to see the different data types.

```
# check structure of each data frame
str(happiness_2018_df)
str(lifestyle_df)
str(human_df)
```

I did not really like how the original column names were written, so I made some slight adjustments to make it easier for me to understand and code later on.

```
# rename columns for easier coding
old1 <- c('Overall.rank', 'Country.or.region', 'GDP.per.capita', 'Social.support',
          'Healthy.life.expectancy', 'Freedom.to.make.life.choices', 'Perceptions.of.corruption')
new1 <- c('Rank', 'Country', 'GDP', 'socialSupport', 'lifeExpectancy', 'choiceFreedom',
          'Corruption')
old2 <- c('HDI.Rank', 'Human.Development.Index..HDI.', 'Life.Expectancy.at.Birth',
          'Expected.Years.of.Education' ,'Mean.Years.of.Education',
          'Gross.National.Income..GNI..per.Capita','GNI.per.Capita.Rank.Minus.HDI.Rank')
new2 <- c('HDIrank', 'HDI', 'lifeExpectancy', 'expectedEducationYears',
          'meanEducationYears', 'GNI', 'GNIminusHDIrank')
setnames(happiness_2018_df, old=old1, new=new1)
setnames(human_df, old=old2, new=new2)
```

Right off the bat, I noticed that the `Corruption` column had chr values. To fix this, I first set the imputed all of the NA values with the mean of the distribution for that column. Then, I converted the column's values to numeric values using the as.numeric() function.

```
# convert perception.of.corruption from chr to num
happiness_2018_df$Corruption[is.na(happiness_2018_df$Corruption)] <- mean(happiness_2018_df$Corruption,
```

```
## Warning in mean.default(happiness_2018_df$Corruption, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
happiness_2018_df$Corruption <- as.numeric(as.character(happiness_2018_df$Corruption))
```

```
## Warning: NAs introduced by coercion
```

```
str(happiness_2018_df)
```

In a similar vein, the `DAILY_STRESS` column in the lifestyle data frame was also interpreted as characters. I simply followed the same logic I used above to correct for this.

```
lifestyle_df$DAILY_STRESS[is.na(lifestyle_df$DAILY_STRESS)] <- mean(lifestyle_df$DAILY_STRESS, na.rm=TRU
```

```
## Warning in mean.default(lifestyle_df$DAILY_STRESS, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
lifestyle_df$DAILY_STRESS <- as.numeric(as.character(lifestyle_df$DAILY_STRESS))
```

```
## Warning: NAs introduced by coercion
```

```
str(lifestyle_df)
```

Next, in the lifestyle data frame, the column `GENDER` used character values to denote gender, so I went ahead and converted these to binary values so that they could be better used during analysis. I created a new column `SEX` and converted all Male values to 1 and all female values to 0.

```
# convert GENDER column to binary values
lifestyle_df$SEX[lifestyle_df$GENDER=='Male'] <- 1
lifestyle_df$SEX[lifestyle_df$GENDER=='Female'] <- 0
str(lifestyle_df)
```

Since the lifestyle data frame doesn't have any values that can be used to merge with the other data frames, I decided to use it to analyze which lifestyle factors can improve overall health (this data set used a calculated work-life-balance). The other two data frames both contained country names, so merging them together was straightforward. Using the merge() function, I was able to combine the two together.

```
# merge happiness and HDI dfs together on country name
combo_df <- happiness_2018_df %>% merge(human_df, by='Country', all.x = T)
str(combo_df)
```

In this new data frame, the `GNI` column contained character values. I predicted this column would be a useful predictor, so I need to convert it to numeric values. Following a similar procedure I used earlier, I converted all the values using the as.numeric() function. A pesky issue that was giving me grief was the starting values all contained commas. To remedy this, I used gsub() to replace them.

```
# replace commas with white space
combo_df$GNI <- as.numeric(gsub(",", "", combo_df$GNI))
combo_df$GNI[is.na(combo_df$GNI)] <- mean(combo_df$GNI, na.rm=TRUE)
combo_df$GNI <- as.numeric(combo_df$GNI)
```

The next thing I decided to do was make sure that my data did not contain any NA values. I found a pretty slick implementation of the map_df from the purrr package that runs through each column and sums up all the NA values. There were quite a few NA values for columns that were added during the merge. This was expected, as there were likely country's not included in both. To remedy this, I opted to impute the means for these NA values. Lastly, I did the same operation to check for NA values on the lifestyle data frame. However, fortunately all of the values contained in this data set were "clean" and contained no NA values.

```
# check for NA values using map_df from purrr
combo_df %>% map_df(~sum(is.na(.)))

# impute mean for columns with NA values
for(i in 1:ncol(combo_df)){
  combo_df[, i][is.na(combo_df[,i])] <- mean(combo_df[,i], na.rm=TRUE)
}
```

```
## Warning in mean.default(combo_df[, i], na.rm = TRUE): argument is not numeric or
## logical: returning NA
```

```
# check again for NA values to ensure imputation was successful
combo_df %>% map_df(~sum(is.na(.)))

# check for NA values in lifestyle_df
lifestyle_df %>% map_df(~sum(is.na(.)))

# impute mean for columns with NA values
for(i in 1:ncol(lifestyle_df)){
  lifestyle_df[, i][is.na(lifestyle_df[,i])] <- mean(lifestyle_df[,i], na.rm=TRUE)
}
```

```
## Warning in mean.default(lifestyle_df[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(lifestyle_df[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(lifestyle_df[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
# convert lifestyle values from integer to numeric
lifestyle_df[1:22] <- lapply(lifestyle_df[1:22], as.numeric)
```

```
## Warning in lapply(lifestyle_df[1:22], as.numeric): NAs introduced by coercion

## Warning in lapply(lifestyle_df[1:22], as.numeric): NAs introduced by coercion
```

```
str(lifestyle_df)
```

The last thing I decided to do during the data cleaning stage was narrow down columns for each data frame. During the merge of the HDI and WHR data sets, two life expectancy columns were created. However, only one seems to use years as a metric, so I opted to keep this column. There are two rank scores as well, both of which I decided would not be useful for analysis so they were dropped. In the lifestyle data frame, I decided

the timestamp info would not be useful. Also, since I created a new variable of binary values based on the GENDER column, it was dropped. The age column also contained some strange values, so I thought it would be easier to just leave it out of the final data frame.

