6372: Unit 9 Homework

Matt Farrow

PCA Conceptual questions

1. TRUE/FALSE Principle component analysis is a predictive modeling technique such as linear regression and LDA.

   FALSE

2. TRUE or FALSE? Technically speaking, PCA should not be applied to categorical variables.

   TRUE

3. An analyst conducts a PCA on continuous variables 1 through 20 and settled on reducing the variables down to 4. The analyst then proceeds to conduct a linear regression using the 4 PC components as predictors and the response is variable 1. Why is this a horrible idea?

Because variable 1 is also included in the PC components, it cannot then be used as a response variable to the linear regression.

4. Why is it important to conduct PCA on standardized variables (aka using the correlation matrix)?

Standardizing variables makes the variables comparable and is especially important when the variables are in different scales.

Exercise #1 PCA Basics

The example conducted in class did not do a very good job of illustrating the interpretation goals of PCA. For this reason, we will switch to a baseball data set to go over the basics and play around with interpretation. The baseball data set is located in the Lahman package. The data set is quite comprehensive having baseball player statistics dating back to 1871. We are going to examine the earliest year, 2016, by itself. Lets take a quick summary to see what variables we have.

```r
library(Lahman)

# Get data from Batting data set for 2016
bat_16 <- tibble(Batting %>%
  filter(yearID == 2016))

# Look at the summary statistics
summary(bat_16)
```

```
##    playerID             yearID         stint          teamID      lgID
##  Length:1483        Min.   :2016   Min.   :1.000   ATL    :  60   AA:  0
##  Class :character   1st Qu.:2016   1st Qu.:1.000   SDN    :  58   AL:734
##  Mode  :character   Median :2016   Median :1.000   LAN    :  55   FL:  0
##                     Mean   :2016   Mean   :1.092   PIT    :  55   NA:  0
##                     3rd Qu.:2016   3rd Qu.:1.000   SEA    :  54   NL:749
##                     Max.   :2016   Max.   :4.000   LAA    :  53   PL:  0
##                                                    (Other):1148   UA:  0
##        G                AB              R                H
##  Min.   :  1.00   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 13.00   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:  0.00
##  Median : 31.00   Median : 11.0   Median :  1.00   Median :  1.00
##  Mean   : 47.51   Mean   :111.6   Mean   : 14.66   Mean   : 28.51
##  3rd Qu.: 69.00   3rd Qu.:155.0   3rd Qu.: 18.00   3rd Qu.: 36.00
##  Max.   :162.00   Max.   :672.0   Max.   :123.00   Max.   :216.00
##
##       X2B              X3B               HR              RBI
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000   Min.   :  0.00
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:  0.00
##  Median : 0.000   Median : 0.0000   Median : 0.000   Median :  0.00
##  Mean   : 5.566   Mean   : 0.5887   Mean   : 3.783   Mean   : 13.99
##  3rd Qu.: 7.000   3rd Qu.: 0.0000   3rd Qu.: 3.000   3rd Qu.: 15.50
##  Max.   :48.000   Max.   :11.0000   Max.   :47.000   Max.   :133.00
##
##        SB               CS               BB               SO
##  Min.   : 0.000   Min.   : 0.000   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:  0.00   1st Qu.:  0.00
##  Median : 0.000   Median : 0.000   Median :  0.00   Median :  4.00
##  Mean   : 1.711   Mean   : 0.675   Mean   : 10.17   Mean   : 26.29
##  3rd Qu.: 1.000   3rd Qu.: 0.000   3rd Qu.: 13.00   3rd Qu.: 38.00
##  Max.   :62.000   Max.   :18.000   Max.   :116.00   Max.   :219.00
##
##       IBB               HBP               SH                SF
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.6285   Mean   : 1.113   Mean   : 0.6912   Mean   : 0.8186
##  3rd Qu.: 0.0000   3rd Qu.: 1.000   3rd Qu.: 1.0000   3rd Qu.: 1.0000
##  Max.   :20.0000   Max.   :24.000   Max.   :13.0000   Max.   :15.0000
##
##       GIDP
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 2.508
##  3rd Qu.: 3.000
##  Max.   :26.000
##
```

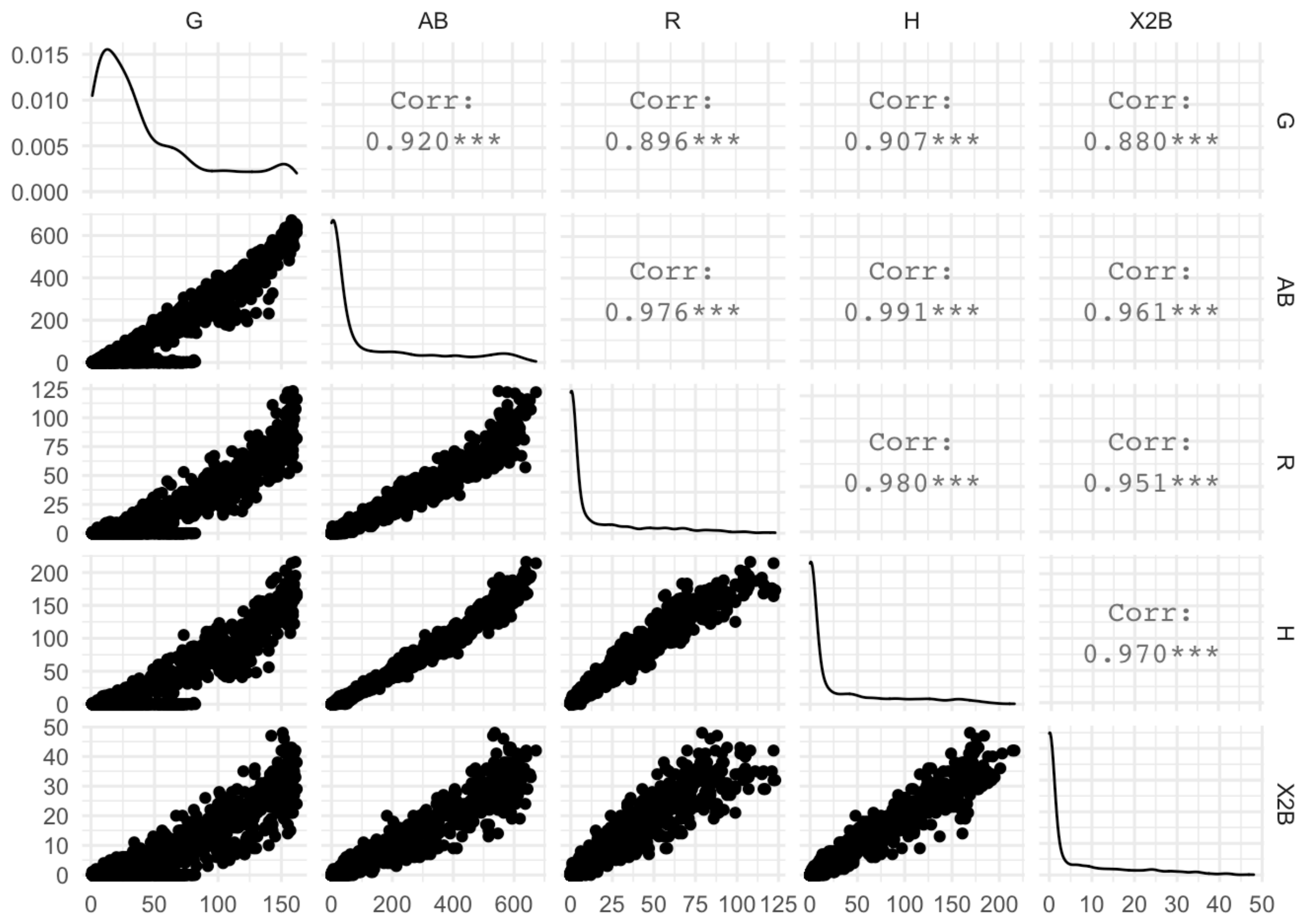
For those of you who do not know too much about baseball, the first 5 variables are just general information, G is the number of games played while the rest are information for players batting ability. G is games, AB is number of batting attempts, R-Runs, H-hits, X2B and X3B are doubles and triples, HR-home runs, RBI-Runs Batted In. These are all general stats on how well the batters can hit the ball. SB (stolen bases) and CS (caught stealing) are statistics about a players ability to run the bases. BB and IBB are when the batter gets a walk. The other stats are HBP (hit by a pitch), SH (sacrifice hit) and SF (sacrifice fly), and GIDP (grounded into double play).

