Homework 5

Matt Johnson

10/23/2020

Q1

Read in data

```
DF <- read_csv("https://raw.githubusercontent.com/Vincent-Toups/bios611-project1/master/source_data/data
write.csv(DF, "Data.csv")
```

Split up data

```
DF$Gender = ifelse(DF$Gender == "Male", 1, 0) #Male is 1, female is 0

set.seed <- 18 #this is my lucky number
spec = c(train = .6, test = .2, validate = .2)
DF1 = sample(cut(
  seq(nrow(DF)),
  nrow(DF)*cumsum(c(0,spec)),
  labels = names(spec)
))

DF.Split = split(DF, DF1)
```

GBM

```
gbm1 <- gbm(Gender ~ Height + Weight, data = DF.Split$train, distribution = "bernoulli")

DF.Split$validate$gbm1.probs <- predict(gbm1, newdata = DF.Split$validate, type = "response")
DF.Split$validate <- DF.Split$validate %>%
  mutate(gbm1_pred = 1*(gbm1.probs > .5) + 0) %>%
  mutate(accurate.gbm1 = 1*(gbm1_pred == Gender))
sum(DF.Split$validate$accurate.gbm1)/nrow(DF.Split$validate)
```

```
## [1] 0.9145
```

The Accuracy of the GBM model is about .9 on the validation set. This is much much better than the previous gbm that had an accuracy around .5

Q2

Read in data

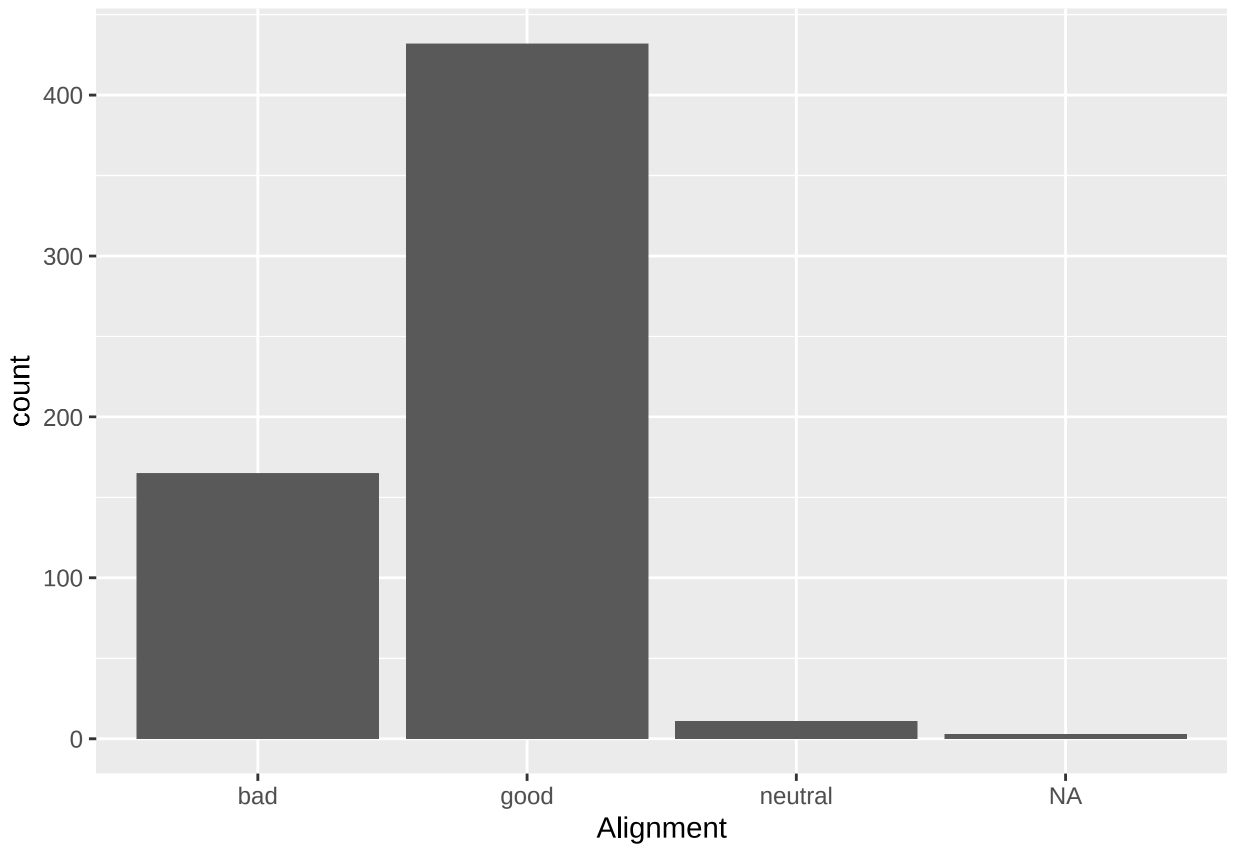
```
DF2 <- read_csv("https://raw.githubusercontent.com/Vincent-Toups/bios611-project1/master/source_data/da
```

