Final Project 3106

Since its birth in the nineteenth century, baseball has become one of America's favorite pastimes. Starting as an offshoot of popular British game cricket in the 18th century, baseball has rapidly exploded into one of the most financially lucrative entertainment ventures, with the average player in the Major League Baseball organization pulling in an average salary of $4.38 million. In fact, in 2017, the Major League Baseball Association's revenue hit an all-time high, exceeding 10 billion dollars for the first time! Baseball's popularity is proven just through the sheer amount of money that goes into advertising, merchandise, training, and selling tickets for games, and therefore it only makes sense to use statistics to further our knowledge of the sport itself.

A full season of major league baseball is made up of one-hundred and sixty two games. There are thirty major league baseball teams in total, fifteen of which are in the National League and the other fifteen are in the American League. Within each league, there are three divisions titled the East, West, and Central Divisions. Five teams make up each division based on their relative location within the United States. The World Series is a playoff tournament between the teams who have finished first in their respective divisions. One can earn a "wild card" into the world series by being the "best" second place team in their league, meaning that they have the next best winning record of a team that did not initially make the world series, which is a binary response variable in our data set.

For our data, I want to use the the "History of Baseball" set from Kaggle, which compiled baseball statistics from the official Major League Baseball database. It contains the MLB's complete batting and pitching statistics from 1871 to 2015, plus fielding

statistics, standings, team stats, park stats, player demographics, managerial records, awards, post-season data, and more. More specifically, I plan on using using the csv file named “teams” from this database, which contains the performance of individual teams from every season. The World Series is the annual championship series of the Major League in Baseball (MLB), played between one qualifying champion teams of the American League (AL) and the National League (NL).

The unit of analysis is the performance of a single team during one season - the data contains the records of every team since 1871-2015.

The goal of my analysis is to analyze baseball team’s performance over a given season, and use the metrics of a team’s offensive and defensive skills in order to determine a team’s likelihood of winning the world series during a given season. One part of my analysis will include Principal Component Analysis, with which I will attempt to determine whether the given statistics of a team during a season (such as hits scored, hits allowed, home runs, etc.) can be grouped into different factors or principal components (perhaps one such factor would represent a team’s performance while they’re at bat or when they’re defending). Principal Component Analysis attempts to Once I have a set of principal components and factors, I will attempt to use PCA + OLS in order to predict whether or not a team will win the world series, I also want to use time series analysis in order to determine whether particular statistics have become more or less important over time, and whether teams have been steadily improving in all statistics throughout the course of the Major League Baseball organization. Afterward, I use Naïve Bayes, Lasso, and Logistic

regression as classifiers, which attempt to "classify" a team as either a World Series Winner or Loser using a number of different criteria and predictors.

Some individuals who would be interested in the analysis done here would-be owners of baseball teams, and their coaching staff, since knowing whether or no particular types of skills or statistics are more important to focus on in order to maximize their chances of winning the World Series. For example, if the PCA tells us that "offensive" stats like hits scored, home runs, and runs scored mattered more than "defensive stats" like fielding percentages or hits allowed, maybe a team owner would invest more in training the offensive part of their team, and divert coaching attention to their team's offensive performance if that's where they're lagging behind. In the context of our principal components analysis, one particular important principal component could be a grouping of skills and stats that are especially important to focus on to maximize your chances of winning the World Series.

One obvious pattern I would expect to see within my data is that generally better performance increases a team's chances of winning the World Series – obviously, a team that's going to win the World Series will have generally excellent performance, but I'm more interested in seeing if there are particular skills or groups of sills that have a larger influence on whether or not a team will win the world series. Our dataset is potentially at risk for data-dredging, which is when data is either mis-analyzed or manipulated in order to present bogus statistically-significant results, or when only statistically-significant results are displayed in order to present a particular narrative. Since the proportion of World Series winners is so small compared to the rest of the dataset, wherever random sampling is involved, there's potential to get massively different results from performing

the same statistical tests several times. For example, in splitting our test and train data, it's possible to have World Series Winners be massively over-represented in our test dataset, which could potentially lead to our classifier algorithms seeming much more powerful than they actually are. In order to prevent this, I make sure the proportion of World Series Winners to World Series Losers remains constant while sampling for our training and test datasets. In addition, the statistics for MLB basketball varies wildly over time, as you'll see soon in our analysis. Obscuring parts of the dataset or over-representing parts of the dataset also provide the potential for data-dredging and data mis-representation.

Variable Explanation

Next, I'll explain each of the variables in the dataset:

Variable Explanations:

Hits Scored **-** This statistic tells us how many hits were made by batters during one regular season by a team. I think the number of hits by batters will likely have a positive effect on both the number of home runs and the number of runs – more hits by batters means more possibilities of scoring either a home run or a run, since the batter both becomes a baserunner and gives other runners a chance to make it home. More hits by batters means more opportunities for the offense to move runners around the bases. Therefore, we would expect more hits by batters to be a positive indicator of a team's offensive skill. This is coded in the data as Hits.

Hits Scored **-** This statistic tells us how many hits were made by batters during one regular season by a team. I think the number of hits by batters will likely have a positive effect on both the number of home runs and the number of runs – more hits by batters means more possibilities of scoring either a home run or a run, since the batter both becomes a baserunner and gives other runners a chance to make it home. We would expect more hits by batters to be a positive indicator of a team's offensive skill. This is coded in the data as Hits.

Runs Scored - A run is when a player is able to advance around first, second and third base and returns safely to home plate, touching the bases in that order. This statistic tells us the number of runs that were scored by a team in a given competitive year. In the data this is represented with the variable r. We would expect a team able to score more runs to have a higher level of offensive skill. This is in our data as RunsScored.

Walk by Batters - This variable tells us how many walks were accomplished by batters on a team throughout the course of one competitive season. In baseball, a walk is when a pitcher throws four hits outside of the strike zone (meaning the pitch isn't considered legitimate) and the hitter swings at none of them, after which the batter is awarded first base. If the batter successfully recognizes and doesn't swing at four illegitimate pitches, it's better for the team currently in the offensive position, since they get to advance forward a base. In the data, this is represented by WalksBatted.

Saves - This variable tells us how many saves were awarded to pitchers on the team during the course of a given season. A save is when a pitcher single-handedly wins a game for their team under a number of circumstances (such as entering a game with a lead of less than three runs and pitching a perfect game). This is a metric of defensive performance and is coded in the data as Saves.

Fielding Percentage - The fielding percentage tells us the number of times that a defensive player either properly handled or fielded a ball. It's a measure of the defensive skill of a team, calculated taking the number of putouts and assists by defenders on a team and dividing them by the total number of chances to do so (putouts, assists, and errors). In our data, this variable is labelled FieldingPercentage. We would expect a higher fielding percentage to be indicative of a team having more defensive skill.

Runs Allowed – This variable tells us how many runs were allowed while this team was defending – it tells us how many times the offensive team was able to score on the defending team, and more runs allowed is indicative of poorer defensive performance. This is recorded in our data as RunsAllowed.

Hits Allowed – This variable tells us how many hits were allowed while this team was defending – it tells us how many times the offensive team was able to score hits on the defending team, and more hits allowed is indicative of poorer defensive performance. This is recorded in our data as HitsAllowed.

Home Runs - This variable simply tells us the number of home runs that a team is able to make during a given season. In baseball, a home run is when a batter successfully hits the ball and is then able to circle all the bases and reach home. We would expect more home runs to be an indicator of more offensive skill. In the data, this is represented by HomeRuns.

Strikeouts by Pitchers – In baseball, a strikeout occurs when a batter accumulates three strikes while they're at bat, which usually means the batter is out. It contributes to the performance of a team overall while they're in the defending position. We would expect more strikeouts by pitchers to indicate that a team is more skilled defensively. In the data, this is the variable Strikeouts

Number of Triples – This variable tells us the number of times batters on the team were able to safely reach third base after hitting the ball. It's a metric of offensive performance, and is recorded in the data as Triples.

Errors – An error occurs when a fielding player mishandles the bar, after which a batter or baserunner from the offensive team is able to advance one or more bases. More errors is a sign of poor defensive performance. In our data, this is recorded as Errors.

Earned Runs– This statistic tells tell us how many runs the pitcher allowed during their time at the pitchers mound (aka, how many runs were scored by the other team were the result of poor play from the defending team and not the result of an error or passed ball). This is a metric of defensive prowess, and lower numbers of earned runs indicate more defensive skills.

Average Earned Runs – This statistic divides Earned Runs by the number of games to give us the average number of earned runs per game.

Strikeouts – This statistic tells us the number of times that a batter struck out. Striking out is when a batter earns three strikes while at-bat, after which point they are out. More strikeouts mean less chances to score, so a high number of strikeouts indicates poor offensive performance.

At Bats – This statistic tells us the number of times that a team had a batter actively batting against a pitcher. At Bats are only recorded as long as the batter meets a number of conditions, such as not being hit by a pitch or being walked to first base.

OutsPitched – This statistic tells us the number of outs pitched by a team during a given season. For reference, an out is one-third of an inning. An out is when a hitter or baserunner is retired from the game, such as after they accumulate three strikes. More outs pitches is an indicator of more defensive skill.

Data Cleaning and Setup

```
Baseball <- read.csv("~/Downloads/team.csv")
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.
3.1 ──

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.6     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.4     ✓ stringr 1.4.0
## ✓ readr   2.0.2     ✓ forcats 0.5.1
```

```
## ── Conflicts ─────────────────────────────────────── tidyverse_conflict
s() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(FactoMineR)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa

library(psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

library(stargazer)

##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary St
atistics Tables.

##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

library(ggcorrplot)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(glmnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-2
```

```r
string_names <- c("h", "r", "bb", "sv", "fp", "soa", "ra", "er", "era", "ws_w
in", "triple" ,"ab", "hr", "ha", "ipouts", "e")



playball <- Baseball[, string_names]

#looking for missing values in our variables

for (i in seq_along(string_names)){

  print(sum(is.na(playball[,i])))
}
```

```
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
```

```r
table(playball$ws_win)
```

```
##
##         N    Y
##  357 2332  116
```

```r
#We want to clean the data here and turn this into a binary numerical respons
e variable. We want to replace all the Y's with a 1 (positive outcome represe
nting those who've won the world series) and the blanks and N's with zeroes,
indicating that team has not won the World Series.

playball <- playball %>%
    mutate(ws_win = case_when(
      ws_win == "Y" ~ 1,
      ws_win != "Y" ~ 0,
    ))
table(playball$ws_win)
```

```
##
##    0    1
## 2689  116
```

```
cat("The dataset has", paste(dim(playball)[1]), "individuals, and", paste(dim
(playball)[2]), "variables for each one of them", '\n')
```

```
## The dataset has 2805 individuals, and 16 variables for each one of them
```

```
#I would also like to engineer a new feature. The hits allowed (ha) variable
tells us how many hits a pitcher gives up (higher levels of hits allowed indi
cate worse defensive performance.) The hits variable (h) tells us how many ht
is were scored by batters throughout the course of a given season for a given
team, and is a metric of offensive performance. In order to create a variable
that combines elements of a team's performance during both the defensive and
offensive performance of a team, I will be creating a new variable, HitsRatio
, that divides the number of hits scored by batters by the number of hits all
owed.
```

```
playball$HitsRatio <- playball$h/playball$ha
```