Sports data sets lend themselves well to PCA. We will use this example to go through similar concepts discussed in class. For starters let's start off with just a few variables in the set to verify PCA is doing what we expect it to. Here is a quick scatter plot matrix. The variables here are highly correlated with each other.

```r
# Create a reduced data set that contains only columns 6:10
reduced <- bat_16 %>%
  select(6:10)

# Create a pairs plot
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
reduced %>%
  ggpairs() +
  theme_minimal()
```



Let's take a quick look at the summary statistics and in particular let's calculate the variance of each variable and add them up to obtain the total variance.

```
# Summary statistics by variable
apply(reduced, 2, summary)
```

```
##                  G       AB         R         H       X2B
## Min.       1.00000   0.0000   0.00000   0.00000  0.000000
## 1st Qu.   13.00000   0.0000   0.00000   0.00000  0.000000
## Median    31.00000  11.0000   1.00000   1.00000  0.000000
## Mean      47.50573 111.6392  14.66217  28.50708  5.565745
## 3rd Qu.   69.00000 155.0000  18.00000  36.00000  7.000000
## Max.     162.00000 672.0000 123.00000 216.00000 48.000000
```

```
# Total variance of each variable
apply(reduced, 2, var)
```

```
##           G          AB           R           H         X2B
##  2054.72517 31658.76653   665.77851  2363.88035    96.32951
```

```
# Total variance
sum(apply(reduced, 2, var))
```

```
## [1] 36839.48
```

We have been talking about the covariance matrix a lot lately. An estimate for any given set of continuous variables can be obtained using the `cov` function. You can see that the diagonals of this matrix are the same as the variances calculated one at a time from before.

```
# Create a covariance matrix
cov(reduced)
```

```
##             G        AB         R         H         X2B
## G   2054.7252  7423.272 1048.0832 1999.3891  391.50924
## AB  7423.2723 31658.767 4479.3423 8574.1257 1678.82164
## R   1048.0832  4479.342  665.7785 1229.5547  240.89233
## H   1999.3891  8574.126 1229.5547 2363.8803  462.93830
## X2B  391.5092  1678.822  240.8923  462.9383   96.32951
```

```
# Another way to get total variance
sum(diag(cov(reduced)))
```

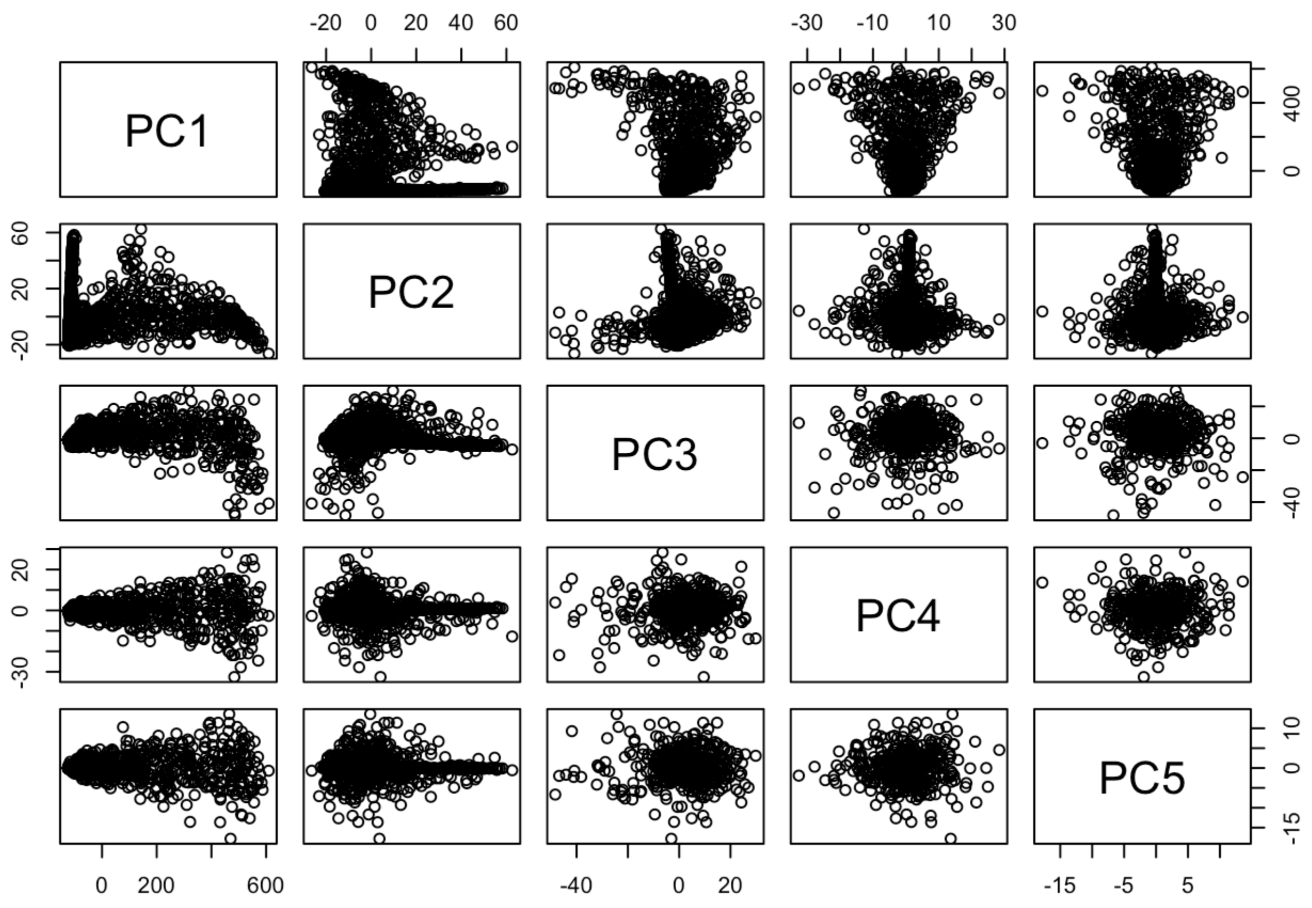
```
## [1] 36839.48
```

Running PCA is relatively straightforward. The following script conducts a PCA using the covariance matrix (nonstandardized variables) and stores the results in an object. This object contains the eigenvectors, eigenvalue, and the new principle component vectors. Lets start by producing a correlation matrix to verify that new principle component variables are uncorrelated.

```
# Run principle component analysis on reduced data
pc_result <- prcomp(reduced, scale. = FALSE)

# View results of the PCA
pc_result
```

```
## Standard deviations (1, .., p=5):
## [1] 190.951787  17.361212   7.078405   4.468044   2.327266
##
## Rotation (n x k) = (5 x 5):
##            PC1         PC2         PC3         PC4          PC5
## G   0.22024127  0.97397742 -0.05059877  0.01730636 -0.001420923
## AB  0.93156585 -0.19396824  0.30281622 -0.05283321  0.008507431
## R   0.13202633 -0.04546002 -0.57626905 -0.80501489  0.019168837
## H   0.25258860 -0.10598044 -0.73885937  0.57082967 -0.230667442
## X2B 0.04946609 -0.02111491 -0.16656204  0.15170255  0.972805577
```

```
# Get scores
pc_scores <- pc_result$x

# Pairs plots of PC Scores
pairs(pc_scores)
```



```
# Correlation of PC Scores
cor(pc_scores)
```

```
##               PC1           PC2           PC3           PC4           PC5
## PC1  1.000000e+00  3.883426e-15 -9.277437e-15  1.552181e-14 -1.433233e-14
## PC2  3.883426e-15  1.000000e+00 -7.001366e-17  5.291058e-15 -5.983462e-15
## PC3 -9.277437e-15 -7.001366e-17  1.000000e+00 -1.141962e-15 -6.650849e-16
## PC4  1.552181e-14  5.291058e-15 -1.141962e-15  1.000000e+00  3.636272e-16
## PC5 -1.433233e-14 -5.983462e-15 -6.650849e-16  3.636272e-16  1.000000e+00
```