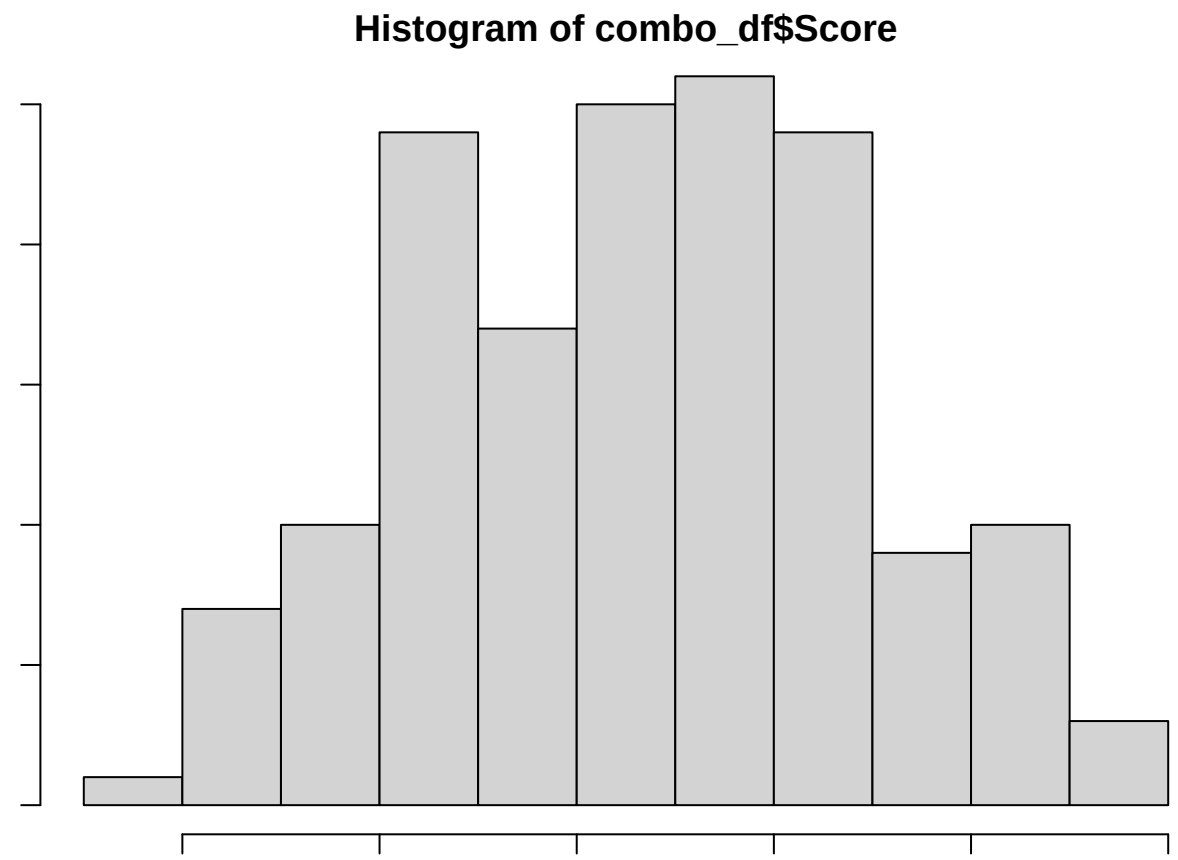
```
#
dropCols1 <- c('Rank','lifeExpectancy.x','HDIrank', 'GNIminusHDIrank')
combo_df <- combo_df[,!(names(combo_df) %in% dropCols1)]

dropCols2 <- c('ï..Timestamp', 'AGE', 'GENDER')
lifestyle_df <- lifestyle_df[,!(names(lifestyle_df) %in% dropCols2)]
str(lifestyle_df)
```

Final Two "Cleaned" Data Frames For Analysis After performing the munging operations outlined above, I was pretty happy with my two data sets. Below is the output of the str() function on both of them for review.