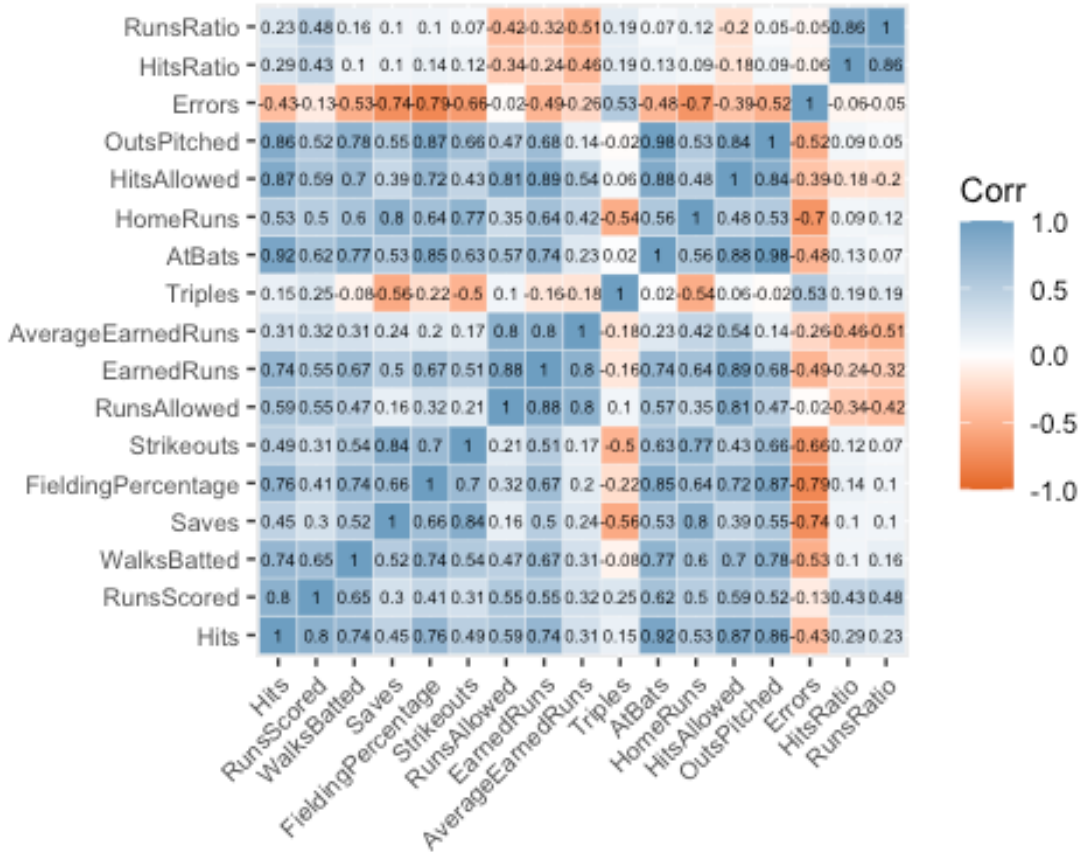
```
#I will also do the same with Runs Scored/Runs Allowed
```

```
playball$RunsRatio <- playball$r/playball$ra
#Now, I want to isolate just the variables I want to plan on using as depende
nt variables aqnd visualize the relationships between them.
```

```
#Renaming Variables so it's more obvious what they are
```

```
playball <- playball %>%
  rename(
    Hits = h,
    RunsScored = r,
    HomeRuns = hr,
    WalksBatted = bb,
    Saves = sv,
    FieldingPercentage = fp,
    RunsAllowed = ra,
    Errors = e,
    AverageEarnedRuns = era,
    Triples = triple,
    AtBats = ab,
    HitsAllowed = ha,
    OutsPitched = ipouts,
    Errors = e,
    EarnedRuns = er,
    Strikeouts = soa,
    )
```

```
#Normalizing the dataset

playball_with_response <- playball
playball <- select(playball, -c("ws_win"))

#nroamlizing the dataframe

mins <- apply(playball,2,min)
maxs <- apply(playball,2,max)
playball_scaled  <- data.frame(scale(playball,center=mins,scale=maxs-mins))

corr <- round(cor(playball_scaled, use = "complete.obs"), 2)

ggcorrplot(corr, type = "full", lab = TRUE, outline.col = "white", ggtheme =
ggplot2::theme_gray, colors = c("#E46726", "white", "#6D9EC1"), lab_col = "bl
ack", lab_size = 2,tl.cex = 8, tl.col = "black")
```



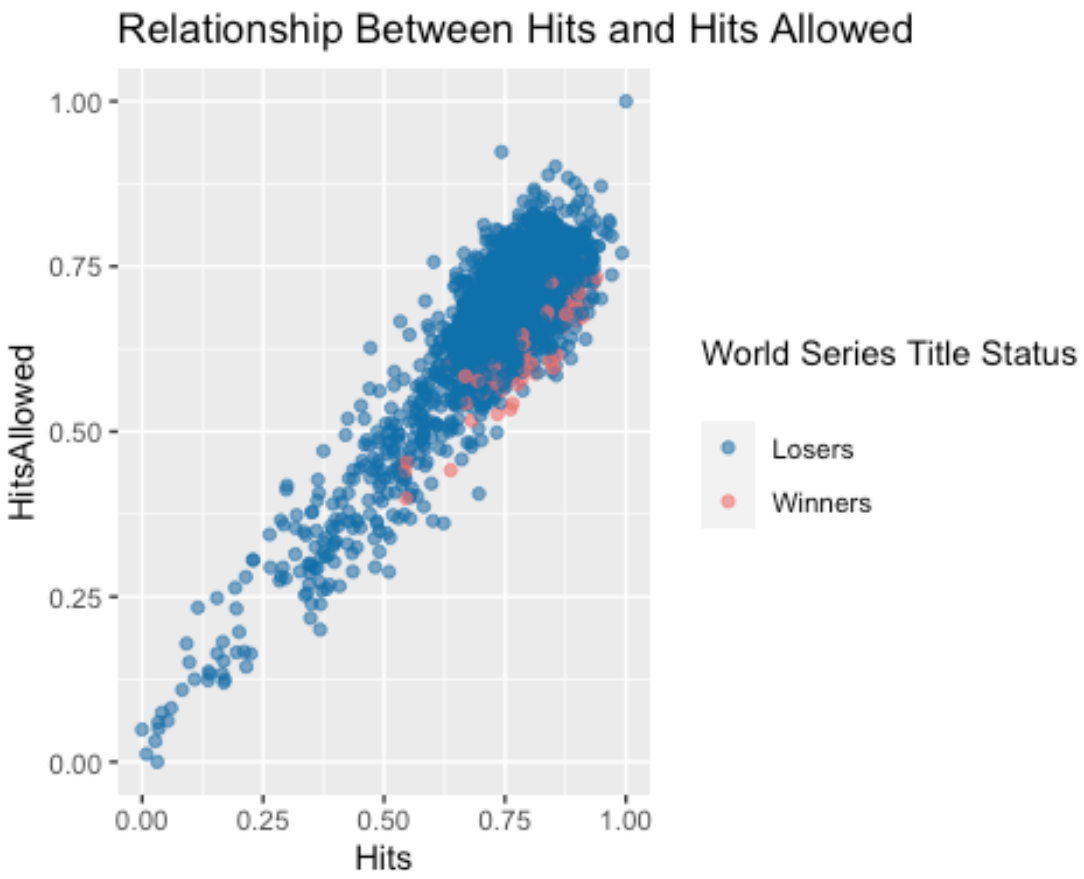

```
playball_scaled_response <- playball_scaled
playball_scaled_response$ws_win <- playball_with_response$ws_win

summary(playball)
```

```
##       Hits        RunsScored      WalksBatted        Saves
##  Min.   :  33   Min.   :  24.0   Min.   :  0.0   Min.   : 0.00
##  1st Qu.:1299   1st Qu.: 613.0   1st Qu.:425.0   1st Qu.: 9.00
##  Median :1393   Median : 690.0   Median :493.0   Median :24.00
##  Mean   :1346   Mean   : 681.9   Mean   :473.6   Mean   :23.67
##  3rd Qu.:1467   3rd Qu.: 763.0   3rd Qu.:554.0   3rd Qu.:38.00
##  Max.   :1783   Max.   :1220.0   Max.   :835.0   Max.   :68.00
##  FieldingPercentage   Strikeouts      RunsAllowed       EarnedRuns
##  Min.   :0.7600     Min.   :   0.0   Min.   :  34.0   Min.   :  25.0
##  1st Qu.:0.9600     1st Qu.: 501.0   1st Qu.: 609.0   1st Qu.: 500.0
##  Median :0.9700     Median : 735.0   Median : 688.0   Median : 590.0
##  Mean   :0.9615     Mean   : 731.2   Mean   : 681.9   Mean   : 570.9
##  3rd Qu.:0.9800     3rd Qu.: 965.0   3rd Qu.: 764.0   3rd Qu.: 666.0
##  Max.   :0.9910     Max.   :1450.0   Max.   :1252.0   Max.   :1023.0
##  AverageEarnedRuns    Triples          AtBats        HomeRuns
##  Min.   :1.220     Min.   :  0.0   Min.   : 211   Min.   :  0.0
##  1st Qu.:3.340     1st Qu.: 31.0   1st Qu.:5127   1st Qu.: 42.0
##  Median :3.820     Median : 41.0   Median :5389   Median :107.0
##  Mean   :3.815     Mean   : 47.1   Mean   :5142   Mean   :101.1
##  3rd Qu.:4.300     3rd Qu.: 60.0   3rd Qu.:5517   3rd Qu.:149.0
##  Max.   :8.000     Max.   :150.0   Max.   :5781   Max.   :264.0
##   HitsAllowed    OutsPitched       Errors        HitsRatio
##  Min.   :  49   Min.   : 162   Min.   : 47.0   Min.   :0.2292
##  1st Qu.:1288   1st Qu.:4077   1st Qu.:116.0   1st Qu.:0.9455
##  Median :1392   Median :4236   Median :145.0   Median :1.0025
##  Mean   :1346   Mean   :4022   Mean   :186.3   Mean   :1.0030
##  3rd Qu.:1470   3rd Qu.:4341   3rd Qu.:217.0   3rd Qu.:1.0594
##  Max.   :1993   Max.   :4518   Max.   :639.0   Max.   :1.7959
##    RunsRatio
##  Min.   :0.1711
##  1st Qu.:0.8813
##  Median :1.0086
##  Mean   :1.0171
##  3rd Qu.:1.1329
##  Max.   :2.4280
```

Now, I want to conduct some data visualizations - specifically, I want to graph scatter plots for different variables in our set nd differentiate between our two groups of interest - World Series Winners and World Series Losers. Specifically, I want to see whether or not the relationships between these variables, the spread, or the center differentiates in any way for the two groups that's immediately obvious.

```
colors <- c("#1170AA", "#", "#55AD89")

ggplot(playball_scaled_response, aes(x = Hits, y = HitsAllowed, color = as.fa
ctor(ws_win))) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = colors) +
  labs(color = "World Series Title Status\n", title = "Relationship Between H
its and Hits Allowed") +
```

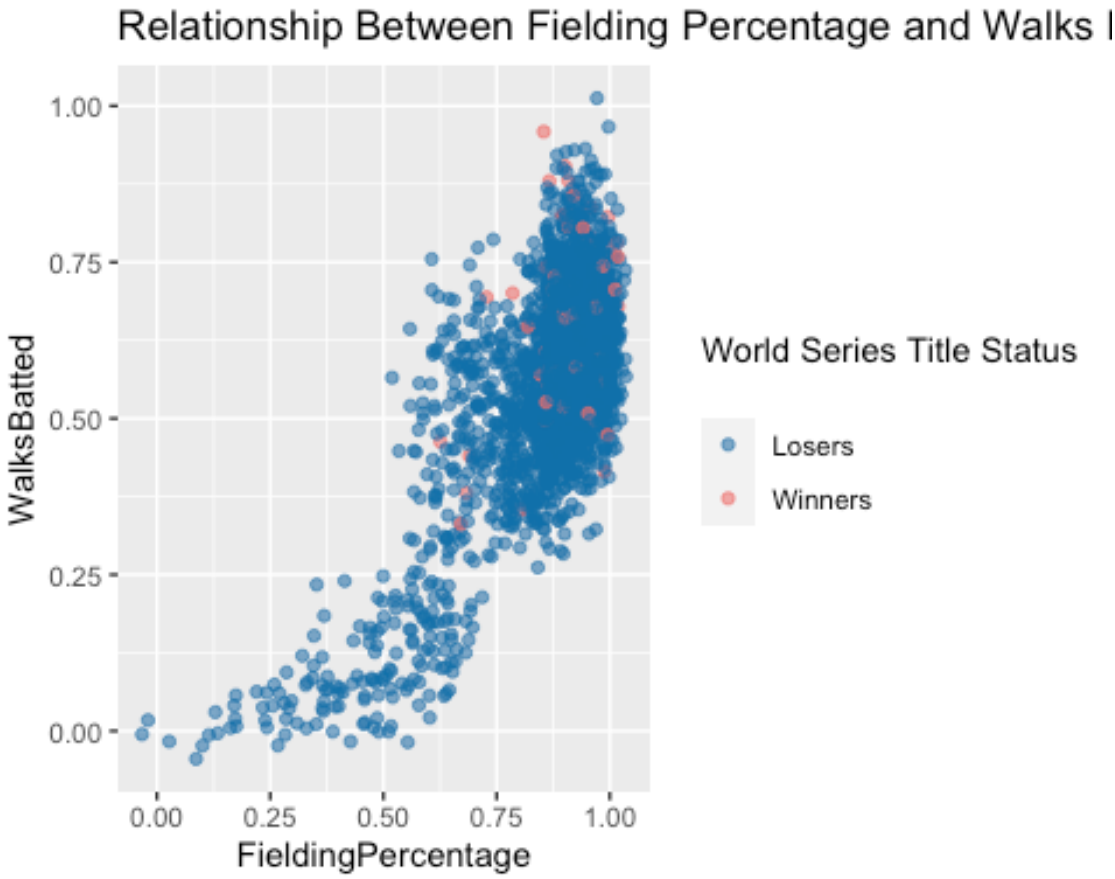
```
  scale_color_manual(labels = c("Losers", "Winners"), values = c("#1170AA", "
#EF6F6A"))

## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



As we can see, World Series Winners appear to have both higher numbers of hits and higher numbers of hits allowed - this is surprising, given that we would expect high-performing teams to have a lower number of hits allowed. However, high-performing teams also tend to perform in more games, since they're more likely to make it to playoffs - this would explain some, but not all of the higher number of hits allowed. However, World Series Winner also appearar to be more closely clustered in the top right, and also have lower number of hits allowed comparative to the number of hits when visualized next to teams that did not win the World Series.