We can again verify that the total variance in the new PC variables is exactly the same as the original data. The eigenvectors are stored inside of “pc_result” as well in the “rotation” object.

```
var_pca <- apply(pc_scores, 2, var)
var_pca
```

```
##          PC1          PC2          PC3          PC4          PC5
## 36462.584984   301.411677    50.103822    19.963419     5.416166
```

```
# Total Variance of PC's
sum(var_pca)
```

```
## [1] 36839.48
```

```
# Total Variance of Original Variables
sum(apply(reduced, 2, var))
```

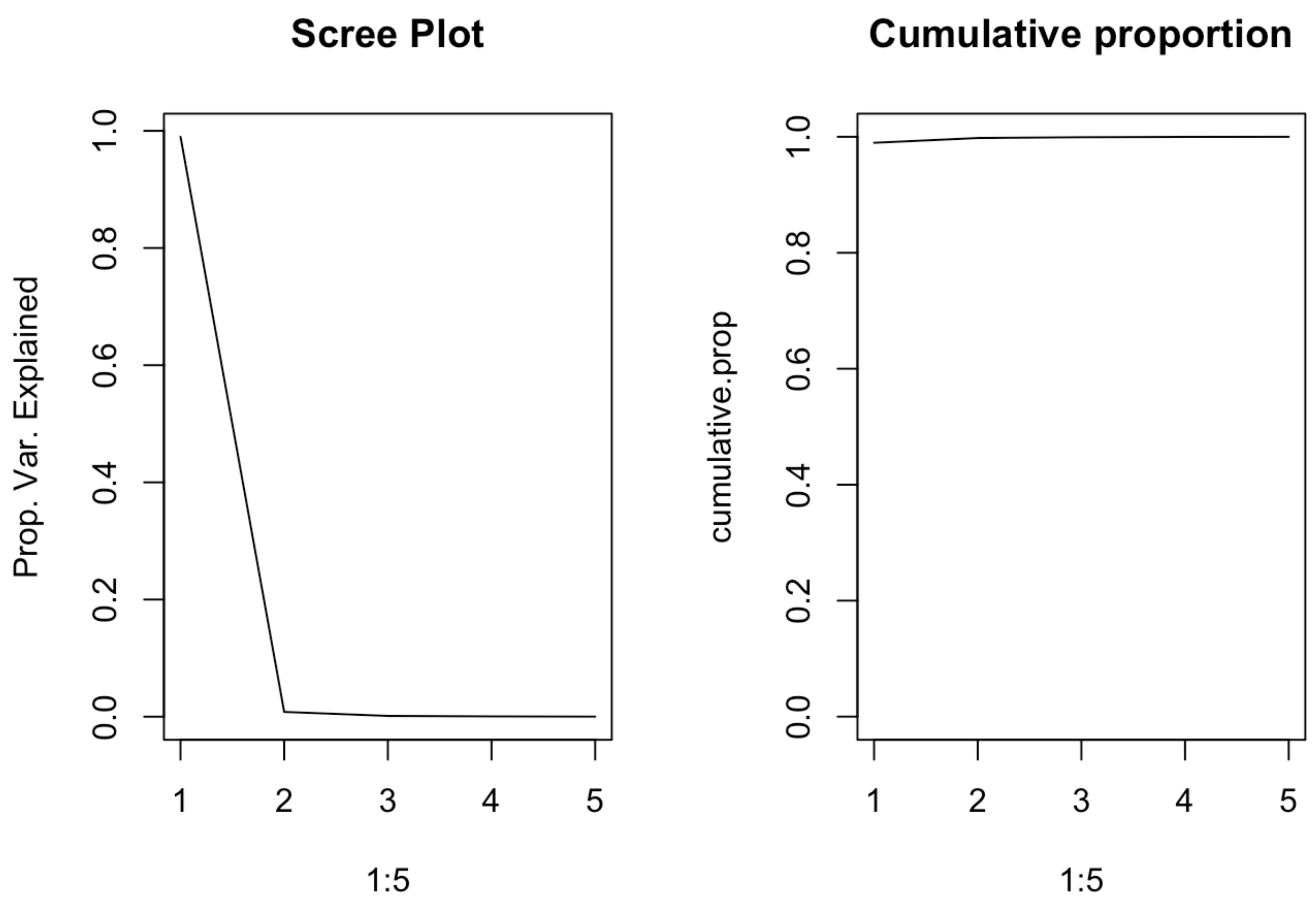
```
## [1] 36839.48
```

```
# List of eigenvectors
pc_result$rotation
```

```
##            PC1         PC2         PC3         PC4          PC5
## G   0.22024127  0.97397742 -0.05059877  0.01730636 -0.001420923
## AB  0.93156585 -0.19396824  0.30281622 -0.05283321  0.008507431
## R   0.13202633 -0.04546002 -0.57626905 -0.80501489  0.019168837
## H   0.25258860 -0.10598044 -0.73885937  0.57082967 -0.230667442
## X2B 0.04946609 -0.02111491 -0.16656204  0.15170255  0.972805577
```

A scree plot of the eigenvalues used to determine how many PC’s to keep can be plotted in the following way:

```
par(mfrow = c(1, 2))
eigenvals <- (pc_result$sdev)^2
plot(1:5, eigenvals / sum(eigenvals), type = "l", main = "Scree Plot", ylab = "Prop.
Var. Explained")
cumulative.prop <- cumsum(eigenvals / sum(eigenvals))
plot(1:5, cumulative.prop, type = "l", main = "Cumulative proportion", ylim = c(0, 1)
)
```