Q2-1

```
summary(DF2)
```

```
##      Name             Alignment          Intelligence      Strength     
##  Length:611         Length:611         Min.   :  1.0   Min.   :  1.00  
##  Class :character   Class :character   1st Qu.:  1.0   1st Qu.:  1.00  
##  Mode  :character   Mode  :character   Median : 50.0   Median : 10.00  
##                                        Mean   : 44.5   Mean   : 29.08  
##                                        3rd Qu.: 75.0   3rd Qu.: 53.00  
##                                        Max.   :113.0   Max.   :100.00  
##      Speed          Durability         Power            Combat      
##  Min.   :  1.00   Min.   :  1.00   Min.   :  0.00   Min.   :  1.00  
##  1st Qu.:  1.00   1st Qu.:  1.00   1st Qu.:  0.00   1st Qu.:  1.00  
##  Median : 23.00   Median : 32.00   Median : 37.00   Median : 50.00  
##  Mean   : 27.31   Mean   : 41.84   Mean   : 40.31   Mean   : 43.21  
##  3rd Qu.: 42.00   3rd Qu.: 80.00   3rd Qu.: 67.00   3rd Qu.: 70.00  
##  Max.   :100.00   Max.   :120.00   Max.   :100.00   Max.   :101.00  
##      Total      
##  Min.   :  5.0  
##  1st Qu.:  5.0  
##  Median :255.0  
##  Mean   :226.3  
##  3rd Qu.:351.5  
##  Max.   :581.0
```

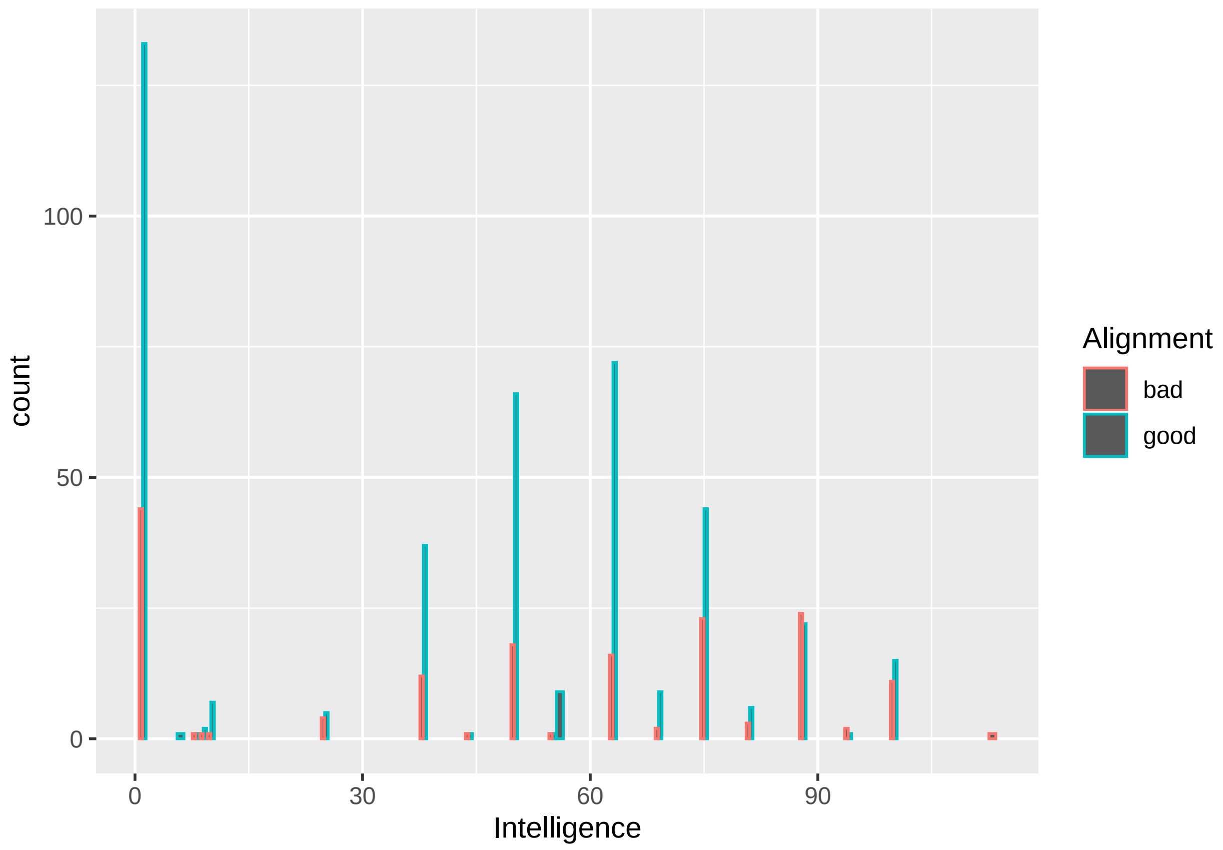
```
#nothing seems to out of the ordinary right
#here, lets make a couple plots

DF2 %>%
  ggplot(aes(x = Alignment)) +
  geom_bar()
```



```r
#there are a lot of goods and not too many neutrals.
#Also seems to be a couple NAs, before modeling we should remove the NAs

DF2 <- na.omit(DF2)

#lets take a look at how many nuetrals there are
nrow(DF2 %>% filter(Alignment == "neutral"))
```

```
## [1] 11
```

```r
#since there are only 11 neutrals and we are mostly
#concerned with good vs bad lets just take these neutral people out

DF2.no.neut <- DF2 %>%
  filter(Alignment != "neutral")

#lets check for some outliers here too
DF2.no.neut %>%
  ggplot(aes(x = Intelligence)) +
  geom_bar(aes(color = Alignment), position = "dodge")
```



```
#nothing really irregular here, seems like bad people
#are a bit more inteligent though which is interesting

#checked similar plots with all other qualities
#and did not find anything too out of the ordinary
```