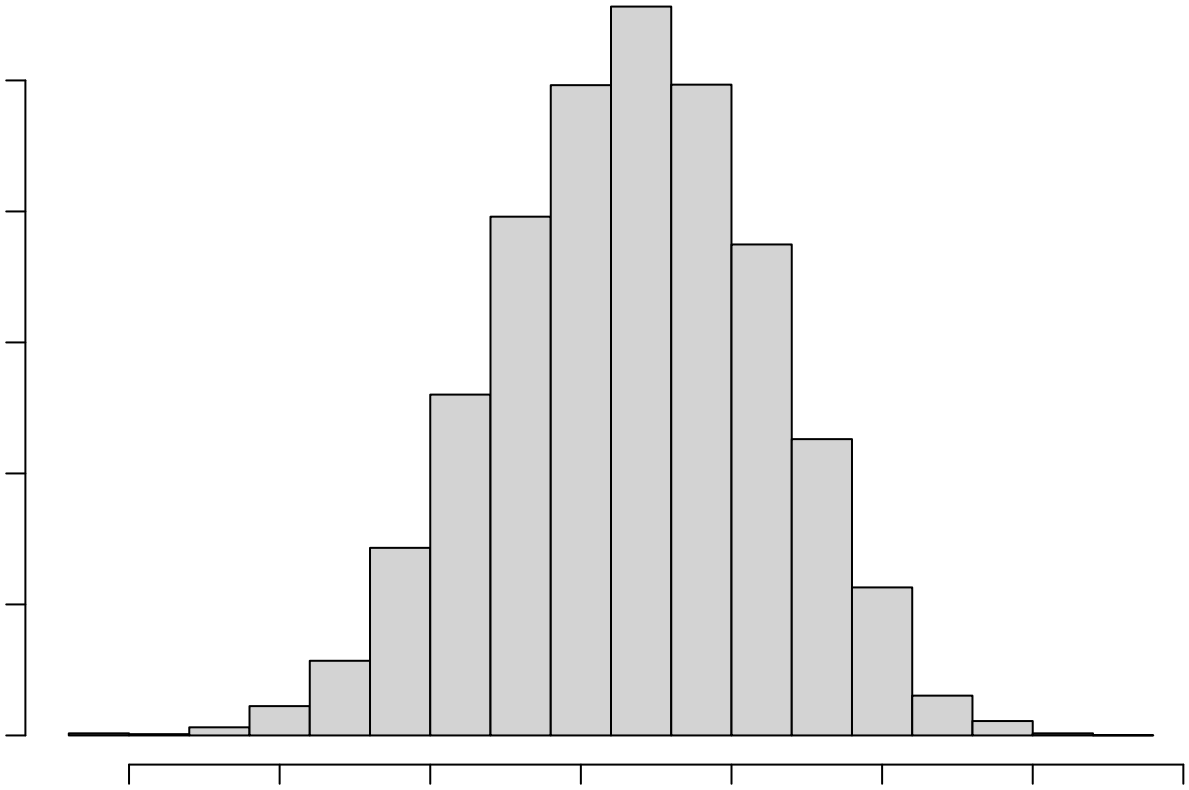
```
## 'data.frame':    156 obs. of  12 variables:
##  $ Country               : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ Score                 : num  3.63 4.59 5.29 3.79 6.39 ...
##  $ GDP                   : num  0.332 0.916 0.979 0.73 1.073 ...
##  $ socialSupport         : num  0.537 0.817 1.154 1.125 1.468 ...
##  $ choiceFreedom         : num  0.085 0.419 0.077 0 0.57 0.26 0.647 0.617 0.43 0.594 ...
##  $ Generosity            : num  0.191 0.149 0.055 0.079 0.062 0.077 0.361 0.242 0.031 0.243 ...
##  $ Corruption            : num  0.036 0.032 0.135 0.061 0.054 0.028 0.302 0.224 0.176 0.123 ...
##  $ HDI                   : num  0.465 0.733 0.736 0.532 0.836 0.733 0.935 0.885 0.751 0.824 ...
##  $ lifeExpectancy.y      : num  60.4 77.8 74.8 52.3 76.3 74.7 82.4 81.4 70.8 76.6 ...
##  $ expectedEducationYears: num  9.3 11.8 14 11.4 17.9 12.3 20.2 15.7 11.9 14.4 ...
##  $ meanEducationYears    : num  3.2 9.3 7.6 4.7 9.8 10.9 13 10.8 11.2 9.4 ...
##  $ GNI                   : num  1885 9943 13054 6822 22050 ...

## 'data.frame':    15972 obs. of  22 variables:
##  $ FRUITS_VEGGIES         : num  3 2 2 3 5 3 4 3 5 4 ...
##  $ DAILY_STRESS           : num  2 3 3 3 1 2 2 4 3 4 ...
##  $ PLACES_VISITED         : num  2 4 3 10 3 3 10 5 6 2 ...
##  $ CORE_CIRCLE            : num  5 3 4 3 3 9 6 3 4 6 ...
##  $ SUPPORTING_OTHERS      : num  0 8 4 10 10 10 10 5 3 10 ...
##  $ SOCIAL_NETWORK         : num  5 10 10 7 4 10 10 7 3 10 ...
##  $ ACHIEVEMENT            : num  2 5 3 2 2 2 3 4 5 0 ...
##  $ DONATION               : num  0 2 2 5 4 3 5 0 4 4 ...
##  $ BMI_RANGE              : num  1 2 2 2 2 1 2 1 1 2 ...
##  $ TODO_COMPLETED         : num  6 5 2 3 5 6 8 8 10 3 ...
##  $ FLOW                   : num  4 2 2 5 0 1 8 2 2 2 ...
##  $ DAILY_STEPS            : num  5 5 4 5 5 7 7 8 1 3 ...
##  $ LIVE_VISION            : num  0 5 5 0 0 10 5 10 5 0 ...
##  $ SLEEP_HOURS            : num  7 8 8 5 7 8 7 6 10 6 ...
##  $ LOST_VACATION          : num  5 2 10 7 0 0 10 0 0 0 ...
##  $ DAILY_SHOUTING         : num  5 2 2 5 0 2 0 2 2 0 ...
##  $ SUFFICIENT_INCOME      : num  1 2 2 1 2 2 2 2 2 1 ...
##  $ PERSONAL_AWARDS        : num  4 3 4 5 8 10 10 8 10 3 ...
##  $ TIME_FOR_PASSION       : num  0 2 8 2 1 8 8 2 3 8 ...
##  $ WEEKLY_MEDITATION      : num  5 6 3 0 5 3 10 2 10 1 ...
##  $ WORK_LIFE_BALANCE_SCORE: num  610 656 632 623 664 ...
##  $ SEX                    : num  0 0 0 0 0 0 1 0 0 0 ...
```