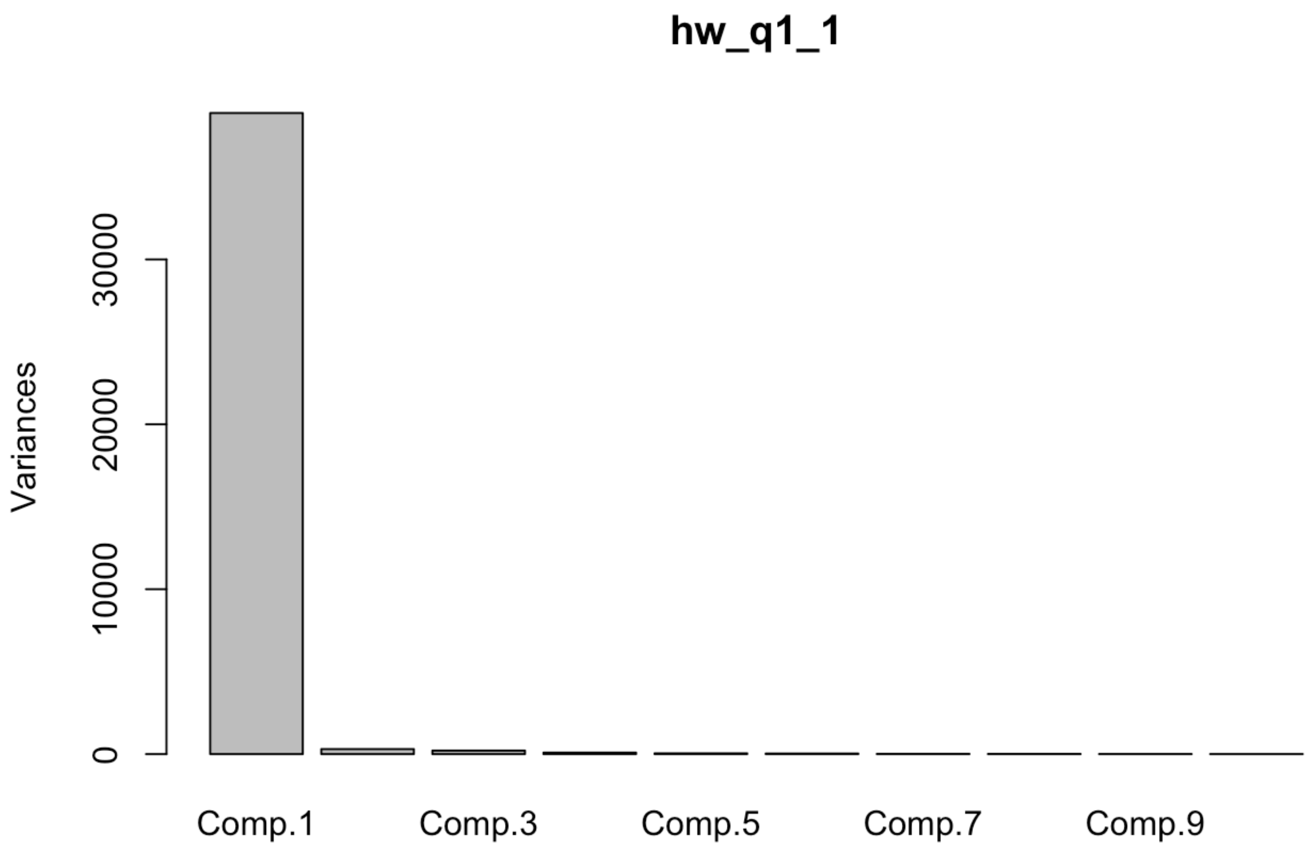
```
par(mfrow = c(1, 1))
```

Since all of the variables are not on the same scale, we see a very similar phenomenon that we discussed in the pre-live session. To conduct the PCA on the correlation matrix, just set `scale. = TRUE` inside of the `prcomp` function.

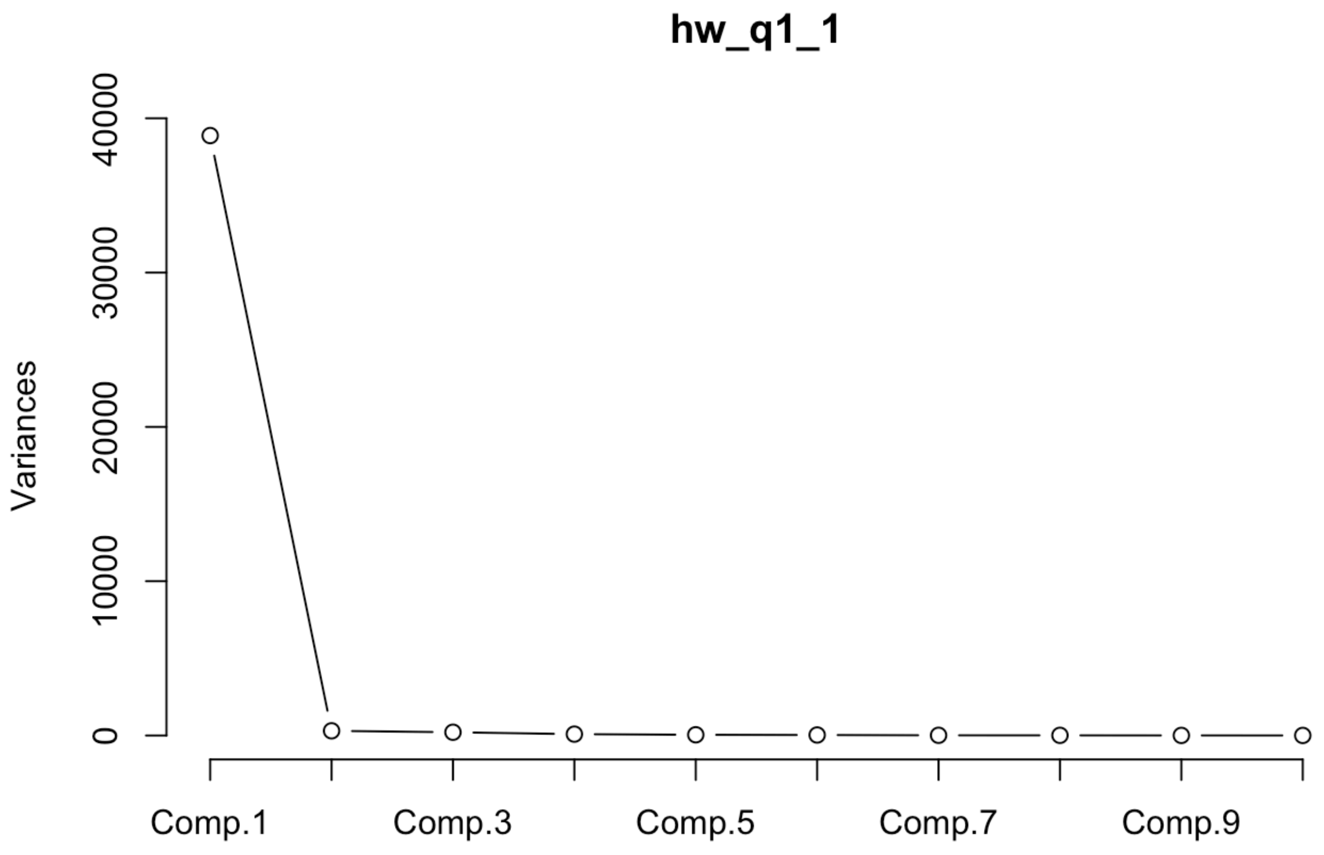
HW Assignment #1

1. Conduct the PCA analysis but use the entire set of variables starting with column 6, the Games played variable, all the way down to the end at GIDP. Provide a scree plot and determine the amount of PC’s needed to retain approximately 90% of the total variation in the data set.

```
# Run PCA
hw_q1_1 <- princomp(bat_16[, 6:22], scores = TRUE)

# Scree Plot
screeplot(hw_q1_1)                   # bars
```



```
screeplot(hw_q1_1, type = "lines")  # lines
```



It looks as though only one PC is needed to retain approximately 90% of the total variation in the data set.

2. Provide the eigenvector matrix and examine the loading (coefficients) that determine the linear combinations of each principle component. Verify that PC1 is essentially a weighted average of all the variables together (minus the SH, sacrifice hit variable.)

```
# Summary Stats
# summary(hw_q1_1)
hw_q1_1$loadings
```

```
##
## Loadings:
##      Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
## G     0.213  0.963  0.152
## AB    0.902 -0.202  0.114 -0.248        -0.215 -0.147
## R     0.128                0.392 -0.250  0.440 -0.372  0.571 -0.136  0.162
## H     0.244 -0.127  0.284  0.256         0.512  0.623 -0.126  0.322
## X2B                                      0.100  0.286        -0.889  0.125
## X3B                                                                 -0.119
## HR                 -0.111  0.244  0.244        -0.194  0.197  0.227
## RBI   0.124                0.586  0.609 -0.104 -0.186 -0.321
## SB                                -0.286  0.401 -0.472 -0.673
## CS                                                    -0.119
## BB                 -0.179  0.527 -0.635 -0.469  0.144 -0.145
## SO    0.195        -0.910 -0.131         0.249  0.191
## IBB                                                                 -0.102
## HBP                                                                 -0.499
## SH
## SF
## GIDP                                    -0.130                0.153  0.816
##      Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17
## G
## AB
## R    -0.200                  -0.101
## H
## X2B   0.224  -0.122   0.127
## X3B  -0.116   0.155          -0.126  -0.474  -0.821   0.160
## HR    0.703  -0.236   0.355   0.137          -0.201
## RBI  -0.259   0.122  -0.109
## SB    0.127  -0.136                   0.114           0.155
## CS                            0.106  -0.557   0.173  -0.778
## BB
## SO
## IBB                   0.377  -0.890           0.168
## HBP  -0.341  -0.779           0.119
## SH   -0.354   0.250   0.831   0.329
## SF   -0.146                  -0.114   0.647  -0.451  -0.579
## GIDP -0.216  -0.443                  -0.148  -0.101
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.059  0.059  0.059  0.059  0.059  0.059  0.059  0.059  0.059
## Cumulative Var  0.059  0.118  0.176  0.235  0.294  0.353  0.412  0.471  0.529
##                Comp.10 Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17
## SS loadings      1.000   1.000   1.000   1.000   1.000   1.000   1.000   1.000
## Proportion Var   0.059   0.059   0.059   0.059   0.059   0.059   0.059   0.059
## Cumulative Var   0.588   0.647   0.706   0.765   0.824   0.882   0.941   1.000
```

```
# hw_q1_1$scores
```

