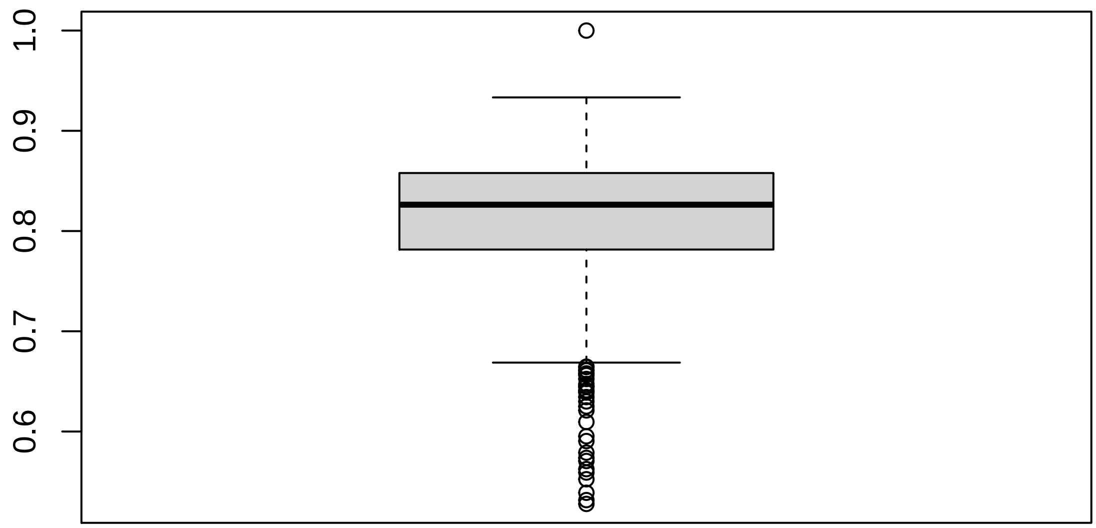
Homework 6

Ryan Richardson

08/12/2020

Question 1

```
scorePlot = boxplot(demo$Prof_Score)
```




```
scoreOutliers = scorePlot$out
scoreBreaks = scorePlot$stats

outlierMin = min(scoreOutliers)
outlierMax = max(scoreOutliers)
```

IQR

```
minRange = c(outlierMin, first(scoreBreaks))
maxRange = c(last(scoreBreaks), outlierMax)
```

```
outlierCount = length(scoreOutliers)

minRange
```

```
## [1] 0.5277778 0.6687500
```

```
maxRange
```

```
## [1] 0.9333333 1.0000000
```