Plots and Data Transformations For my two final data frames, I decided that `Score` and `WORK_LIFE_BALANCE_SCORE` would be the best target variables. I ran a quick hist() on both of these features to ensure that they were both normally distributed. Both appeared to be normal distributions. In fact, the `WORK_LIFE_BALANCE_SCORE` looked like a textbook example of the normal curve.

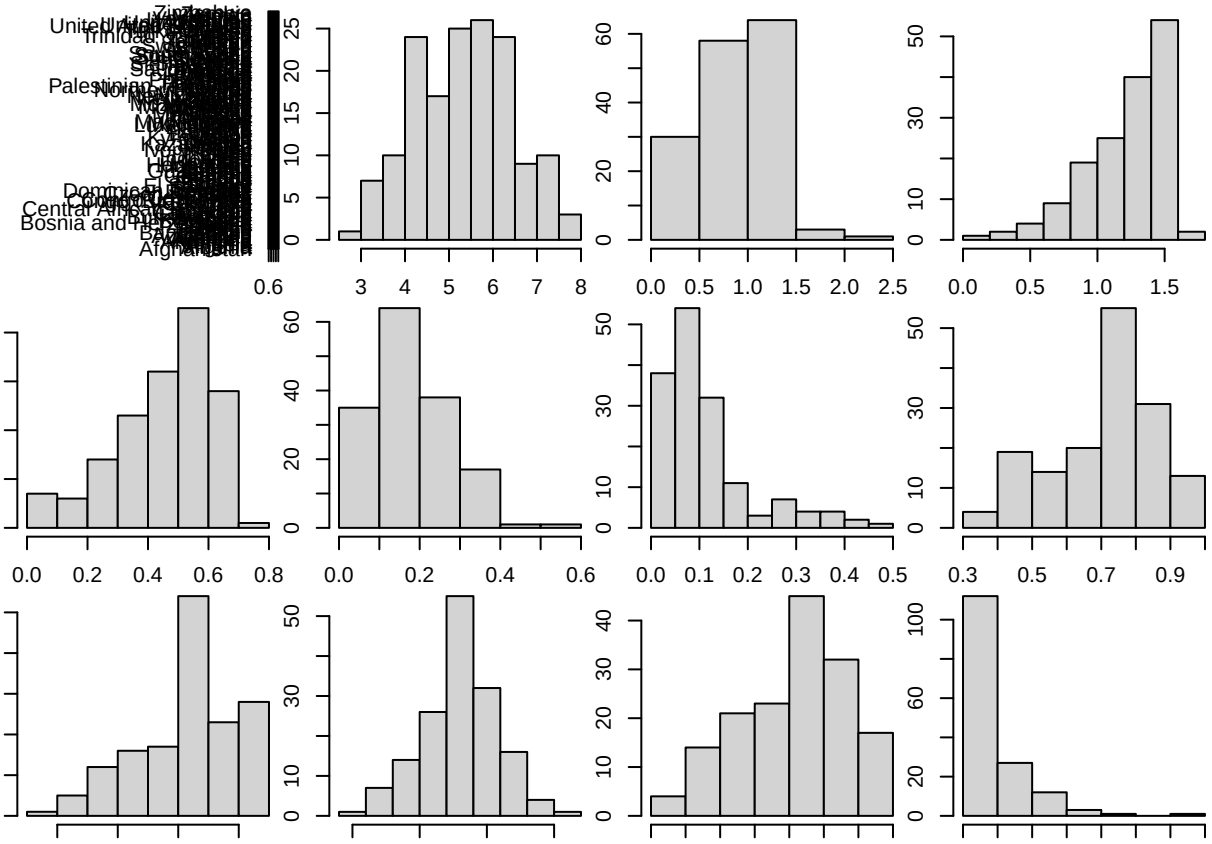
```
par(mar=c(1,1,1,1))

# check distribution of target variables for both data frames
hist(combo_df$Score)
```