I guess just taking out the 11 neutrals is all I am going to do in this section. This seems reasonable becasue there are not many of them and a person identified as "neutral" is not very interesting for the analysis.

Q2-2

```
pca <- prcomp(DF2.no.neut[, -c(1, 2)])
summary(pca)
```

```
## Importance of components:
##                              PC1      PC2      PC3      PC4      PC5
## Standard deviation     180.7021 24.80395 19.63123 16.03270 14.83526
## Proportion of Variance   0.9509  0.01792  0.01122  0.00749  0.00641
## Cumulative Proportion    0.9509  0.96877  0.97999  0.98748  0.99389
##                              PC6       PC7
## Standard deviation     14.48722 1.356e-13
## Proportion of Variance  0.00611 0.000e+00
## Cumulative Proportion   1.00000 1.000e+00
```

```
#we would just need the primary principle component
#since it accounts for 95% of the variance
round(pca$rot[,1],2)
```

```
## Intelligence     Strength        Speed   Durability        Power
##        -0.16        -0.15        -0.11        -0.18        -0.17
##       Combat        Total
##        -0.15        -0.92
```

```
#how the 1st principle component is made up...
#seems to be mostly made of the total column
```

Q2-3

Since the columns we are interested in, the numerical ones exept total, are already on the same scale from 0-100 we should not need to normalize anything here. We could normalize everything if we wanted to include the total column, but we see above that we should probably not include this column.

Q2-4

```
Total.Test = rowSums(DF2.no.neut[, -c(1, 2, 9)])
DF2.no.neut$Total == Total.Test #seems like this is true
```