```
colors <- c("#1170AA", "#EF6F6A", "#55AD89")

jitter <- position_jitter(width = 0.05, height = 0.05)

ggplot(playball_scaled_response, aes(x = FieldingPercentage, y = WalksBatted,
color = as.factor(ws_win))) +
  geom_point(position =jitter,alpha = 0.6) +
  scale_color_manual(values = colors) +
```

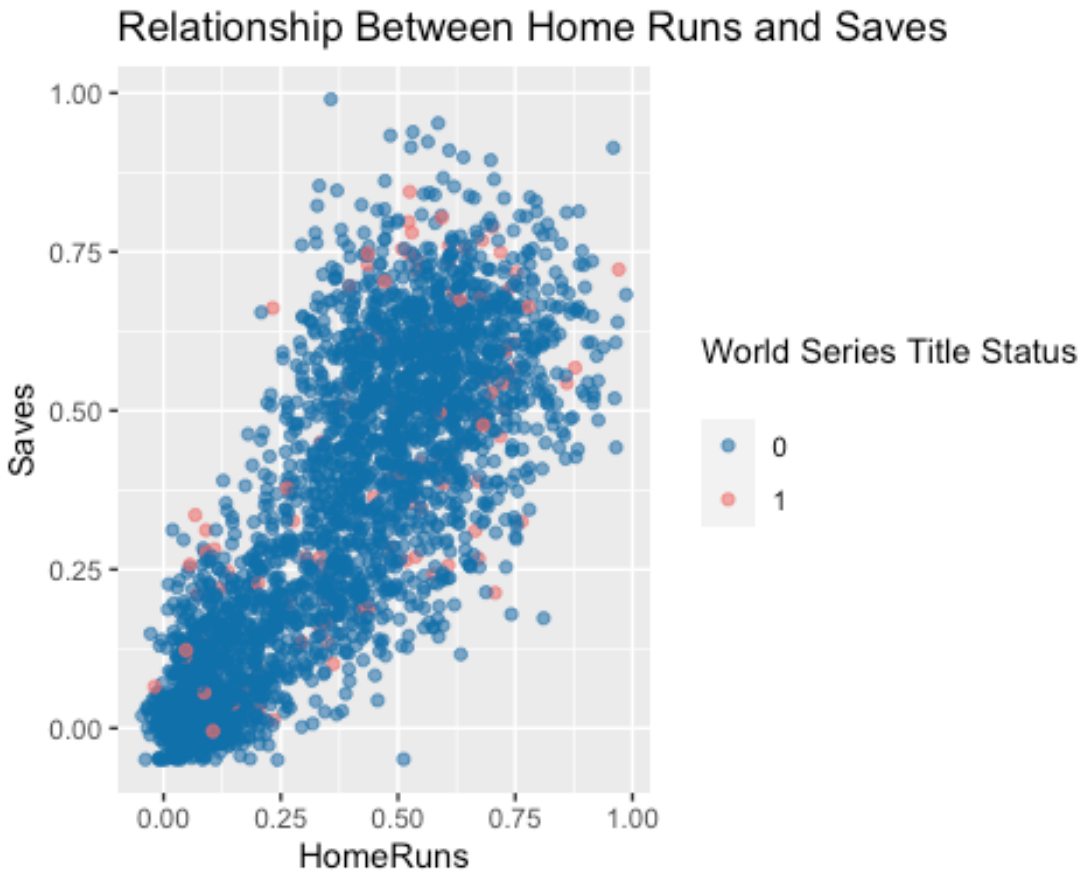
```
  labs(color = "World Series Title Status\n", title = "Relationship Between F
ielding Percentage and Walks Batted") +
  scale_color_manual(labels = c("Losers", "Winners"), values = c("#1170AA", "
#EF6F6A"))

## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



World Series Winners tend to have both higher fielding percentages and more walks batted. Once again, the winners tend to be more closely clustered in the top right, but both of these metrics are considered "positive" metrics, so it makes more sense that World Series winners would be more featured in the upper ranges of both statistics.

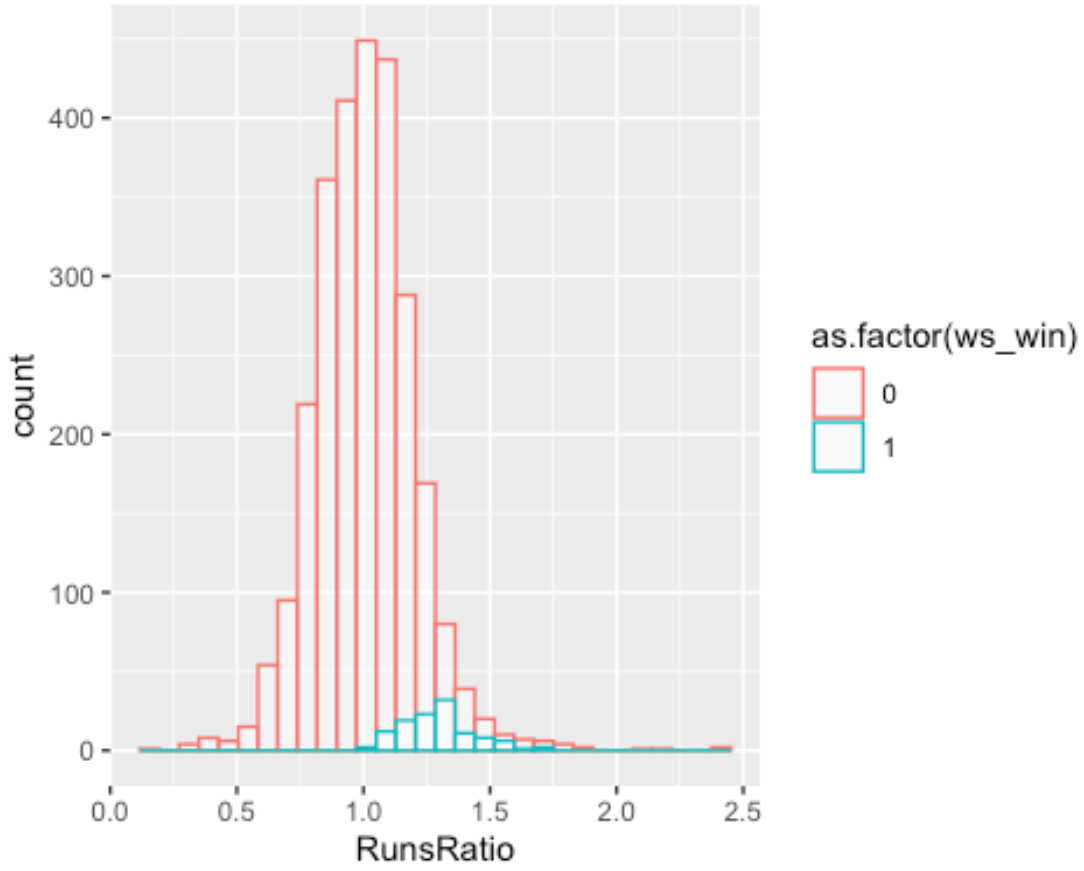
```
colors <- c("#1170AA", "#EF6F6A", "#55AD89")


ggplot(playball_scaled_response, aes(x = HomeRuns, y = Saves, color = as.fact
or(ws_win))) +
  geom_point(position = jitter,alpha=0.6) +
  scale_color_manual(values = colors) +
```

```
  labs(color = "World Series Title Status\n", title = "Relationship Between H
ome Runs and Saves")
```