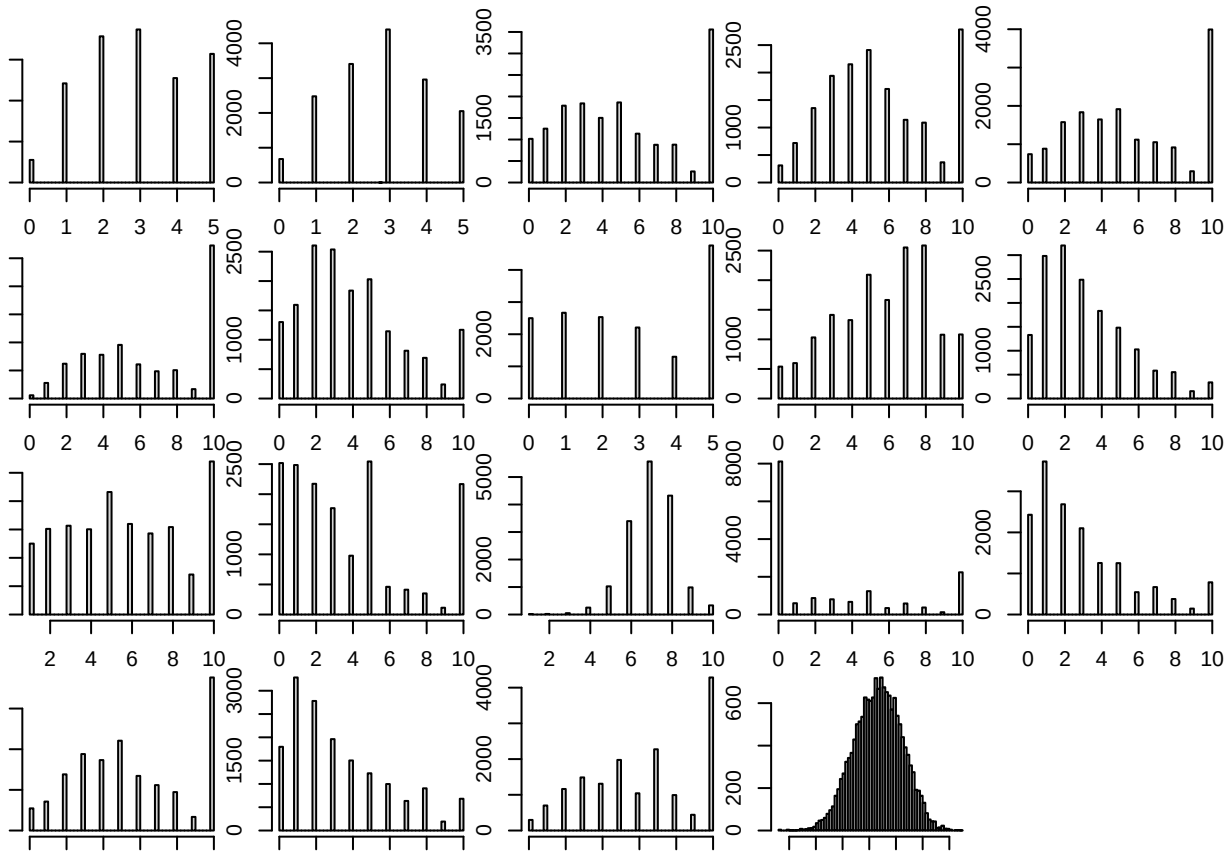

```
hist(lifestyle_df$WORK_LIFE_BALANCE_SCORE)
```




```
# check histograms for every column
hist.data.frame(combo_df)
```



```
hist.data.frame(lifestyle_df)
```