3. Verify that PC2 has big negative loadings on triples (X3B), stolen bases (SB), caught stealing (CS), and sacrifice hits (SH). This variable could be interpreted to be a general indication of a players speed or general utility since all of the variables require situation awareness and running ability. (You don’t need to provide an answer here, just verify))

PCA as an Exploratory Technique for Classification

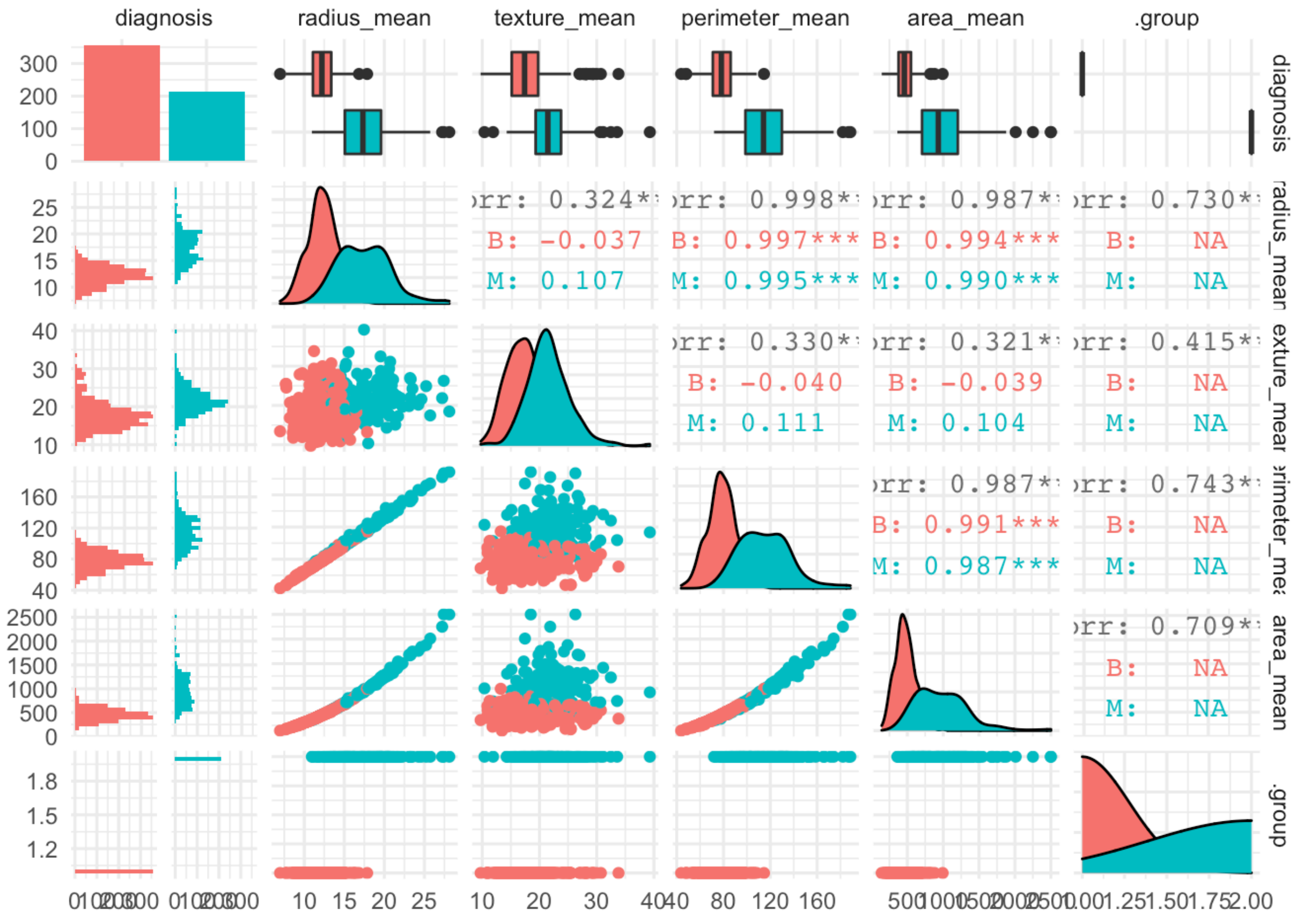
This exercise is designed to walk you through how PCA can be used as an informative unsupervised analysis of your predictors to get a high level view of whether the predictors are actually going to do a good job or not before a predictive model for categorical responses is even applied.

The following data set is a breast cancer data set that has numerous measurements taken from tumor biopsies. The goal of using this data set is to predict using the metrics alone if the biopsy is cancer or not. When continuous variables are available it is often helpful to create a pairs plot of data color coded by the response status (Diagnosis). The first variable is an id number and is not needed.