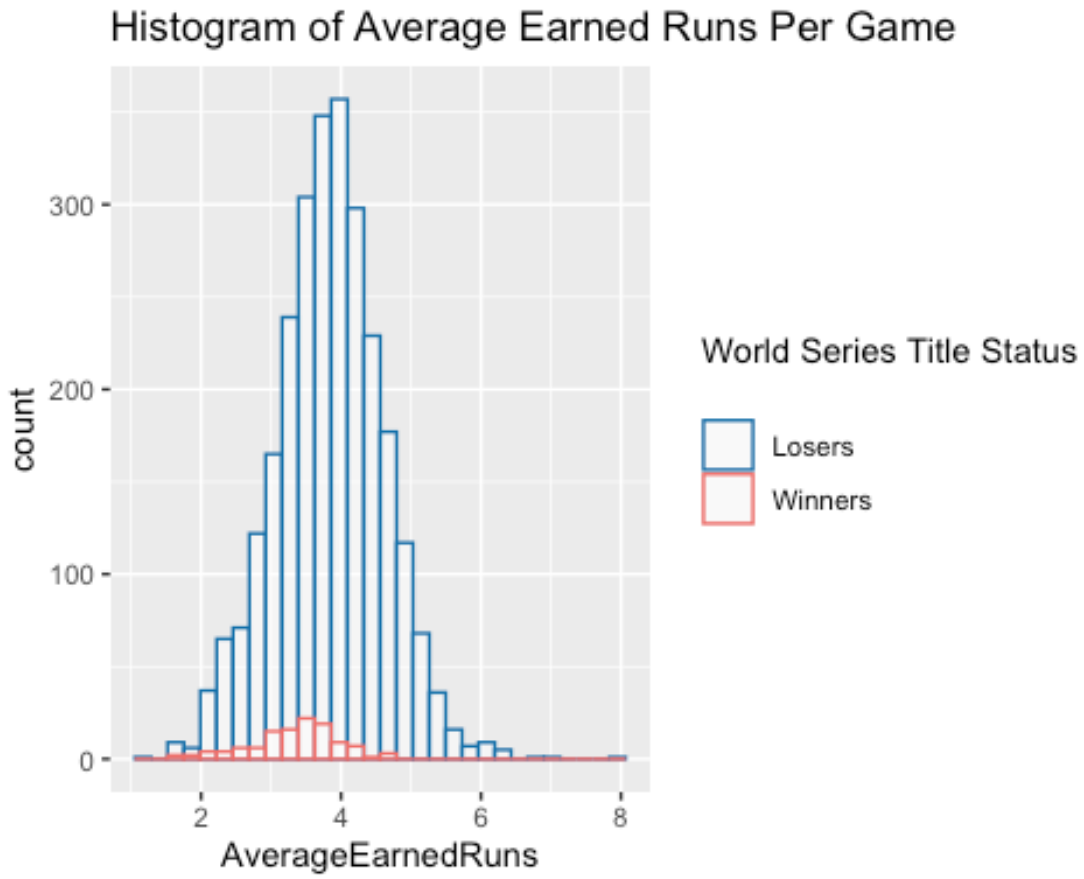
There is no immediately obvious pattern visible here – the distribution of World Series Winners seems to mimic the spread and center of the non-winners.

```
ggplot(playball_with_response, aes(x=RunsRatio, color=as.factor(ws_win))) +
  geom_histogram(fill="white", alpha=0.5, position="identity")

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



Next, I created a grouped histogram of the ratio of Runs over Runs Allowed. A team with more runs is considered to have performed better offensively during a season, and a team with a high number of runs allowed is considered to have performed poorly defensively during a season. As we can see, the center of the distribution of the runs ratio is much higher for World Series Winners, and also appears to be more closely clustered around a value of about 1.25 as a ratio, while the most teams that are not World Series Winners tend to have a runs ratio closer to 1.

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

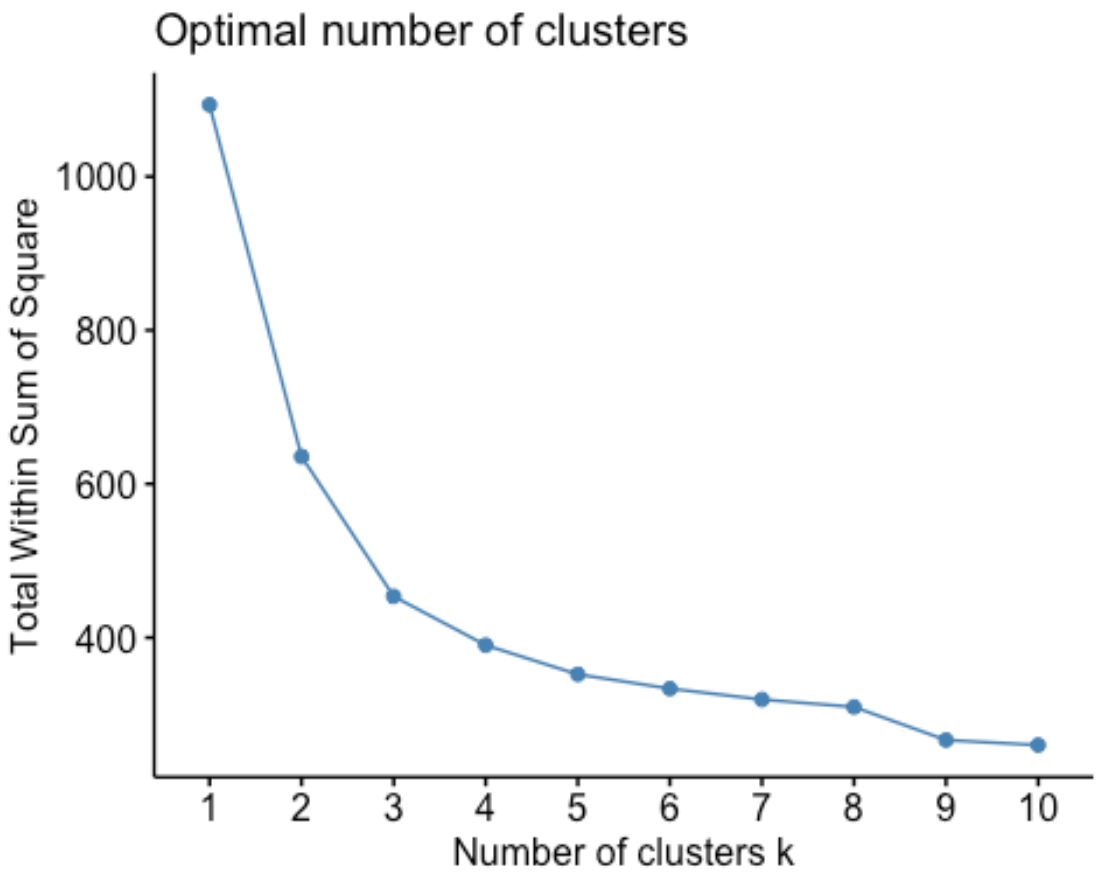


In the grouped histograms of Average Earned Runs, we can observe that World Series Winners tend to have a lower amount of Average Earned Runs. Average Earned Runs are measure of defensive prowess - they tell us how many runs the pitcher allowed during their time at the pitchers mound (aka, how many runs were scored by the other team that were the result of poor play from the defending team and not the result of an error or passed ball). We would expect a "better" performing team to have less earned runs while pitching, so it makes sense that the center is slightly lower for World Series Winners.

Clustering

Next, as a means of finding patterns within our dataset, we'll be using k-means clustering. Clustering is a method that groups observations (data points) together into a different clusters based on similarities they might share. K-means clustering, which we'll be performing, randomly creates a number of centers, and then iterates through the data until it finds the optimal location of those centers within the data, in order to form different clusters. We'll be using clustering in order to explore the data and see if there are any patterns that aren't immediately obvious.

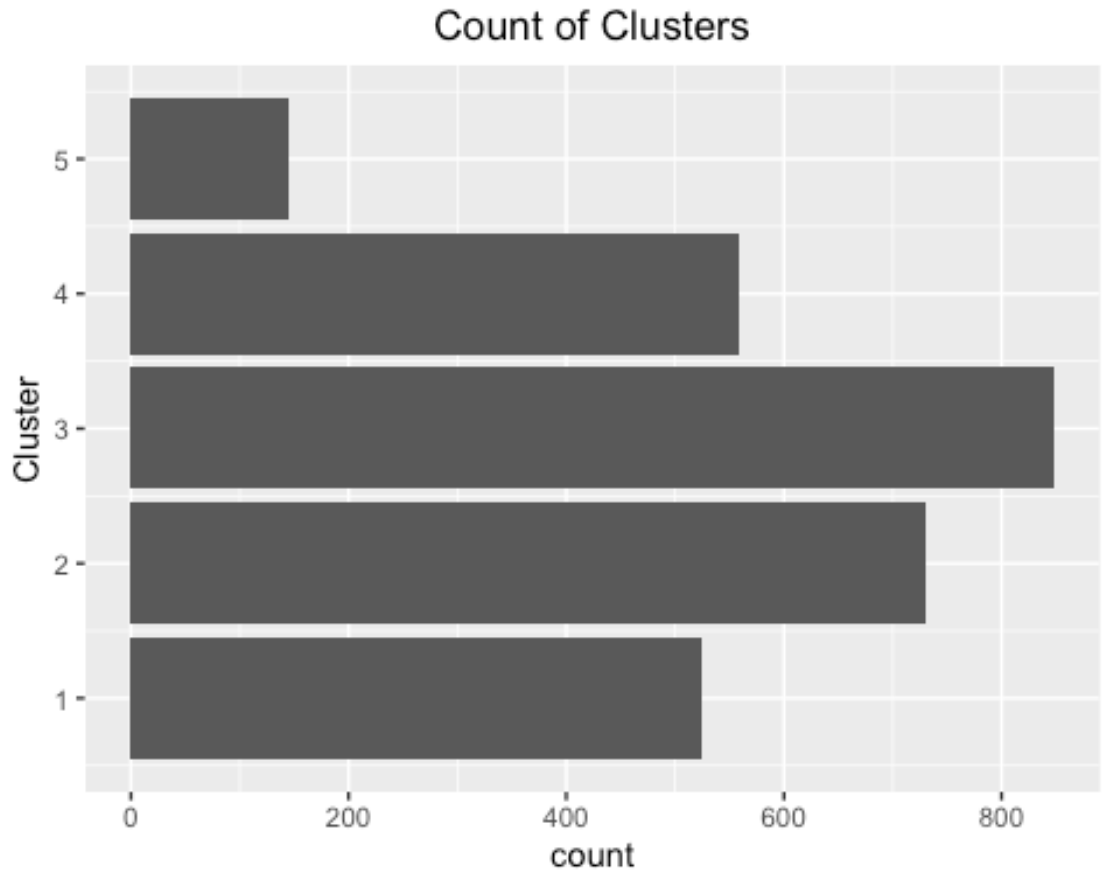
```
fviz_nbclust(as.matrix(playball_scaled), kmeans, method="wss")
```




```
# Based on this graph, it looks like 5 is the optimal number of clusters - we
can see that the kink/change in the slope occurs at 6 clusters, so we will ma
rk that as the point of diminishing returns.

k = 5
cluster_k <- kmeans(playball_scaled, k, nstart = 20 )

kmeans_basic_table <- data.frame(cluster_k$size, cluster_k$centers)
kmeans_df <- data.frame(Cluster = cluster_k$cluster, playball_scaled)
head(kmeans_df)

##   Cluster       Hits RunsScored WalksBatted      Saves FieldingPercentage
## 1       5 0.22457143 0.31521739  0.07185629 0.04411765          0.3030303
## 2       5 0.16571429 0.23244147  0.07185629 0.01470588          0.2597403
## 3       5 0.16857143 0.18812709  0.03113772 0.00000000          0.2164502
## 4       5 0.08285714 0.09448161  0.03952096 0.00000000          0.1731602
## 5       5 0.21142857 0.23244147  0.03952096 0.00000000          0.3030303
## 6       5 0.21542857 0.29431438  0.05508982 0.00000000          0.3463203
##   Strikeouts RunsAllowed EarnedRuns AverageEarnedRuns    Triples     AtBat
s
## 1 0.01586207   0.2208539 0.08416834         0.3436578 0.24666667 0.2084380
```

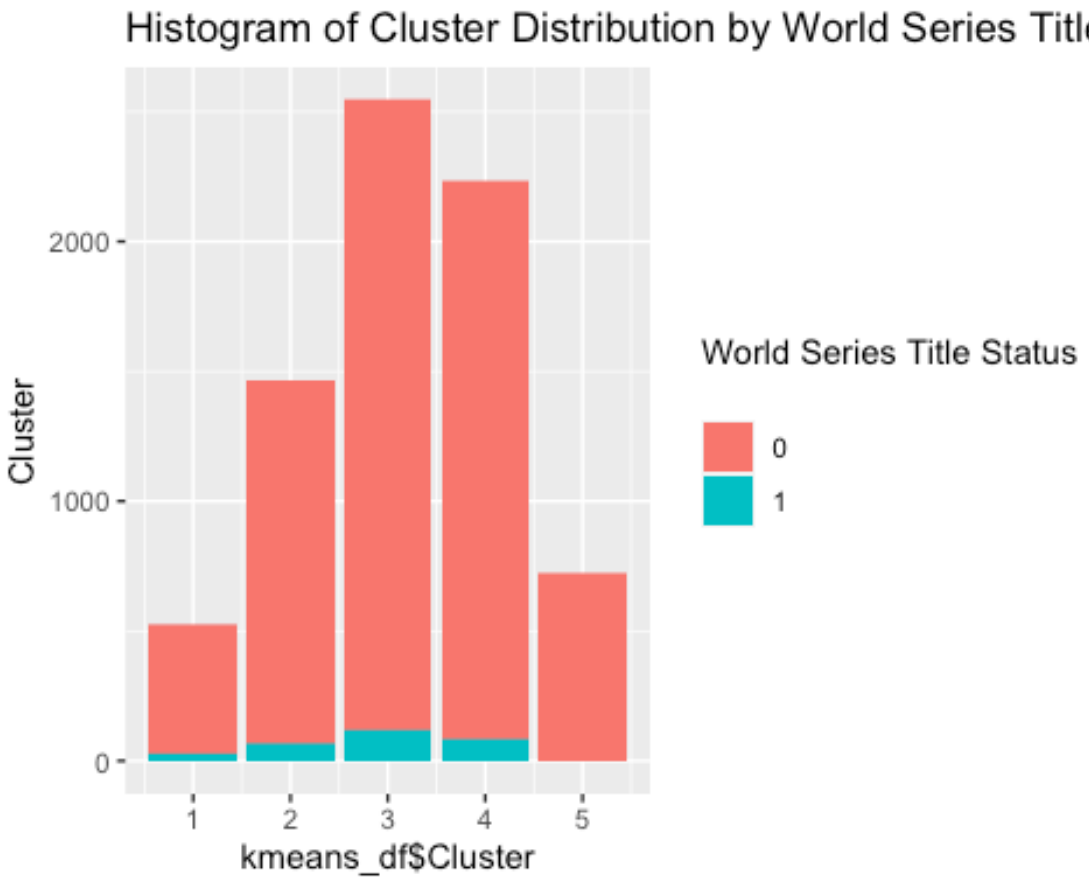
```
6
## 2 0.01517241   0.1699507 0.05210421         0.2271386 0.14000000 0.1768402
2
## 3 0.02344828   0.2520525 0.09118236         0.4262537 0.26666667 0.1750448
8
## 4 0.01172414   0.1715928 0.07214429         0.5825959 0.05333333 0.0960502
7
## 5 0.01517241   0.2290640 0.09619238         0.3687316 0.14000000 0.2141831
2
## 6 0.01103448   0.1904762 0.11222445         0.5501475 0.18000000 0.1921005
4
##      HomeRuns HitsAllowed OutsPitched    Errors HitsRatio RunsRatio
## 1 0.011363636   0.1635802   0.1528926 0.3006757 0.5946036 0.5105887
## 2 0.037878788   0.1332305   0.1356749 0.2888514 0.5230788 0.4794316
## 3 0.026515152   0.1527778   0.1377410 0.2972973 0.4587901 0.2477456
## 4 0.007575758   0.1090535   0.0792011 0.1959459 0.2890220 0.1740097
## 5 0.003787879   0.1666667   0.1646006 0.3040541 0.5433294 0.3517132
## 6 0.034090909   0.1440329   0.1342975 0.2483108 0.6491354 0.5505102
```