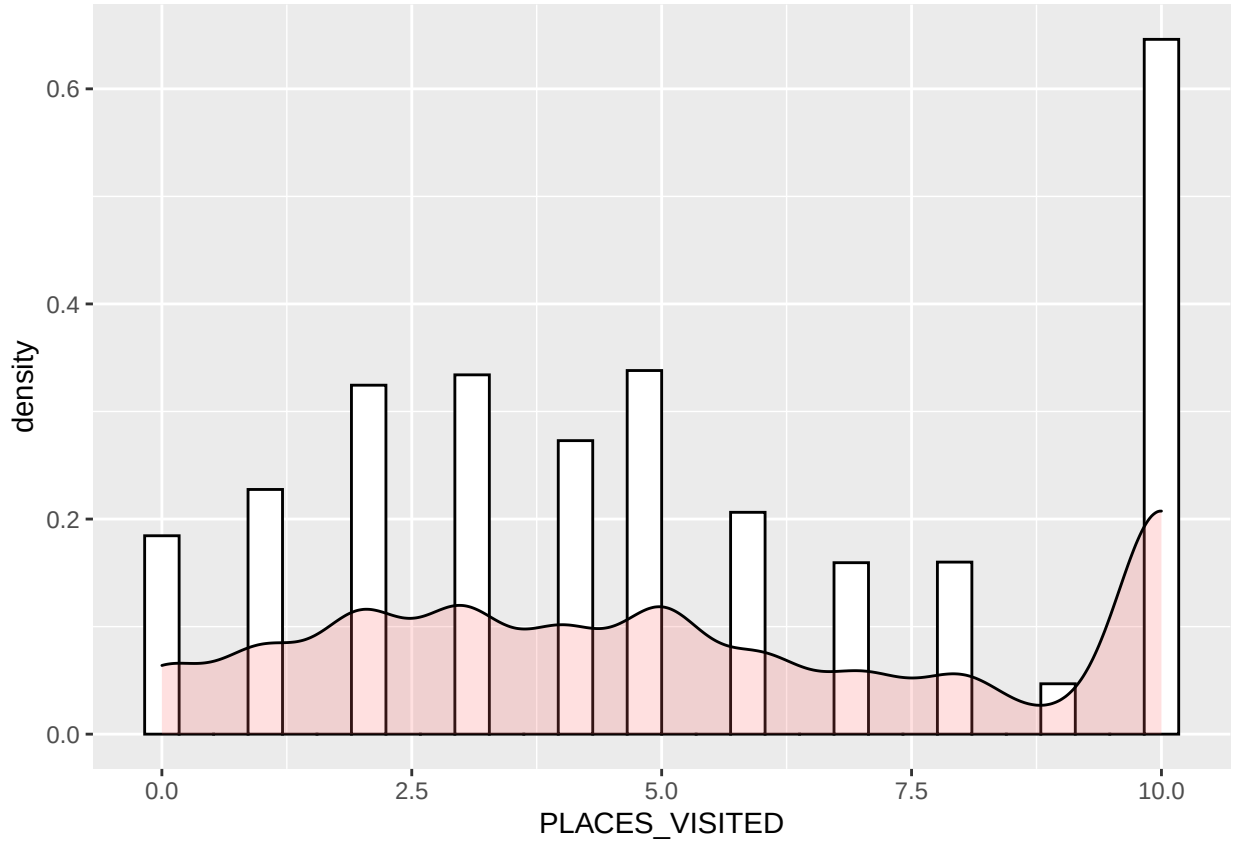


Next, to find which features would make for good predictors, I ran a Pearson's correlation test on the numeric values of each data frame. Some of the features that stuck out for the combo_df were `GDP`, `HDI`, `socialSupport`, and `lifeExpectancy`. After looking at the distributions for each of these potential explanatory features, there were two negatively skewed distributions that could use transformation. These were the `socialSupport` and `lifeExpectancy` columns respectively. After creating a new variable contained the squared values, the distributions did appear to take on a more normal distribution.