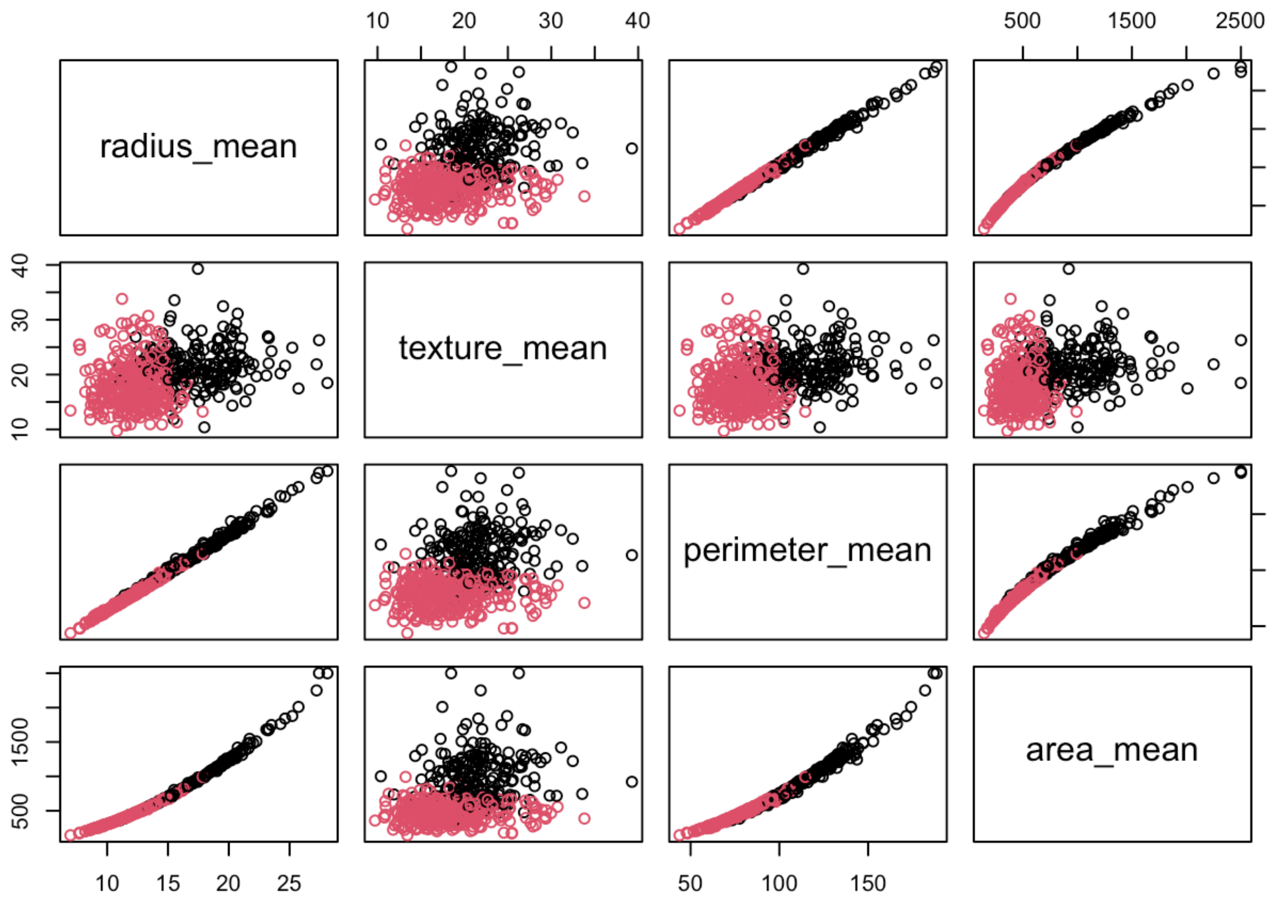
```
bc <-
 read.table(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
wdbc.data",
  header = F,
  sep = ","
 )
names(bc) <- c(
 "id_number",
 "diagnosis",
 "radius_mean",
 "texture_mean",
 "perimeter_mean",
 "area_mean",
 "smoothness_mean",
 "compactness_mean",
 "concavity_mean",
 "concave_points_mean",
 "symmetry_mean",
 "fractal_dimension_mean",
 "radius_se",
 "texture_se",
 "perimeter_se",
 "area_se",
 "smoothness_se",
 "compactness_se",
 "concavity_se",
 "concave_points_se",
 "symmetry_se",
 "fractal_dimension_se",
 "radius_worst",
 "texture_worst",
 "perimeter_worst",
 "area_worst",
 "smoothness_worst",
 "compactness_worst",
 "concavity_worst",
 "concave_points_worst",
 "symmetry_worst",
 "fractal_dimension_worst"
)

# Getting a look at the distribution
table(bc$diagnosis)
```

```
##
##   B   M
## 357 212
```

```
# Scatter plots color coded by response for just the first few variables
bc %>%
  select(2:6) %>%
  group_by(diagnosis) %>%
  ggpairs(aes(color = diagnosis)) +
  theme_minimal()
```



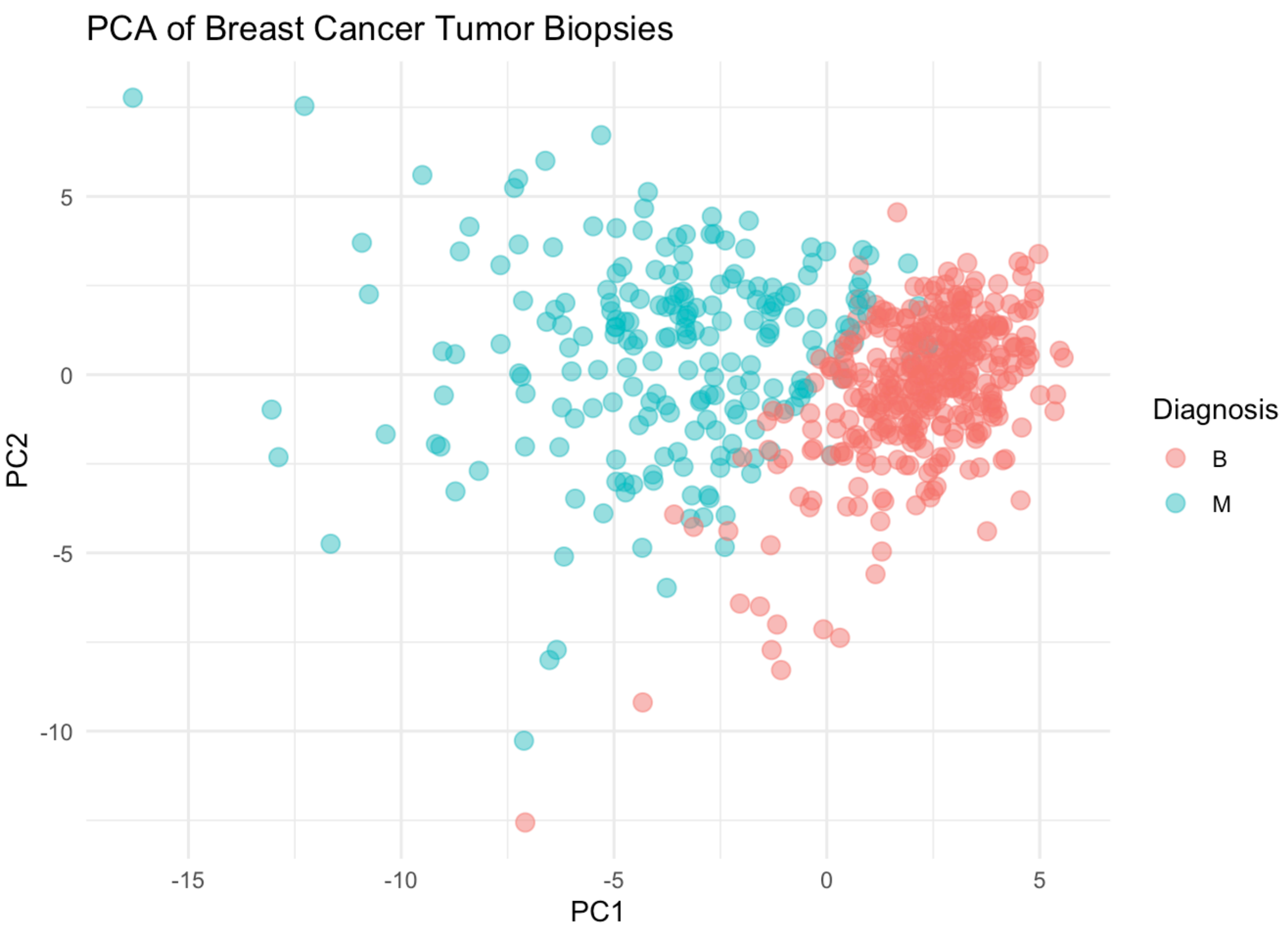
```
pairs(bc[, 3:6], col = as_factor(bc$diagnosis))
```