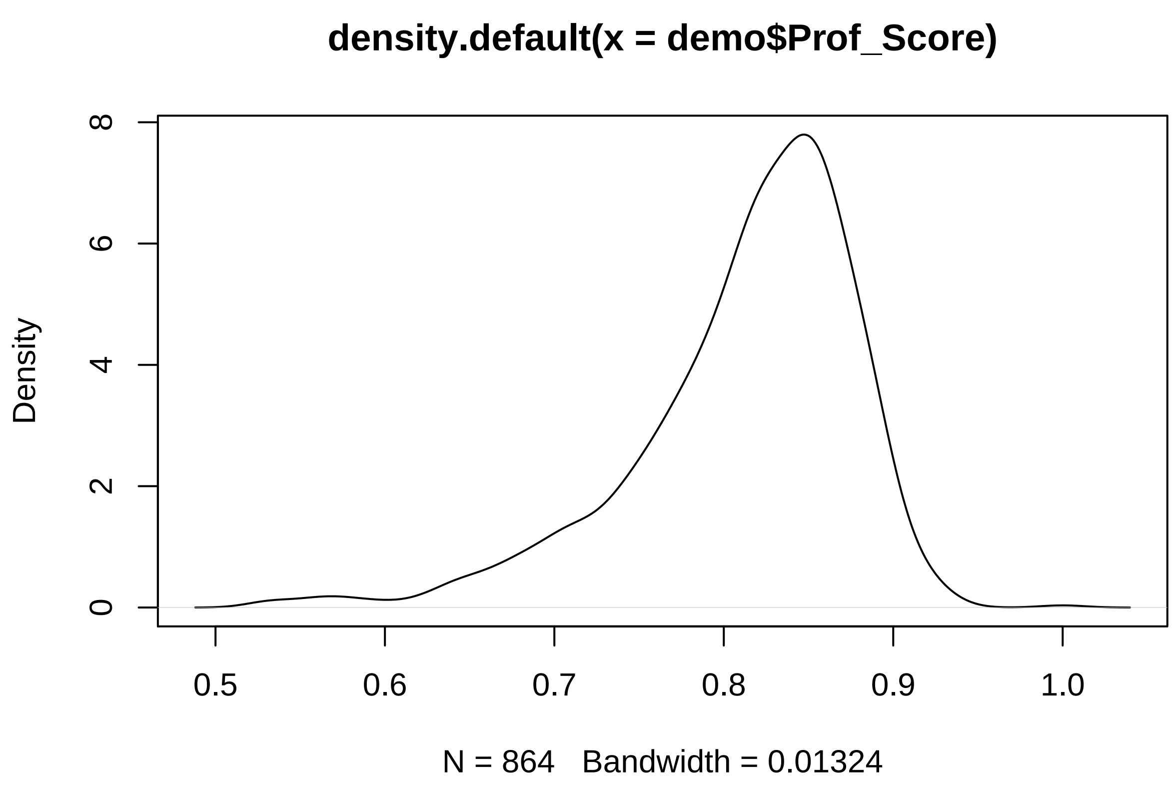
```
outlierCount
```

```
## [1] 32
```

Outliers range from [0.5278, 0.6688) and (0.9333,1] There are a total of 32 outliers using IQR as our detection method.

Standard Deviation

```
plot(density(demo$Prof_Score))
```

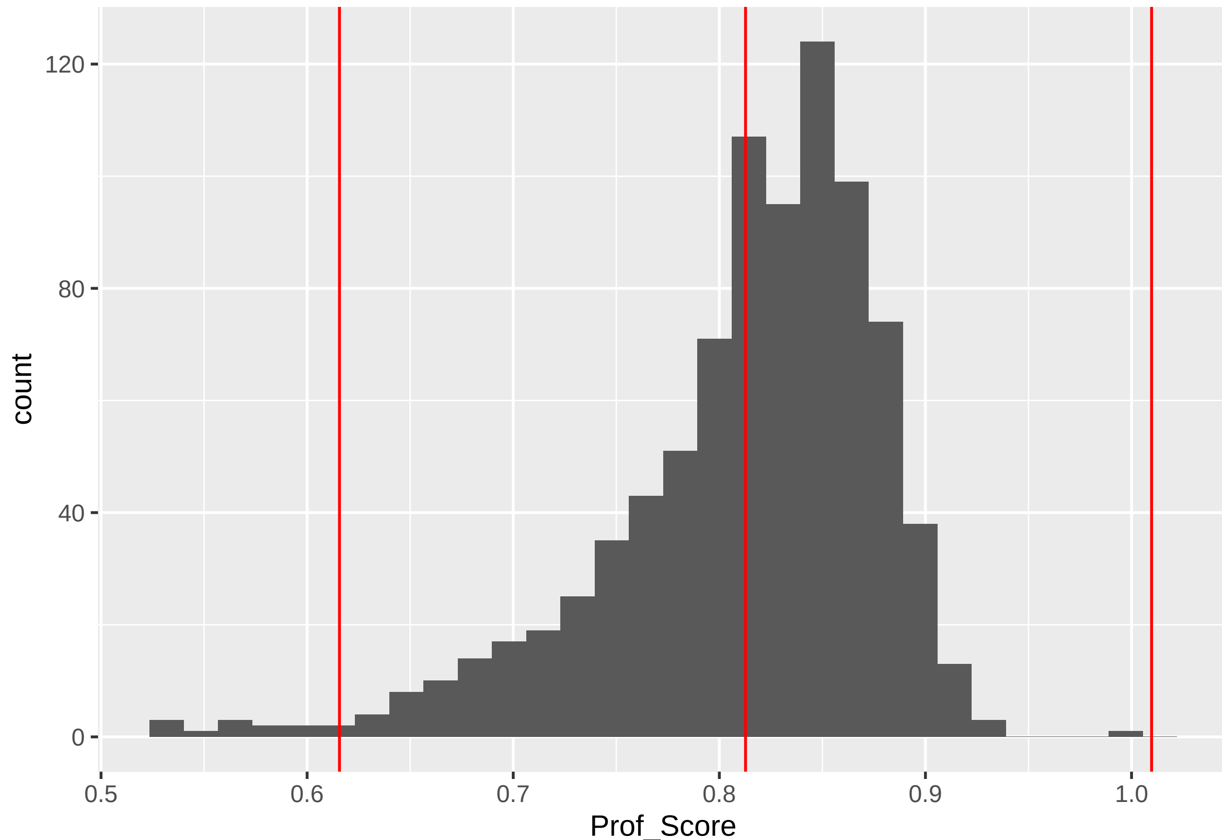



```
scoreSd = sd(demo$Prof_Score)
scoreMean = mean(demo$Prof_Score)

upper = scoreMean + (3*scoreSd)
lower = scoreMean - (3*scoreSd)

upperOut = demo$Prof_Score > upper
```

```r
lowerOut = demo$Prof_Score < lower


sdUpperOutliers = demo$Prof_Score[upperOut]
sdLowerOutliers = demo$Prof_Score[lowerOut]
```

```r
ggplot(demo, aes(x=Prof_Score)) +
  geom_histogram() +
  geom_vline(xintercept = lower, color='red') +
  geom_vline(xintercept = upper, color='red') +
  geom_vline(xintercept = scoreMean, color='red')
```