For the lifestyle data frame with the work-life balance score as the dependent variable, there did not appear to be any variables with strong correlations. The ones of note were `PLACES_VISITED`, `TIME_FOR_PASSION`, `SUPPORTING_OTHERS`, and `ACHIEVEMENT`.

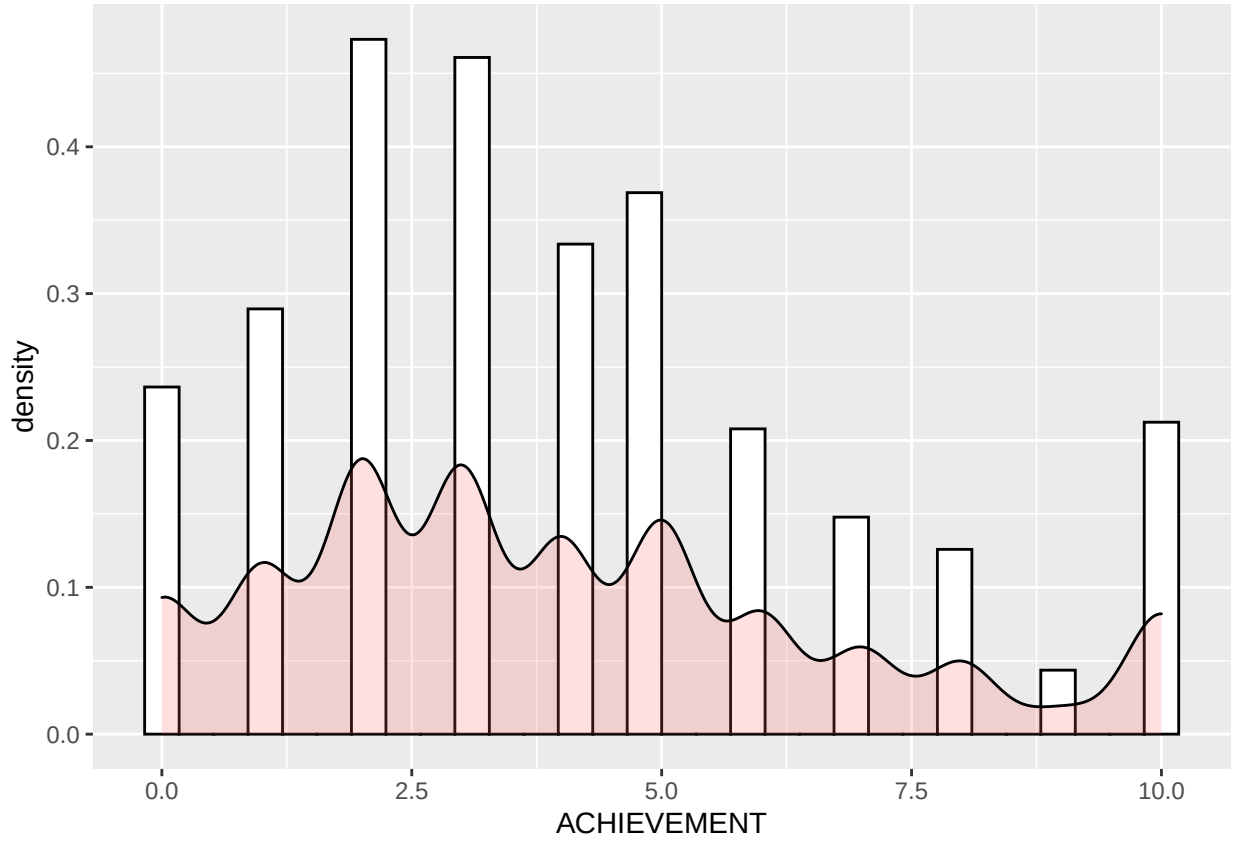
```
par(mar=c(1,1,1,1))

cor(lifestyle_df, method='pearson')
str(lifestyle_df)

ggplot(lifestyle_df, aes(x=PLACES_VISITED)) +
  geom_histogram(aes(y=..density..), colour='black', fill='white') +
  geom_density(alpha=.2, fill='#FF6666')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



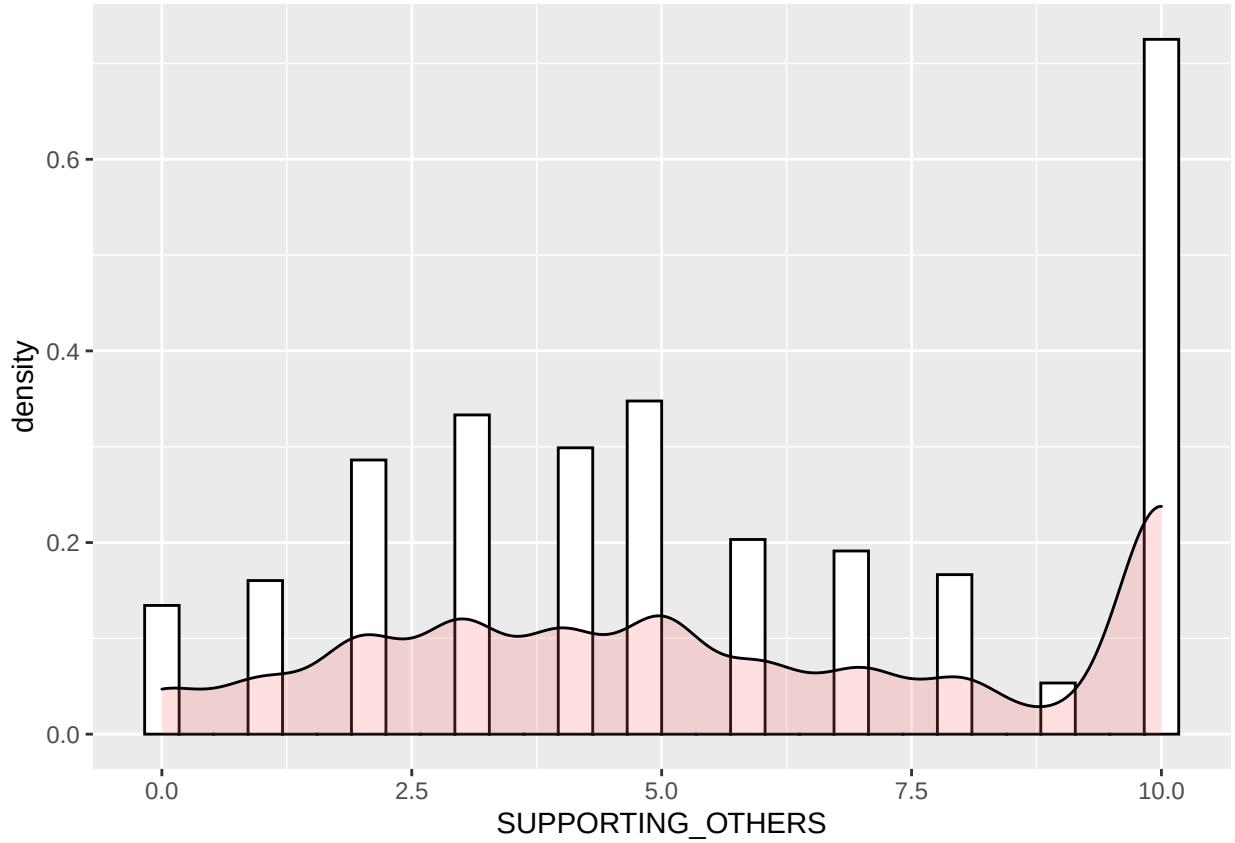