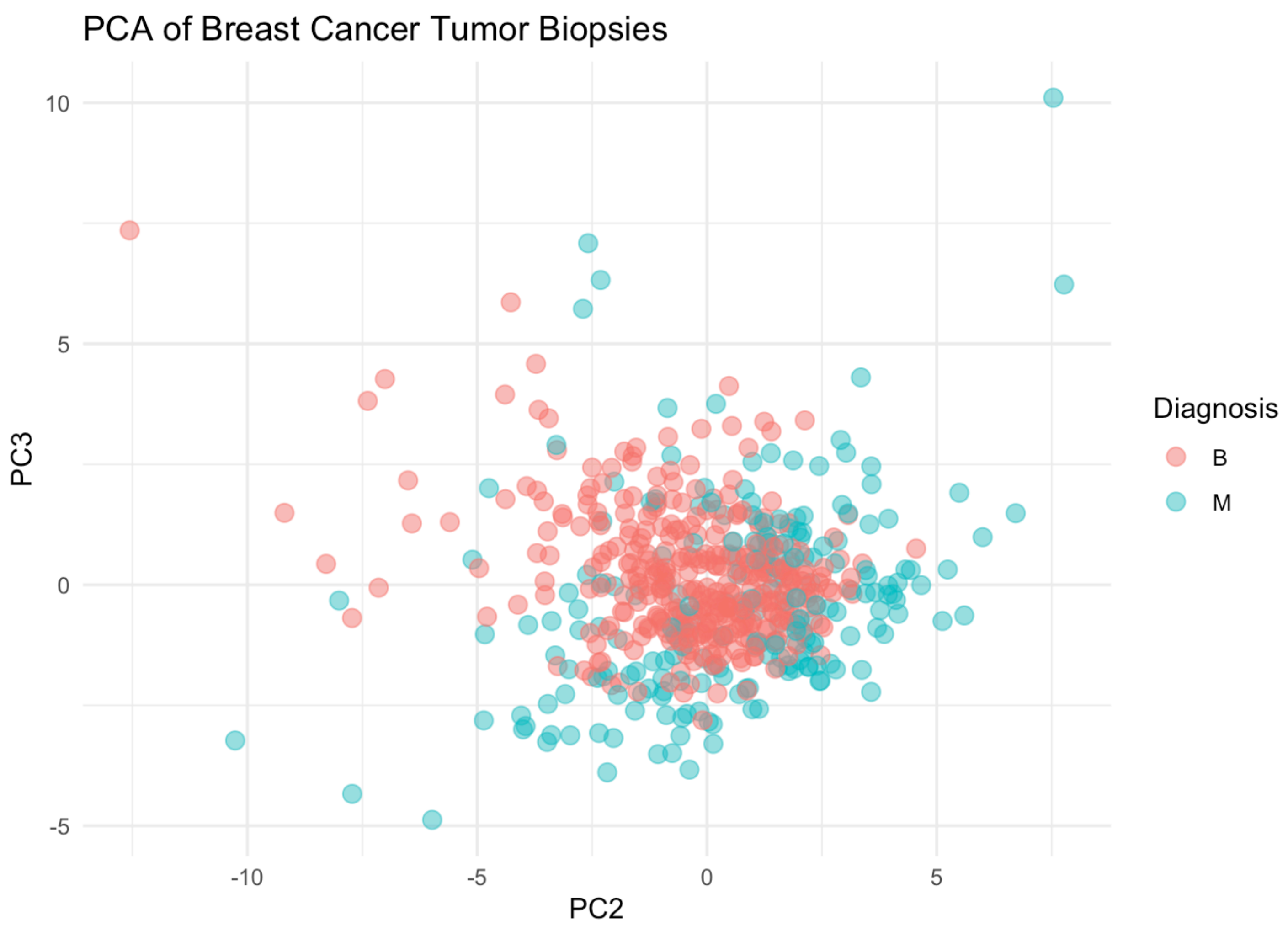
So we can see from this pairs plot of just the first few variables, separation between the cancer and non cancer groups are pretty well separated. Unfortunately we may not always see clear separations but that does not necessarily mean that something like LDA or some other predictive tool won't work. It could be due to the fact we can't see the separation of the groups unless we can actually see in higher dimensions. One way to still get at this is to conduct a PCA analysis and provide a some scatter plots for the first few PC's. If separation exists in the PC's, then a predictive model will probably do well.

Below we will conduct PCA on all of the predictors and plot the first few PC's against each other and look for separation. The number of PCs to explore can be dictated by the scree plot.

```
pc_bc <- prcomp(bc[,-c(1, 2)], scale. = TRUE)
pc_bc_scores <- pc_bc$x

# Adding the response column to the PC's data frame
pc_bc_scores <- as_tibble(pc_bc_scores)
pc_bc_scores$Diagnosis <- bc$diagnosis

# Use ggplot2 to plot the first few PC's
ggplot(data = pc_bc_scores, aes(x = PC1, y = PC2)) +
 geom_point(aes(col = Diagnosis), size = 3, alpha = 0.5) +
 labs(title = "PCA of Breast Cancer Tumor Biopsies") +
  theme_minimal()+
  NULL
```



```
ggplot(data = pc_bc_scores, aes(x = PC2, y = PC3)) +
  geom_point(aes(col = Diagnosis), size = 3, alpha = 0.5) +
 labs(title = "PCA of Breast Cancer Tumor Biopsies") +
  theme_minimal() +
  NULL
```



So we can see in the first graphic a clear separation exists for the two cancer groups. So the PCA is telling us in effect what we already know from looking at the original variables. The power of this approach is that you only need to look at 2-4 graphs each time, versus potentially having to examine massive scatter plot matrices to see if anything is there or not!

HW Assignment 2

1. Given what we see in the PCA analysis, it's not too surprising that an LDA will probably do a good job here in predicting the categorical responses. Perform an LDA on the original set of variables and calculate a confusion matrix. Note: For this problem you do not have to do a training and test set split, let's recognize that the prediction performance that we obtain is potentially biased too low due to overfitting. The main point here is that the accuracy is pretty good as expected via the PCA look.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```
# Run LDA
bc_lda <- lda(diagnosis ~ ., data = bc[, -1], CV = TRUE)