```
ggplot(data = kmeans_df, aes(y = Cluster)) + geom_bar(aes(fill = Cluster)) +
  ggtitle("Count of Clusters") +
  theme(plot.title = element_text(hjust =0.5))
```



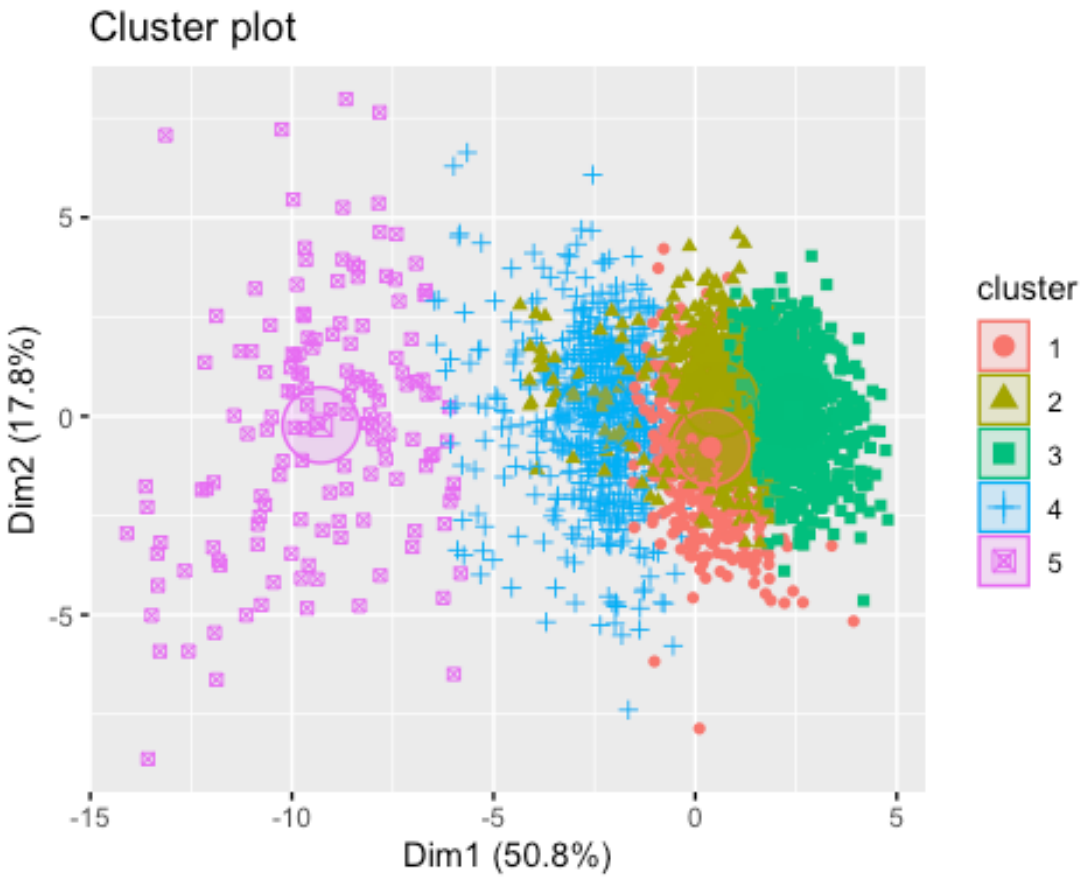
The count of observations within the clusters isn't even – cluster 5 is drastically smaller than the other clusters, and cluster 3 has the most observations.

```
ggplot(kmeans_df, aes(fill=as.factor(playball_scaled_response$ws_win), y=Clus
ter, x=kmeans_df$Cluster)) +
  scale_color_manual(labels = c("Losers", "Winners"), values = c("#1170AA", "
#EF6F6A"))+
  labs(fill = "World Series Title Status\n", title = "Histogram of Cluster Di
stribution by World Series Title Status") +
    geom_bar(position="stack", stat="identity")
```



As we can see, cluster 5 also has a much smaller number of World Series winners! Although the number isn't zero, it's low enough that it becomes extremely difficult to visualize in the bar plot. Let's try to visualize our clusters and figure out if they have any distinctive characteristics that could explain this.

```
fviz_cluster(cluster_k, data = playball_scaled, geom = c("point"), ellipse.ty
pe = "euclid")
```




```
Baseball$Cluster <- as.factor(kmeans_df$Cluster)
playball_scaled_response$Cluster <- as.factor(kmeans_df$Cluster)
```

As we can see, we have five clearly visually distinct clusters, each of which occupies a different amount of Dim1 and Dim2 (the two principal components used to map the clusters).

Now we'll compare the summary stats of each cluster!

```
library("viridis")

## Loading required package: viridisLite

library("hrbrthemes")

## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use th
ese themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto C
ondensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/ari
alnarrow
```

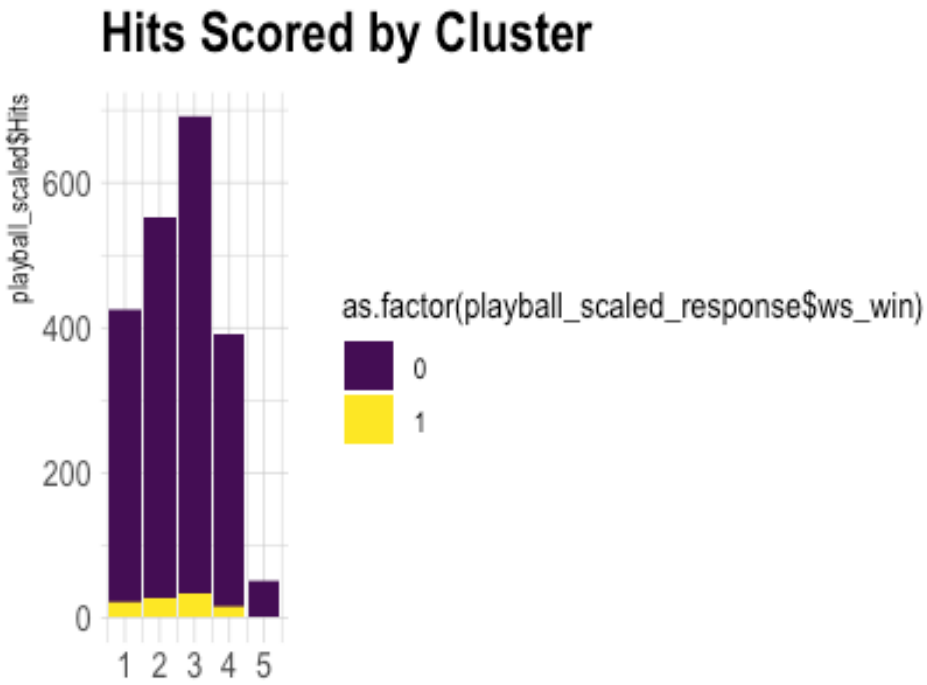
```
cluster_summary <- c()
for (i in 1:5) {

tempo <- c(apply(playball_scaled[cluster_k$cluster==i,],2,mean),table(playbal
l_scaled[cluster_k$cluster==i,]$type))
cluster_summary <- rbind(cluster_summary,tempo)
}
rownames(cluster_summary) <- c("cluster_1","cluster_2","cluster_3","cluster_4
", "cluster_5")
print(cluster_summary)
```

```
##                Hits RunsScored WalksBatted       Saves FieldingPercentage
## cluster_1 0.8095834  0.5768241  0.60601774 0.191569376          0.8906679
## cluster_2 0.7532312  0.5221864  0.60870107 0.444515973          0.9242395
## cluster_3 0.8141078  0.6097871  0.64547085 0.600773446          0.9512681
## cluster_4 0.6984803  0.5337490  0.48022879 0.072238035          0.7765830
## cluster_5 0.3475040  0.3068504  0.09222389 0.008374183          0.4491342
##           Strikeouts RunsAllowed EarnedRuns AverageEarnedRuns   Triples
## cluster_1  0.3468386   0.5815832  0.6067306         0.4315088 0.4058524
## cluster_2  0.5709732   0.5011411  0.5471243         0.3627189 0.2504788
## cluster_3  0.7187093   0.5805953  0.6525610         0.4420361 0.2132075
## cluster_4  0.3396997   0.5108513  0.4279545         0.2988549 0.4903226
## cluster_5  0.1139416   0.3035201  0.1684689         0.2827557 0.2130556
##              AtBats   HomeRuns HitsAllowed OutsPitched    Errors HitsRatio
## cluster_1 0.9166438 0.28983056   0.7316317   0.9073889 0.2285950 0.4845211
## cluster_2 0.9144381 0.44582452   0.6669059   0.9280581 0.1449537 0.4965852
## cluster_3 0.9563635 0.63095430   0.7207937   0.9575474 0.1097853 0.4970020
## cluster_4 0.8324196 0.10346068   0.6179763   0.8360092 0.4755098 0.4984410
## cluster_5 0.4110525 0.02796191   0.3098387   0.3713699 0.5279655 0.4788870
##           RunsRatio
## cluster_1 0.3576909
## cluster_2 0.3757429
## cluster_3 0.3796593
## cluster_4 0.3845790
## cluster_5 0.3670561
```

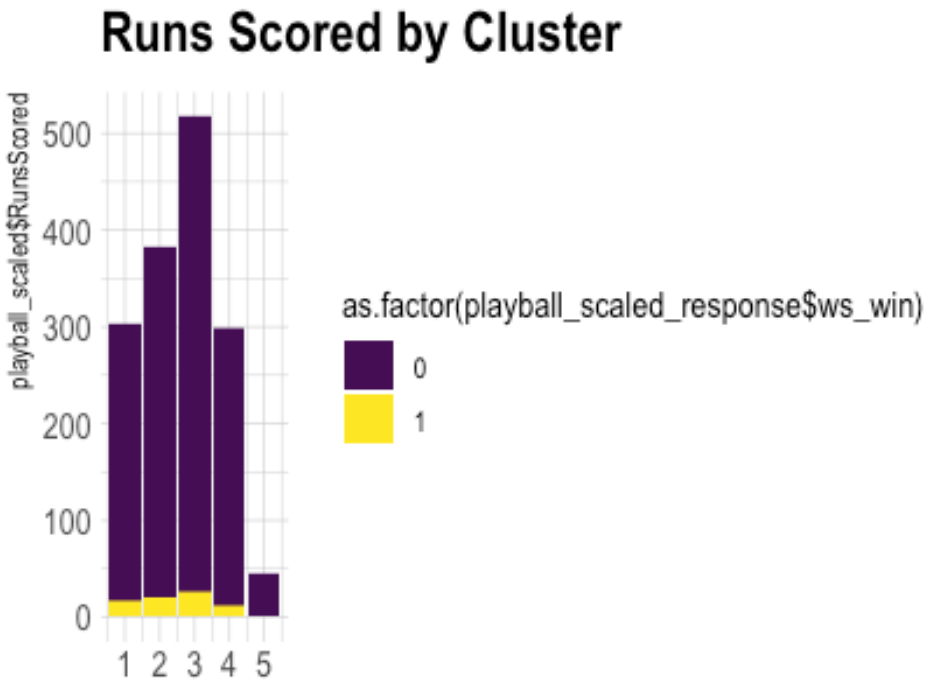
Some notable trends here include the fact that cluster 5 has notably lower scores for most statistics. This could indicate that teams grouped in cluster 5 tend to be generally worse performers. In general, for most of these variables Cluster 1 has the highest mean parameter value, cluster 2 has the second-highest, and so on, with cluster 5 having the lower mean value of the rest of the clusters. This is especially interesting because it suggests there is a potential ordering to our clusters – cluster 2 could all be the second-best teams, cluster 1 could be top-performing teams, and cluster 5 could be bottom performing teams. Let's look at some visualizations of statistics by cluster in order to help us figure out what this potential ordering could be.