```
ggplot(lifestyle_df, aes(x=ACHIEVEMENT)) +
  geom_histogram(aes(y=..density..), colour='black', fill='white') +
  geom_density(alpha=.2, fill='#FF6666')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



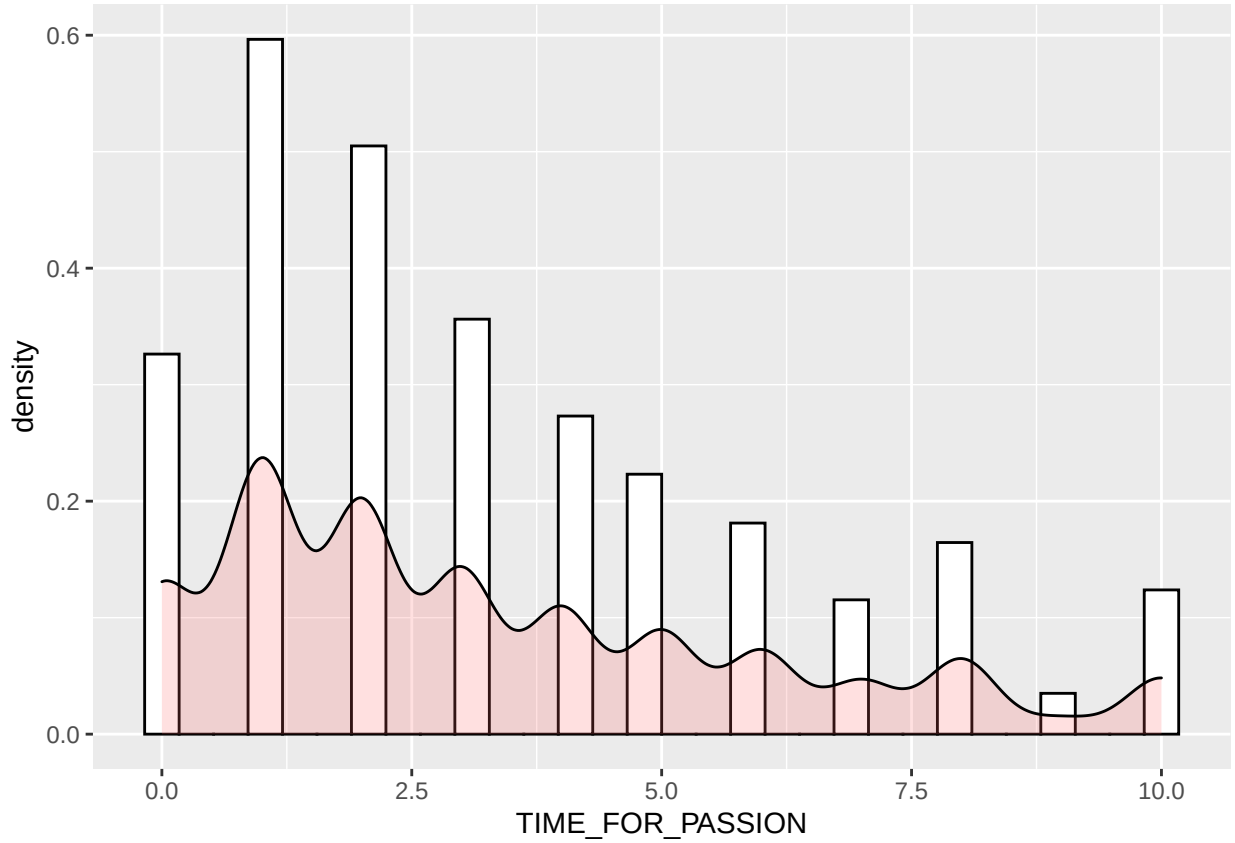

```
ggplot(lifestyle_df, aes(x=SUPPORTING_OTHERS)) +
  geom_histogram(aes(y=..density..), colour='black', fill='white') +
  geom_density(alpha=.2, fill='#FF6666')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



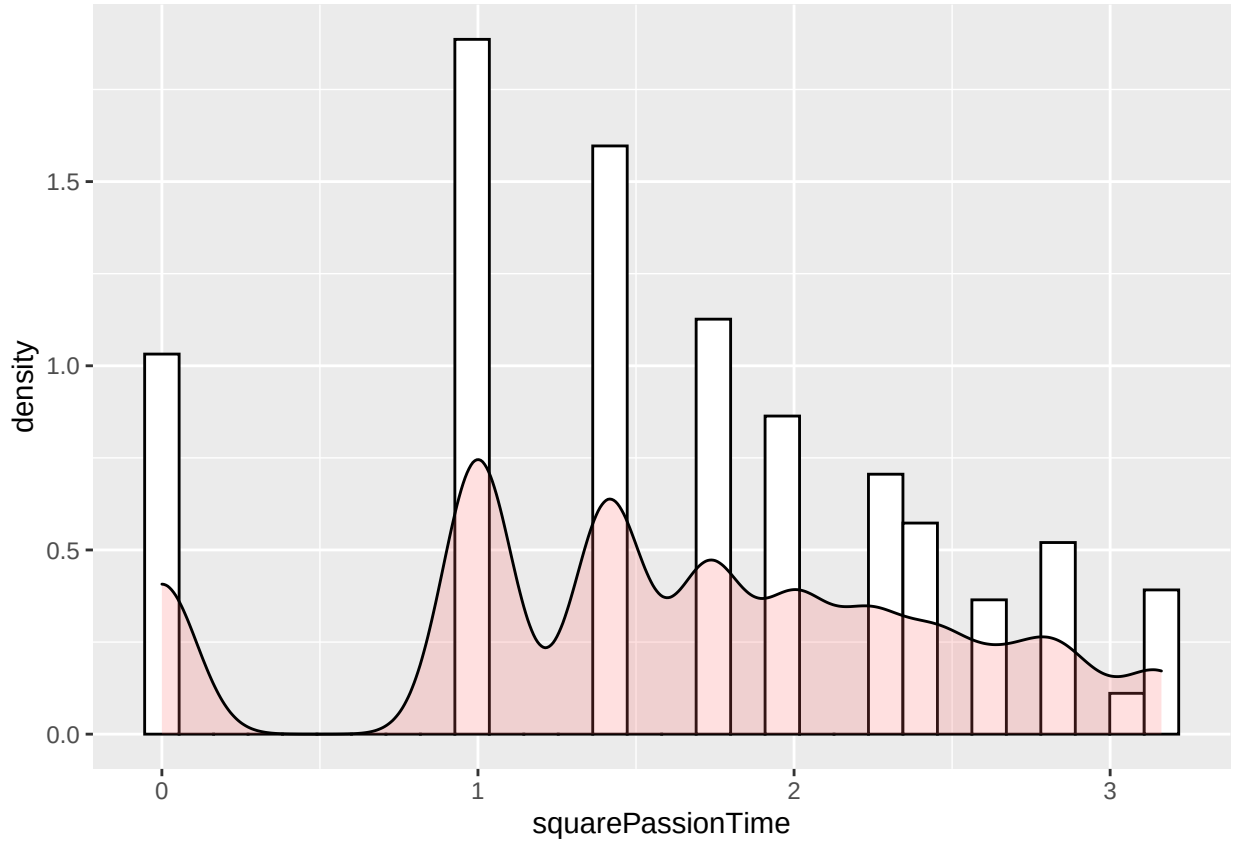

```
# take square root of time for passion to get more normal distro
lifestyle_df$squarePassionTime <- lifestyle_df$TIME_FOR_PASSION ^(1/2)

ggplot(lifestyle_df, aes(x=TIME_FOR_PASSION)) +
  geom_histogram(aes(y=..density..), colour='black', fill='white') +
  geom_density(alpha=.2, fill='#FF6666')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(lifestyle_df, aes(x=squarePassionTime)) +
  geom_histogram(aes(y=..density..), colour='black', fill='white') +
  geom_density(alpha=.2, fill='#FF6666')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


density
1.5
1.0
0.5
0.0
0
1
2
3
squarePassionTime