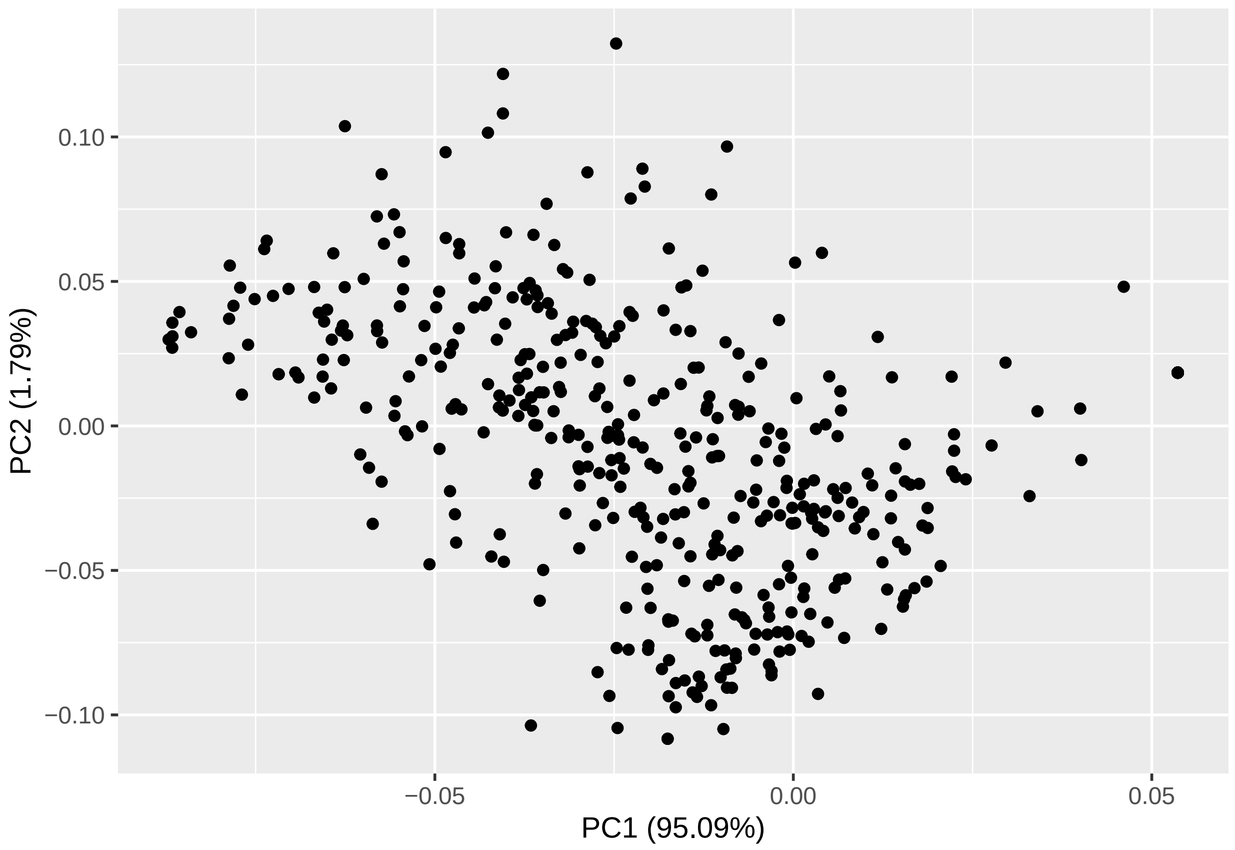
Yes it seems that the 'total' column is the total of the numeric columns

Q2-5

I don't think we should be using the 'total' column in the pca since it is made up of the other columns. You can see from the code in section Q2-3 that the primary component is mostly made up of the total column.