# Create confusion matrix
tab <- table(bc$diagnosis, bc_lda$class)
conCV1 <- rbind(tab[1, ]/sum(tab[1, ]), tab[2, ]/sum(tab[2, ]))
# dimnames(conCV1) <- list(Actual = c("No", "Yes"), "Predicted (cv)" = c("No", + "Yes
"))
print(round(conCV1, 3))
```

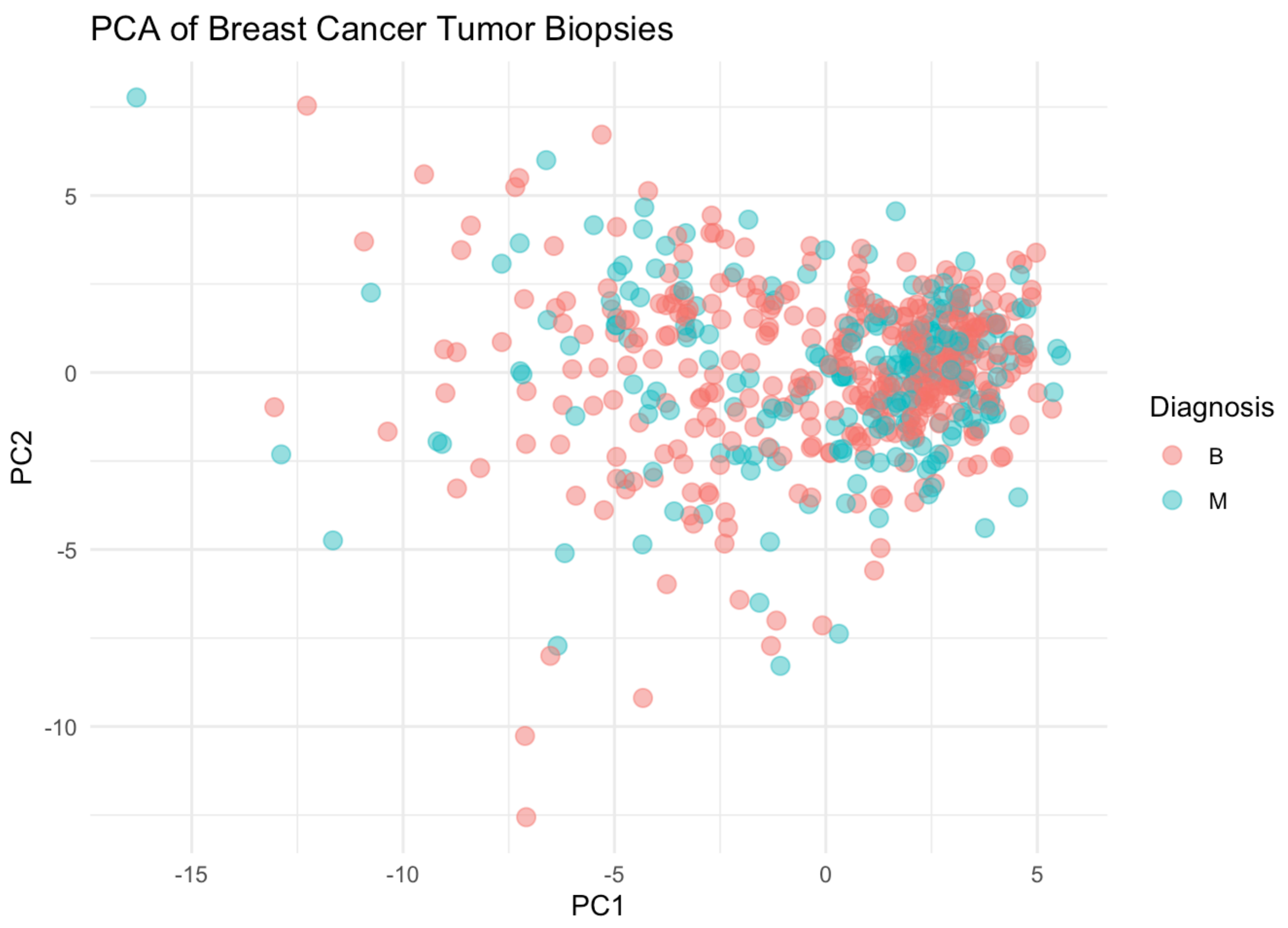
```
##          B     M
## [1,] 0.994 0.006
## [2,] 0.104 0.896
```

2. Consider now another great sanity check when building predictive models. The code below takes the original data set and randomly scrambles the response variable. This effectively breaks up any relationship that existed between the predictors and the response.

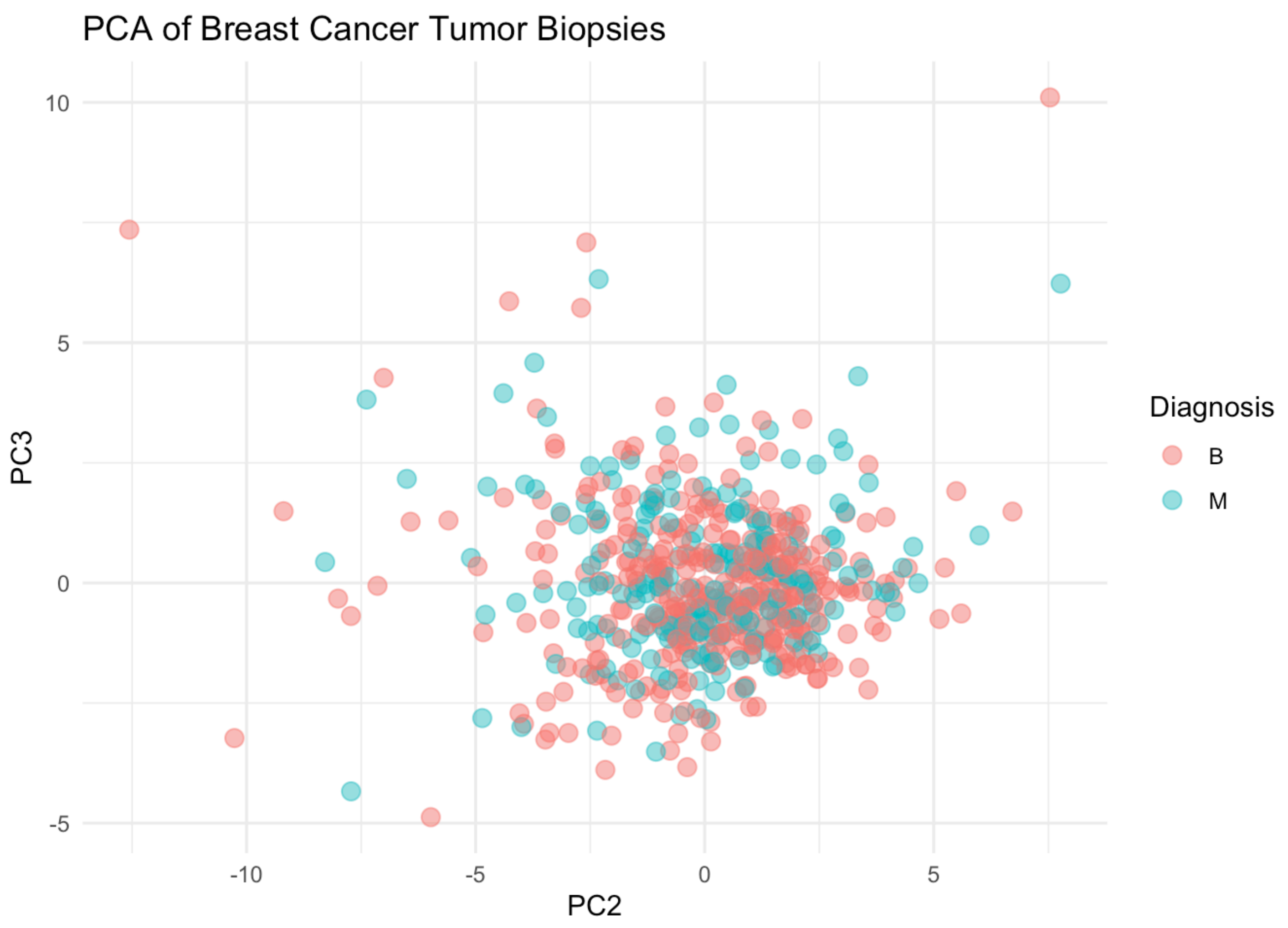
```
fake <- bc
fake$diagnosis <- sample(fake$diagnosis, 569, replace = F)
```

a. Plot PC1 and PC2 using the scrambled data set.

```
# Run PCA
pc_fake <- prcomp(fake[,-c(1, 2)], scale. = TRUE)
pc_fake_scores <- pc_fake$x

# Adding the response column to the PC's data frame
pc_fake_scores <- as_tibble(pc_fake_scores)
pc_fake_scores$Diagnosis <- fake$diagnosis

# Use ggplot2 to plot the first few PC's
ggplot(data = pc_fake_scores, aes(x = PC1, y = PC2)) +
 geom_point(aes(col = Diagnosis), size = 3, alpha = 0.5) +
 labs(title = "PCA of Breast Cancer Tumor Biopsies") +
  theme_minimal()+
  NULL
```



```
ggplot(data = pc_fake_scores, aes(x = PC2, y = PC3)) +
  geom_point(aes(col = Diagnosis), size = 3, alpha = 0.5) +
 labs(title = "PCA of Breast Cancer Tumor Biopsies") +
  theme_minimal() +
  NULL
```



b. Perform an LDA with this data set and look at the confusion matrix. Do they correspond?

Note: This little trick is extremely helpful when you are predicting a response that is heavily imbalanced (ex: lots of Cancer obs, few Healthy ones). LDA and other algorithms can behave quite weirdly in extreme cases and prediction performances can look good all the time. By conducting a separate analysis on scrambled data, if the prediction performance still looks good, you've recognized a problem. We can discuss this topic more as we get closer to finishing up Project 2.

```r
library(MASS)

# Run LDA
fake_lda <- lda(diagnosis ~ ., data = fake[, -1], CV = TRUE)

# Create confusion matrix
tab <- table(fake$diagnosis, fake_lda$class)
conCV1 <- rbind(tab[1, ]/sum(tab[1, ]), tab[2, ]/sum(tab[2, ]))
# dimnames(conCV1) <- list(Actual = c("No", "Yes"), "Predicted (cv)" = c("No", + "Yes
"))
print(round(conCV1, 3))
```

```
##           B     M
## [1,] 0.891 0.109
## [2,] 0.849 0.151
```