```
ggplot(data= kmeans_df,aes(fill=as.factor(playball_scaled_response$ws_win), y
=playball_scaled$Hits, x=Cluster)) +
    geom_bar(position="stack", stat="identity") +
    scale_fill_viridis(discrete = T) +
    theme_ipsum() +
```

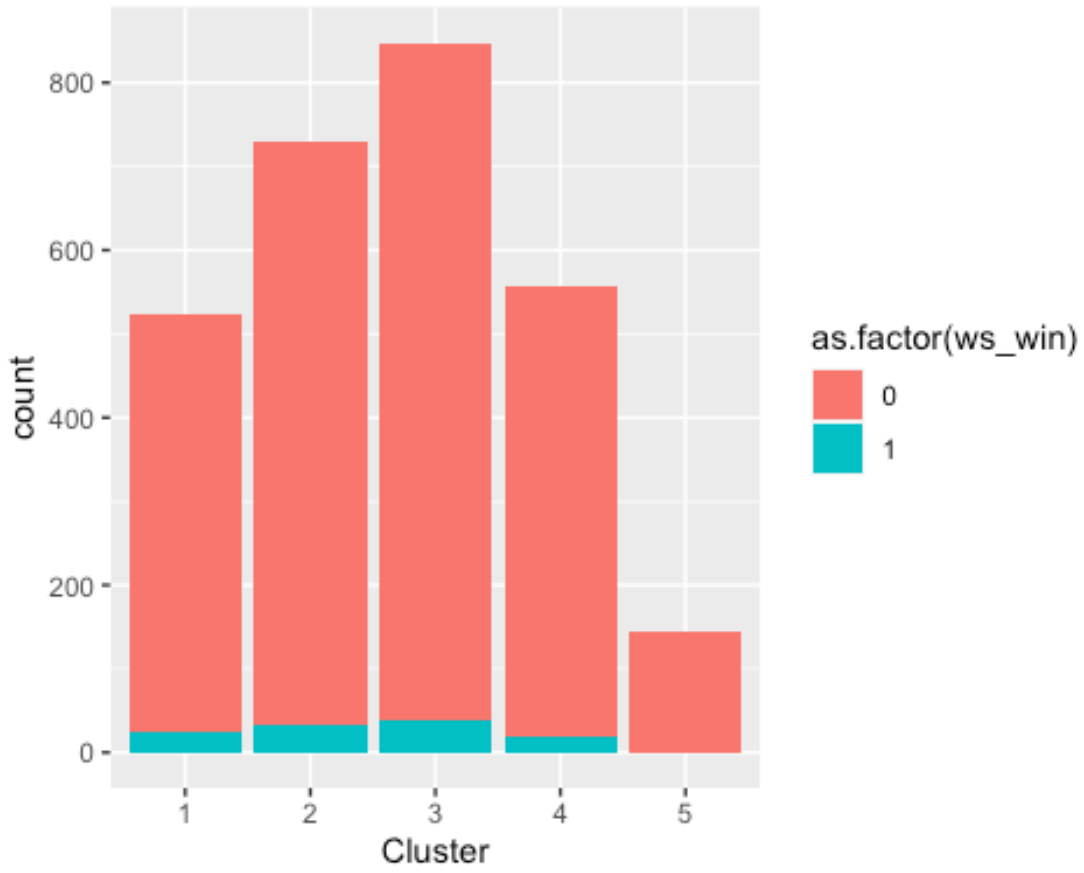
```
    xlab("") + labs(color = "World Series Title Status\n", title = "Hits Scor
ed by Cluster")     +
    scale_color_manual(labels = c("Losers", "Winners"))
```




```
ggplot(data= kmeans_df,aes(fill=as.factor(playball_scaled_response$ws_win), y
=playball_scaled$RunsScored, x=Cluster)) +
    geom_bar(position="stack", stat="identity") +
    scale_fill_viridis(discrete = T) +
    theme_ipsum() +
    xlab("") + labs(color = "World Series Title Status\n", title = "Runs Scor
ed by Cluster")     +
    scale_color_manual(labels = c("Losers", "Winners"))
```

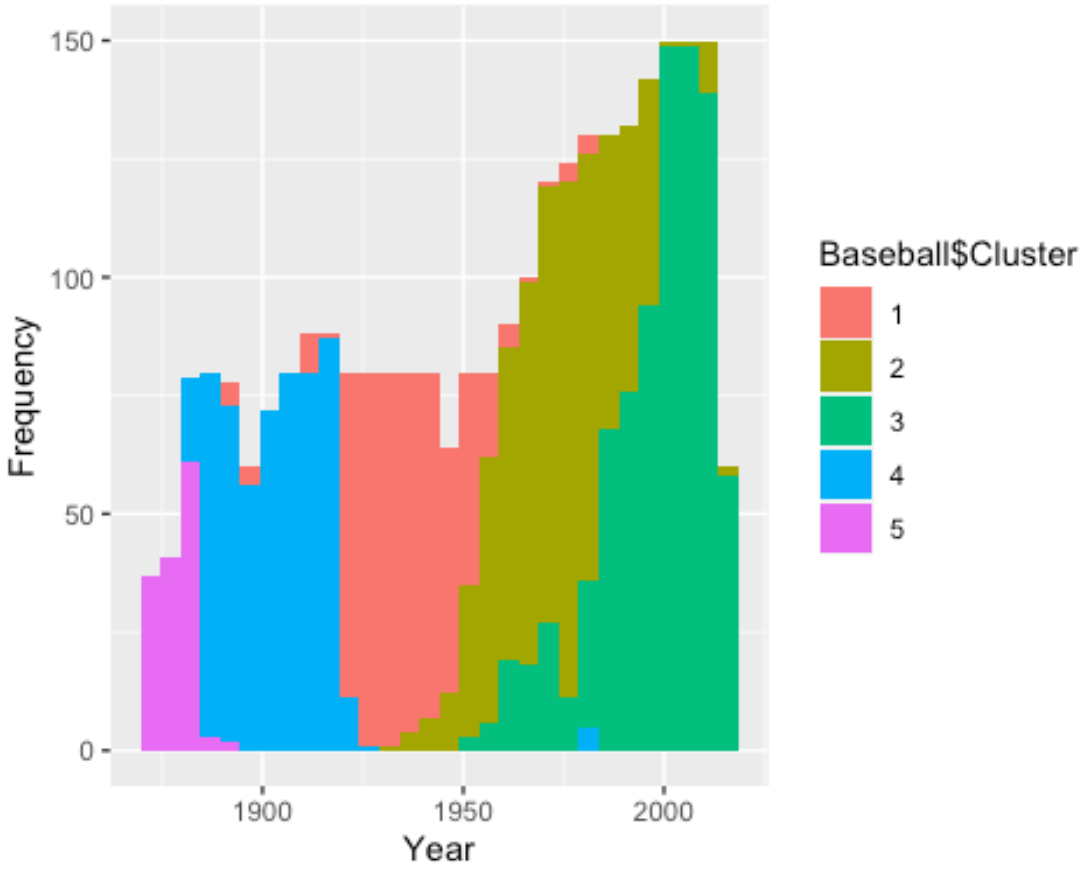



```
ggplot(playball_scaled_response, aes(fill = as.factor(ws_win), x = Cluster))
+
  geom_bar()
```



Generally, these visualizations don't show us any immediately obvious visual trends – the distributions of World Series winners among the bars appears to be roughly the same, and there are no obvious trends in the bar plots of runs scored and hits scored. As we can see from our stacked barplot, cluster 5 contains extremely few World Series winners - the distribution of World Series Winners in the other clusters roughly mirror the distribution of World Series Winners in the overall dataset. This could indicate that all the observations in Cluster 5 share a common factor that make them significantly less likely to win the World Series. There doesn't seem to be a significantly higher portion of World Series Winners in cluster 1, which could indicate that although the teams appear to be higher performing, they still win the World Series at the same rate.

```
ggplot(data=Baseball, aes(x=Baseball$year, fill = Baseball$Cluster)) +
  xlab("Year") + ylab("Frequency") +
  geom_histogram()
```



From our last histogram, which shows the distribution of clusters over time, we can see that that time appears to be the most immediately visible relationship between the different clusters - each cluster seems to be centered around a series of 3-4 decades. This would explain the extremely low number of World Series Winners for cluster 5 - as the earliest cluster, there were simply less teams in the MLB at this time, and so both the number of teams in this cluster and the proportion of World Series Winners appears to be noticeably smaller for this group. In addition, as Major League Baseball has continued to exist, teams continue to be higher performing and would therefore have better scores in nearly every statistic – this both explains the increase in summary statistics we pointed out upon initial observation of the clusters and why the distribution of World Series winners amongst all clusters was roughly the same despite clear differences in performance.

The results of this cluster analysis seem particularly useful to group our observations into different “eras” of Major League Baseball, and as such, they’ve been added to our dataset as a potential variable.

PCA

Next, we’ll be engaging in principal components analysis. Before we attempt any type of prediction, we’ll split our model into a testing and training datasets. This is ultimately

helpful because it helps us objectively evaluate the performance of models and classifiers when they are fit to data that's not used to train the model.

When forming our testing and training datasets, we want to make sure that the distribution of World Series Winners and Losers in each dataset mirrors the distribution of Winners and Losers in the total dataset.

```
playball_scaled_to_split <- playball_scaled_response
playball_scaled_to_split$row_num <- seq.int(nrow(playball_scaled_to_split))
only_winners <- subset(playball_scaled_to_split, ws_win ==1)
table(only_winners$ws_win)

##
##   1
## 116

only_losers <- subset(playball_scaled_to_split, ws_win ==0)
table(only_losers$ws_win)

##
##    0
## 2689

proportion = nrow(only_winners)/nrow(only_losers)
proportion

## [1] 0.04313871

#We'll section off 20% of our data for use in test. This would be 561 rows, s
o we'll make sure that 24 (4% of our observations in the testing dataset) are
composed of World Series Winners.

winners_row_nums <- sample(only_winners$row_num, 24)
losers_row_nums <- sample(only_losers$row_num, 561-24)
total_row_nums <- c(winners_row_nums, losers_row_nums)

winners_sample <- playball_scaled[winners_row_nums, ]
losers_sample <- playball_scaled[losers_row_nums, ]

dim(winners_sample)

## [1] 24 17

dim(losers_sample)

## [1] 537  17

test_scaled <- rbind(winners_sample, losers_sample)
train_scaled <- playball_scaled_to_split[!playball_scaled_to_split$row_num %i
n% total_row_nums, ]
dim(test_scaled)
```

```
## [1] 561  17

dim(train_scaled)

## [1] 2244   20

winners_sample_response <- playball_scaled_response[winners_row_nums, ]
losers_sample_response <- playball_scaled_response[losers_row_nums, ]

#Response Variable for Train
response_train <- playball_scaled_to_split[!playball_scaled_to_split$row_num
%in% total_row_nums, ]$ws_win