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




```r
sdOutliers = c(sdLowerOutliers, sdUpperOutliers)
sdOutliers
```

```
##  [1] 0.5315789 0.5523810 0.5388889 0.5590909 0.6095238 0.5625000 0.5277778
##  [8] 0.5705882 0.5736842 0.5952381 0.5789474 0.5904762
```

```r
sdOutlierRange = c(min(sdOutliers), max(sdOutliers))
sdOutlierRange
```

```
## [1] 0.5277778 0.6095238
```

```r
length(sdOutliers)
```

```
## [1] 12
```

Using the Standard Deviation to catch outliers resulted in only 12 outliers being found between [0.528, 0.61]

The upper range of the outliers ended up being larger a score greater than 100%, which would have been impossilbe to achieve in this case, meaning there were no outliers found in the upper range, this is likely due to the heavily skewed distribution of scores.

Inferential Statistics

```
scores = demo$Prof_Score
gTest = grubbs.test(scores)
gOut = c()
while (gTest$p.value < 0.05)
{
  altValue = gTest$alternative
  altValue = as.numeric(strsplit(altValue," ")[[1]][3])
  gOut = append(gOut, altValue)
  scores = scores[!scores %in% altValue]
  gTest = grubbs.test(scores)
}

gRange = c(min(gOut), max(gOut))
gRange
```

```
## [1] 0.5277778 0.5789474
```

```
length(gOut)
```

```
## [1] 9
```

Using Grubbs test for each value in the list, we end up with 9 outliers between [0.5278, 0.5789]. Grubbs test also missed the outlier at 1.0 due to taking out the smallest values first.

I think that if outliers are to be removed then the IQR method is the one to use. It's the only method that accurately compensated for the outliers at the high end of the distribution, and was the least affected by the extremely skewed dataset. That said, the outliers themselves were never so far out of the range of possibility that they needed to be removed. It's not unreasonable for some students to receive a 50% or for some to receive a 100%, so if I was able to, I would probably choose not to remove any outliers at all.

Question 2

NAs appear only in Image_Quality and Smile. with a total of 3396 total records with NA values where 277 of those records have an NA in both Image_Quality and Smile.

```
missingValuesByColumn = colSums(is.na(df))
```

```
df$Image_QualityNA = as.factor(ifelse(is.na(df$Image_Quality),1,0))
df$SmileNA = as.factor(ifelse(is.na(df$Smile),1,0))

rForestImage = randomForest(Image_QualityNA ~ ., data = df[-c(5,10,13)])
rForestImage
```

```
##
## Call:
##  randomForest(formula = Image_QualityNA ~ ., data = df[-c(5, 10,      13)])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
```

```
##
##         OOB estimate of  error rate: 8.97%
## Confusion matrix:
##      0  1 class.error
## 0 7937 52  0.00650895
## 1  736 62  0.92230576
```

```
rForestSmile = randomForest(SmileNA ~ ., data = df[-c(5,10,12)])
rForestSmile
```

```
##
## Call:
##  randomForest(formula = SmileNA ~ ., data = df[-c(5, 10, 12)])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 33.14%
## Confusion matrix:
##      0  1 class.error
## 0 5836 76  0.01285521
## 1 2836 39  0.98643478
```

Based on the Random Forest test, the Image Quality is neither MCAR nor MAR, as the prediction had a fairly low error rate of 8.93% when estimating NAs for Image Quality, and that the NAs are MNAR.