Q2-6

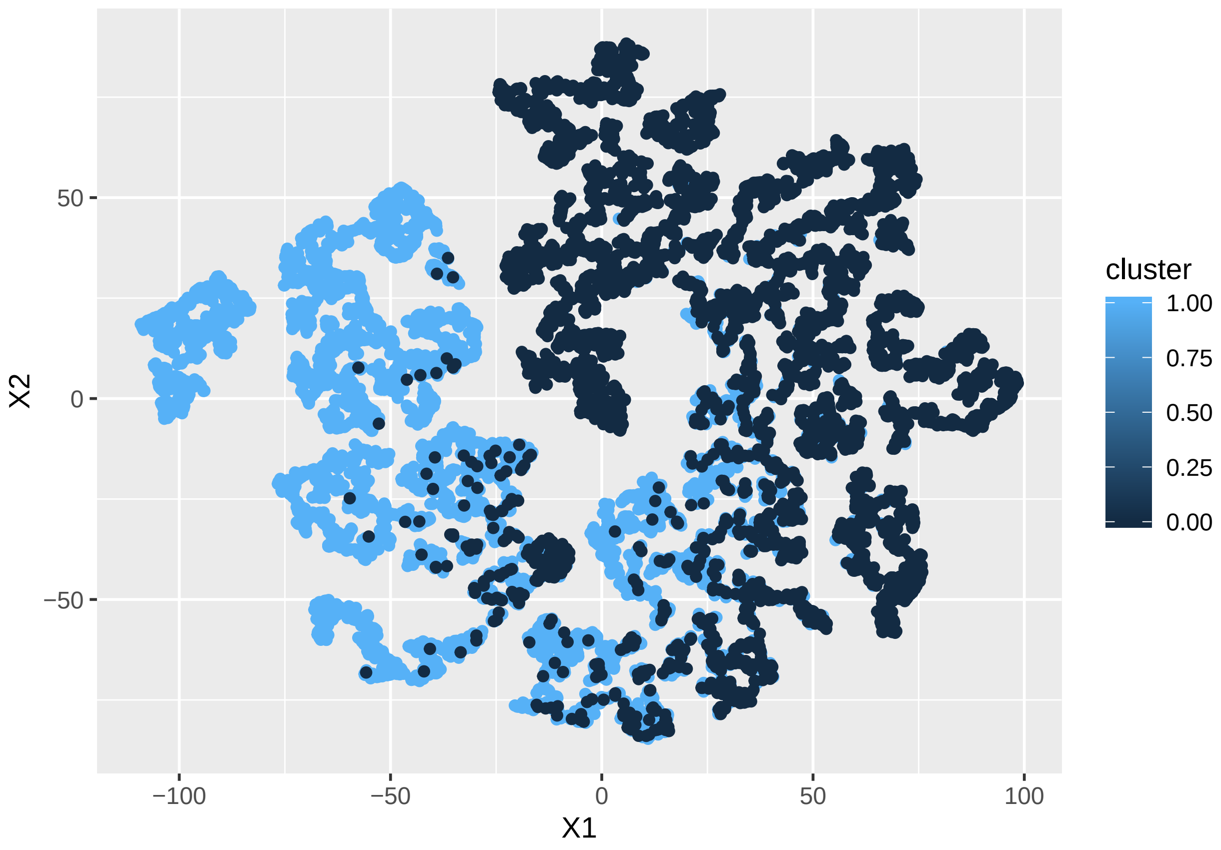
```
autoplot(pca)
```



Seems like a random distribution to me, I wouldn't say there are any insights to be made from this plot.

Q3

```r
DF.TSNE <- read.csv("TSNE.csv")
DF.TSNE %>%
  ggplot(aes(x = X1, y = X2)) +
  geom_point(aes(color = cluster))
```



Insights, seems like there might be two groups, we should do a pca to really check. There is a bit of overlap, but it looks like there could be two groups with some outliers (Small men, big women).

Q4

Done in Home-work-5.ipynb

Q5

```
DF2.no.neut$Alignment.bin <- ifelse(DF2.no.neut$Alignment == "good", 1, 0) #Good = 1, bad = 0

trainIndex <- createDataPartition(y = DF2.no.neut$Alignment.bin, p = .8, times = 1, list = F)
DF2.no.neut$Alignment.bin <- factor(DF2.no.neut$Alignment.bin)
train_ctrl <- trainControl(method = "repeatedcv", number = 50)
gbm2 <- train(Alignment.bin ~ Intelligence +
                Strength +
                Speed +
                Durability +
                Power +
                Combat +
                Total, data = DF2.no.neut %>% slice(trainIndex),
              method = "gbm",
              trControl = train_ctrl,
              verbose = F)
# summary(gbm2)
gbm2
```

```
## Stochastic Gradient Boosting
##
## 478 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (50 fold, repeated 1 times)
## Summary of sample sizes: 469, 468, 469, 468, 468, 468, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7291111  0.06808996
##   1                  100      0.7308889  0.08510682
##   1                  150      0.7142222  0.04168621
##   2                   50      0.7124444  0.05673257
##   2                  100      0.7126667  0.06505584
##   2                  150      0.7166667  0.09323929
##   3                   50      0.7182222  0.09180150
##   3                  100      0.7115556  0.08773270
##   3                  150      0.7031111  0.07621752
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

The Final Values used for the model are shown in the output above.

Q6

Using things such as K-fold cross validation leads to a much more complete analysis. By taking many different samples and testing each individual we are able see how the model performs regardless of the sample taken. If we simply take one single sample we will not see the entire picture. We will only see how the model performs on that very specific set. Using CV better shows how the model will perform on data that is not in the training set, which in the end is the point of modeling.

Q7

RFE works by starting with a full model and removes the least imporant features through fitting and refitting. RFE fits and test the full model and removes the feature that is the least important. It will then refit the model without this feature and test to remove another feature. The algorithm will continue like this until it finds the features that are most important.