#Response Variable for Test
response_test <-  playball_scaled_to_split[playball_scaled_to_split$row_num %
in% total_row_nums, ]$ws_win
```

First, let's calculate the classification error, precision, and recall if we predicted randomly (but proportional to the frequency of World Series Winners in our dataset). This will help us evaluate whether or not our models perform better or worse than simply choosing randomly based on the proportion of World Series Winners. These numbers will become relevant again towards the end of our modelling, when we need a set of baseline metrics to compare our models to.

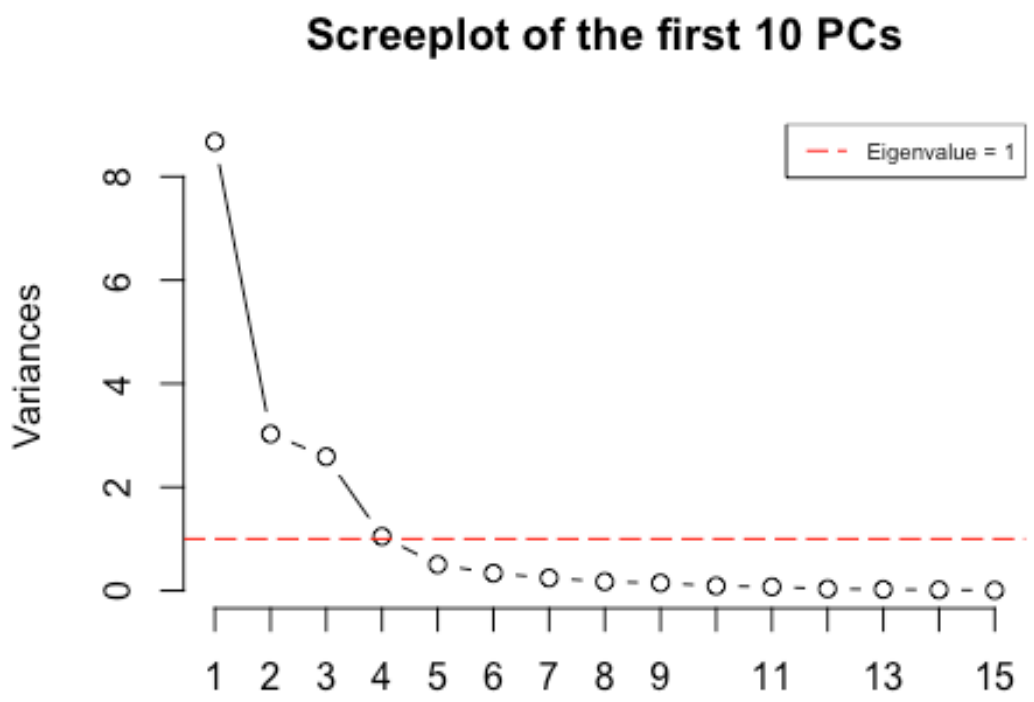
```
random <- as.integer(rbernoulli((561), p = proportion))
error <- mean(response_test != round(random))
cm <- confusionMatrix(as.factor(round(random)), reference = as.factor(respons
e_test), positive = "1")

cm$byClass[c(5,6)]

##  Precision     Recall
## 0.09523810 0.08333333
```

Principal Component Analysis

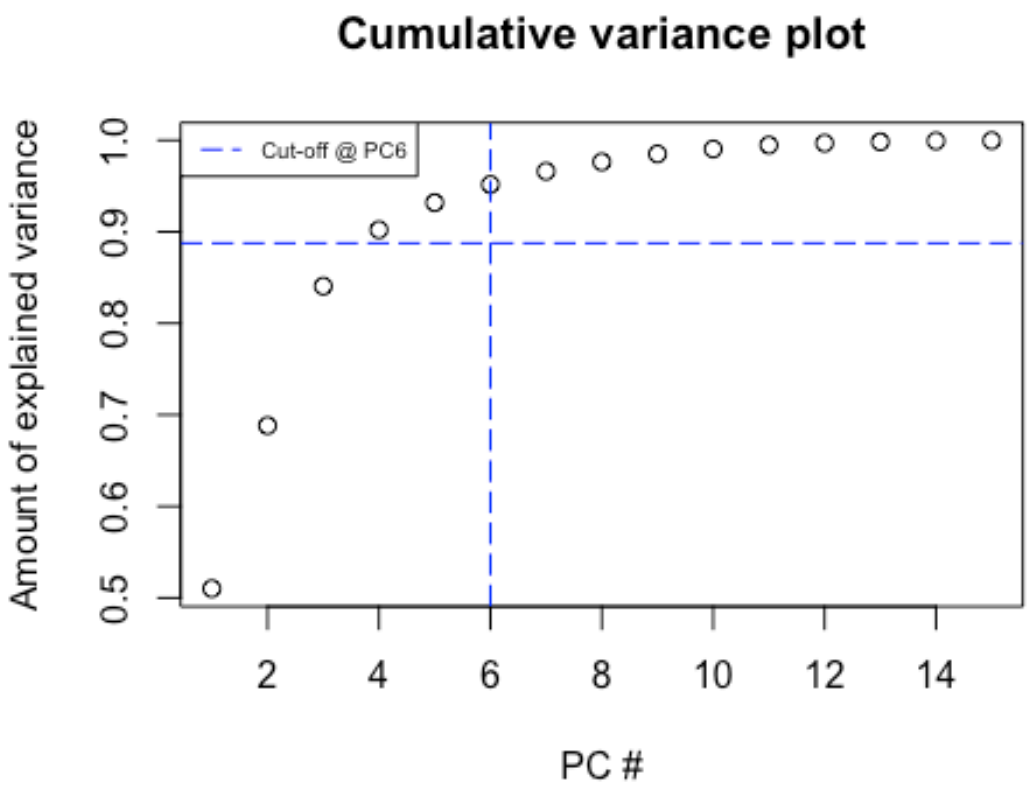
In order to explore patterns within the data that might not be immediately obvious, we'll be conducting principal component analysis, which attempts to reduce the dimensions of the dataset by combining several variables into a groups of latent "factors." The goal of latent variables is to capture the variance of multiple measured variables into a common factor, an unmeasured and unobserved variable. The maximum common variance will be taken from all variables and put into a common score, which can then be used for common analysis.

```
train_scaled_variables <- train_scaled
drop <- c("Cluster","row_num", "ws_win")
train_scaled_variables = train_scaled_variables[,!(names(train_scaled_variabl
es) %in% drop)]
```

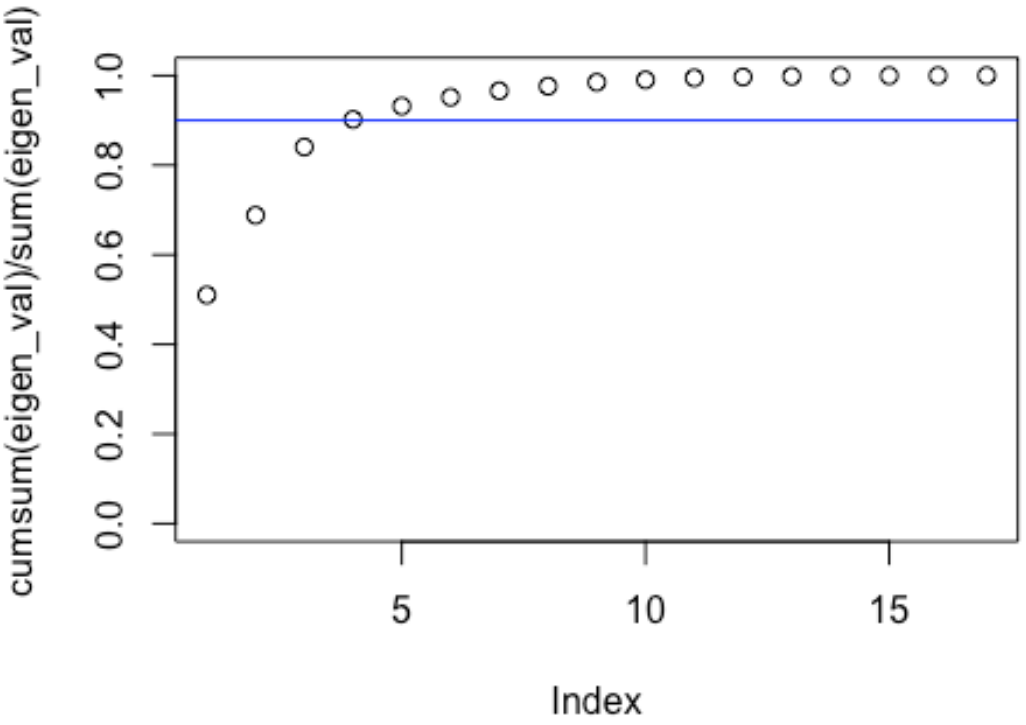
```
pca_out <- prcomp(train_scaled_variables, scale=TRUE)


screeplot(pca_out, type = "l", npcs = 15, main = "Screeplot of the first 10 P
Cs")
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),
       col=c("red"), lty=5, cex=0.6)
```




```
cumpro <- cumsum(pca_out$sdev^2 / sum(pca_out$sdev^2))
plot(cumpro[0:15], xlab = "PC #", ylab = "Amount of explained variance", main
= "Cumulative variance plot")
abline(v = 6, col="blue", lty=5)
abline(h = 0.88759, col="blue", lty=5)
legend("topleft", legend=c("Cut-off @ PC6"),
       col=c("blue"), lty=5, cex=0.6)
```




```
eigen_val <- pca_out$sdev^2
plot(cumsum(eigen_val) / sum(eigen_val),
     ylim=c(0, 1))
abline(h=0.9, col="blue")
```

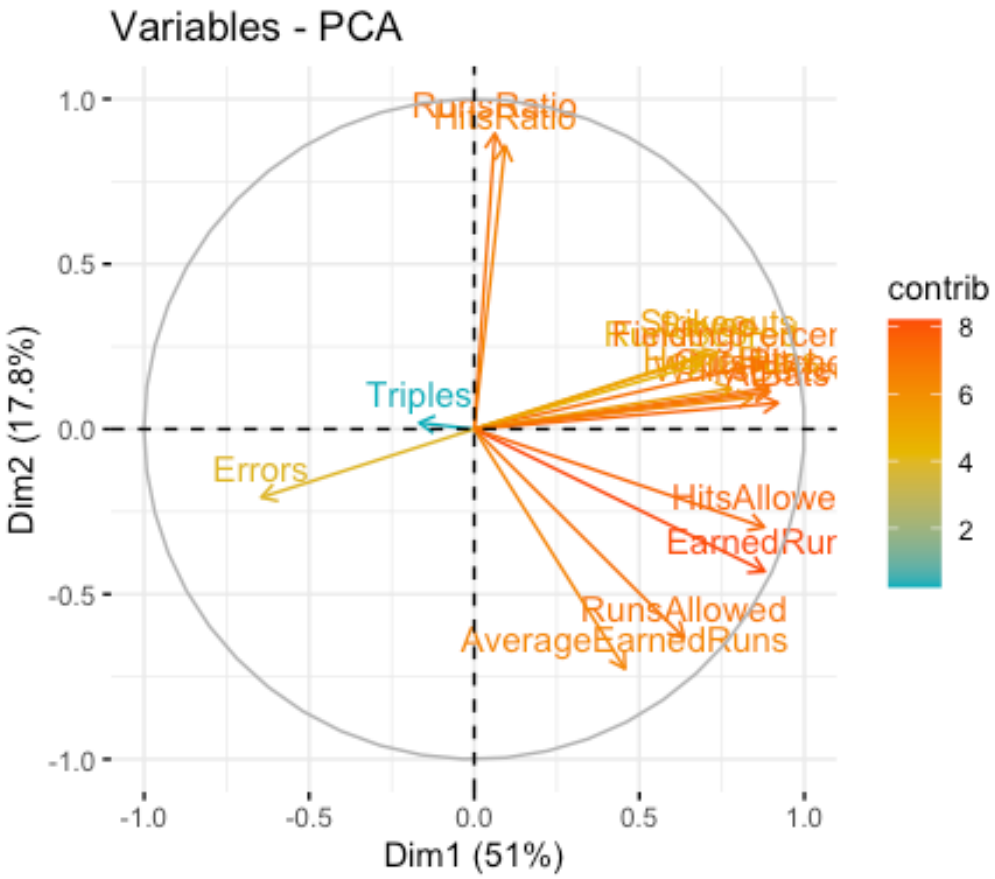



```
#We would want to pick the smallest k above 0.9, because eyeballing for a par
ticular value on this plot would be next to impossible given the sheer number
of pieces to the PCA.


hi <- cumsum(eigen_val) / sum(eigen_val)
which(hi > 0.9)[[1]]

## [1] 4

#The cumulative variance plot shows us the amount of cumulative variance expl
ained as the number of principal components increases.