Based on the Random Forest test, the Smile is also likely not MCAR or MAR. The error rate of the prediction is at 33.24% is still accurate enough to assume that the NAs are MNAR instead.

```
df = df[-c(12,13)]
dfListDelete = df[complete.cases(df),]
imputeModel = mice(df, maxit=0)
imputePredictions = imputeModel$predictorMatrix
dfMultipleImpute = mice(df, maxit=5, predictorMatrix = imputePredictions, method = imputeModel$method, p
dfImputeComplete = complete(dfMultipleImpute,1)

summary(glm(Area~., data = dfListDelete))
```

```
##
## Call:
## glm(formula = Area ~ ., data = dfListDelete)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -153151   -25364    -9754     8395  1443587
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)           43769      12857   3.404 0.000668 ***
## Gendermale             7090       2448   2.896 0.003790 **
## Genderunknown         57567      10593   5.435 5.74e-08 ***
## Adult                 -4571       4463  -1.024 0.305859
## Face_Angle             1131       2373   0.476 0.633767
## Image_Color            3471       2142   1.621 0.105144
## Image_Qualitygood     24091       2186  11.018  < 2e-16 ***
## Image_Type             6051       3415   1.772 0.076460 .
## Contextauthor        -23080       9813  -2.352 0.018706 *
```

```
## Contextcover           68235       5370  12.706  < 2e-16 ***
## Contextfeature        -16050       2663  -6.026 1.79e-09 ***
## Multiface             -25740       2169 -11.867  < 2e-16 ***
## Raceasian              -4762      12134  -0.392 0.694735
## Raceblack             -10334      11869  -0.871 0.383943
## Racepacificislander   -11449      16079  -0.712 0.476464
## Raceunknown             7345      13836   0.531 0.595534
## Racewhite              -8860      11234  -0.789 0.430366
## Smile                  -4582       2204  -2.079 0.037630 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5856141474)
##
##     Null deviance: 3.5574e+13  on 5390  degrees of freedom
## Residual deviance: 3.1465e+13  on 5373  degrees of freedom
## AIC: 136567
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(glm(Area~., data = dfImputeComplete))
```

```
##
## Call:
## glm(formula = Area ~ ., data = dfImputeComplete)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -157044   -26456   -10582     7388  1466060
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)            26825      10387   2.583 0.009823 **
## Gendermale              7480       1923   3.890 0.000101 ***
## Genderunknown          69397       8743   7.937 2.32e-15 ***
## Adult                   1334       3382   0.394 0.693342
## Face_Angle              1278       1856   0.689 0.491115
## Image_Color             5689       1690   3.366 0.000766 ***
## Image_Qualitygood      24365       1720  14.167  < 2e-16 ***
## Image_Type              9198       2589   3.553 0.000383 ***
## Contextauthor         -24864       5931  -4.192 2.79e-05 ***
## Contextcover           54461       4412  12.343  < 2e-16 ***
## Contextfeature        -19259       1972  -9.769  < 2e-16 ***
## Multiface             -24205       1709 -14.165  < 2e-16 ***
## Raceasian               5002       9890   0.506 0.612993
## Raceblack              -3298       9664  -0.341 0.732875
## Racepacificislander   -10332      12796  -0.807 0.419425
## Raceunknown             8214      11118   0.739 0.460035
## Racewhite                189       9186   0.021 0.983585
## Smile                  -3258       1738  -1.875 0.060847 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5917085092)
##
```

```
##     Null deviance: 5.7620e+13  on 8786  degrees of freedom
## Residual deviance: 5.1887e+13  on 8769  degrees of freedom
## AIC: 222674
##
## Number of Fisher Scoring iterations: 2
```

Looking at the two models, the first with Listwise Deletion and the second with Imputed Data, the first model appears to better fit the data based on its lower AIC value. This model also has much lower deviance across the board compared to the imputed model, which is reflected in its lowered AIC. In this case, it seems to be the correct choice to use listwise deletion to remove all records with missing values instead of attempting to impute them.