#based on the Scree Plot, and the plotting of the eigenvalues, we can see tha
t that 4 is the smallest number of principal componentsfor which the ratio of
cumulative sdev^2 over the total sdev^2 is at least 0.9. Therefore, we'll pro
ceed with k=4.

k <- 4

#Let's look at what the top features are in each principal component.
```

```
for (i in 1:4) {
print(paste("For principal component", i))
print(head(pca_out$rotation[,i][order(pca_out$rotation[,i],decreasing=TRUE)],
6))
}
```

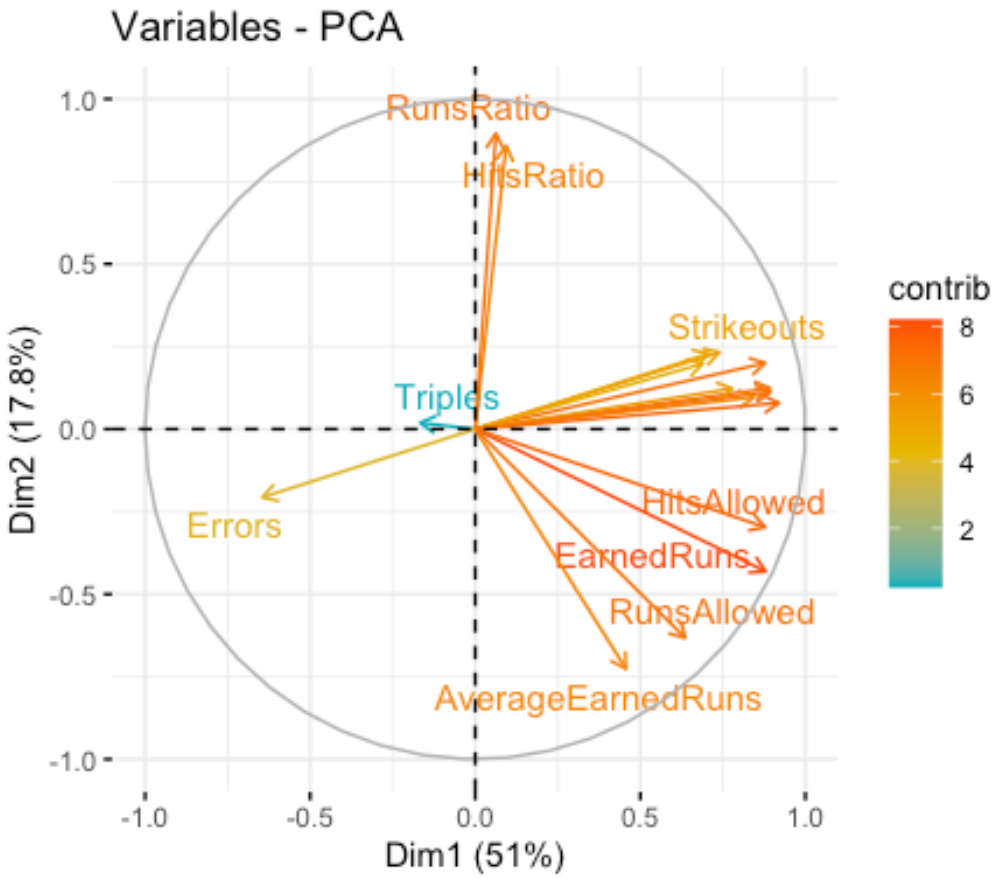
```
## [1] "For principal component 1"
##             AtBats        OutsPitched               Hits         EarnedRun
s
##          0.3124306          0.3038375          0.3033817          0.298667
0
## FieldingPercentage        HitsAllowed
##          0.2982498          0.2982491
## [1] "For principal component 2"
##          RunsRatio          HitsRatio         Strikeouts              Save
s
##          0.5145842          0.4931843          0.1337358          0.131447
7
## FieldingPercentage         RunsScored
##          0.1149050          0.1139118
## [1] "For principal component 3"
##              Saves         Strikeouts           HomeRuns FieldingPercentag
e
##         0.33190744         0.27621500         0.25011778         0.0699479
7
##  AverageEarnedRuns         EarnedRuns
##         0.04110540        -0.02928311
## [1] "For principal component 4"
##        OutsPitched FieldingPercentage            AtBats        HitsAllowe
d
##         0.36861992         0.29095992         0.26242093         0.1593887
2
##            Triples               Hits
##         0.14091351         0.03678373
```

```
fviz_pca_var(pca_out,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")   # Avoid tex
t overlapping
             )
```




```
fviz_pca_var(pca_out,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07") ,
             repel = TRUE# Avoid text overlapping
             )
```

```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
summary(pca_out)

## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5     PC6     PC
7
## Standard deviation     2.9450 1.7399 1.6091 1.02328 0.70992 0.58133 0.4945
2
## Proportion of Variance 0.5102 0.1781 0.1523 0.06159 0.02965 0.01988 0.0143
9
## Cumulative Proportion  0.5102 0.6883 0.8406 0.90217 0.93182 0.95170 0.9660
8
##                            PC8     PC9    PC10    PC11    PC12    PC13
PC14
## Standard deviation     0.41756 0.38500 0.30194 0.27106 0.18737 0.15880 0.1
2580
## Proportion of Variance 0.01026 0.00872 0.00536 0.00432 0.00207 0.00148 0.0
0093
## Cumulative Proportion  0.97634 0.98506 0.99042 0.99474 0.99681 0.99829 0.9
9922
##                           PC15    PC16    PC17
## Standard deviation     0.0824 0.05972 0.05355
## Proportion of Variance 0.0004 0.00021 0.00017
## Cumulative Proportion  0.9996 0.99983 1.00000
```

For the purposes of analysis, we will only be including variables with values of above 0.1 in the rotational description for each Principal Component.

Principal Component 1 includes:

At Bats, Outs Pitched, Hits Scored, Earned Runs, Fielding Percentage, and Hits Allowed.

At Bats can be considered a metric of offensive skill, since it represents the number of times anbatter puts the ball in play intending to get on base. Hits Scored can similarly considered an offensive skill, since it relates to a score that can only happen when a team is in an offensive position.

This principal component alone was able to explain 51.02% of the variance within the data.

Outs Pitched, Fielding Percentage, Earned Runs and Hits Allowed would all be considered metrics of defensive performance.

This Principal Component groups together a number of variables that are "standard" metrics of a teams performance – they're some of the most common metrics used to evaluate a team's success both offensively and defensively, but are rarely analyzed by themselves in the field of sports analysis. Their grouping together indicates that there could be some latent factor that ties these variables together and is able to explain some of the variance in these variables.

Principal Component 2 includes:

Runs Ratio, Hits Ratio, Runs Scored, Strikeouts, Saves, and Fielding Percentage.

Strikeouts are a measure of poor offensive performance (a higher number of strikeouts indicates more poor performances by batters during their time at bat.)

Fielding Percentage, Saves, and Runs are measures of defensive performance.

Runs Ratio and Hits Ratio are the two features we engineered - it's worth noting they don't appear in any of the other principal components in any significant form. These should be considered neutral statistics, since they combine both offensive and defensive features, and the fact that they are the most heavily loaded in variables for this factor could indicate that this is a "hybrid" principal factor that combined offensive and defensive elements.

This principal component alone was able to explain 17.81% of the variance within the data.

Principal Component 3 includes:

Saves, Strikeouts, and Homeruns

Homeruns would be considered an offensive metric, while Saves and Strikeouts would be considered defensive metrics.

One thing that all three of these metrics share in common is that they are potentially game deciding events. A strikeout decides when a batter is done with his time at bat. A save is

awarded to a pitcher when they win the game for the team in just one inning while his team is winning by three of fewer runs. A home run is usually accompanied by a large surge in runs scored, and some home-runs are termed “walk-off home runs” in that they literally end the game.

This principal component alone was able to explain 15.23% of the variance within the data.

Principal Component 4 includes:

Outs Pitched, Fielding Percentage, At Bats, Triples, and Hits Allowed.

Fielding Percentage, Hits Allowed, and Outs Pitched are all metrics of defensive performance.

At Bats and Triples are metrics of offensive performance.

Something notable about the defensive statistics in this principal component is that they all have to do with how much the defending team allows the other team to score.

This principal component alone was able to explain 6.159% of the variance within the data.

All four principal components together explain 90.217% of the variance in the data.

Principal Component Analysis + OLS

Next, we’ll be conducting ordinary least-squares regression using the principal components we just generated from our PCA analysis. We’ll be attempting to predict/classify whether a team is a World Series Winner or not using the four principal components from above.

```
w <- pca_out$x[,1:4]
df_w <- data.frame(response_train, w)
my_ols <- lm(response_train ~ ., df_w)


lm_summary <- as.data.frame(summary(my_ols)$coefficients)
lm_summary
```

```
##                 Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)  0.040998217 0.004029534 10.174431 8.421602e-24
## PC1          0.003102752 0.001368558  2.267170 2.347483e-02
## PC2          0.029320158 0.002316408 12.657595 1.622053e-35
## PC3         -0.006975129 0.002504756 -2.784754 5.401890e-03
## PC4         -0.012034658 0.003938745 -3.055455 2.273700e-03
```

```
# sort features according to p-vals
features<- head(row.names(lm_summary[order(lm_summary$`Pr(>|t|)`, decreasing=
```

```
F),]))
features
```

```
## [1] "PC2"         "(Intercept)" "PC4"         "PC3"         "PC1"
```

```
#transforming our test set using the PCA center and Scale
transformed <- function(x) (x-pca_out$center)/pca_out$scale

PCA_scaled <- apply(test_scaled, 1, transformed)

test_W <- t(PCA_scaled) %*%  pca_out$rotation

my_ols_pred <- predict(my_ols, as.data.frame(test_W))

ols_error <- mean(response_test != round(my_ols_pred))
ols_error
```

```
## [1] 0.04278075
```

```
ols_cm <- confusionMatrix(as.factor(round(my_ols_pred)), reference = as.facto
r(response_test), positive = "1")

ols_cm
```

```
## Confusion Matrix and Statistics
##
##                Accuracy : 0.9572
##                  95% CI : (0.937, 0.9724)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.554
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 2.668e-06
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9572
##          Neg Pred Value :    NaN
##              Prevalence : 0.9572
##          Detection Rate : 0.9572
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 1
##
```

```
ols_cm$byClass[c(5,6)]
```

```
## Precision    Recall
## 0.9572193 1.0000000
```

We have an extremely high level of precision and recall with this model – this could be because the chances of winning the World Series are incredibly low. Our model has a recall of 1, which means it identified all positive cases (positive in this case, being the outcome of winning the World Series). However, these results might be stunted due to the extremely low ratio of World Series Winners to Losers. For now, this model massively outperforms the precision and recall of the random predictions from above.

Naïve Bayes Classifier

Next, we'll be attempting to classify our dataset using a Naïve Bayes classifier. This is a fairly simple classification algorithm based on Bayes' Theorem. It assumes that every given feature in a dataset is unrelated to and independent of all other features.

```
set.seed(400)


library(e1071)
train_scaled_variables_cluster <- train_scaled_variables
train_scaled_variables_cluster$Cluster <- train_scaled$Cluster
testClusters <- c(Baseball[winners_row_nums, ]$Cluster,Baseball[losers_row_nu
ms, ]$Cluster)

test_scaled_cluster <- test_scaled
test_scaled_cluster$Cluster <- testClusters


nb_X <- naiveBayes(response_train ~., data=train_scaled_variables_cluster, la
place = 100)


p_hat <- predict(nb_X, newdata=test_scaled_cluster, type = "raw")
head(p_hat)

##                 0         1
## [1,] 0.070025722 0.9299743
## [2,] 0.231139190 0.7688608
## [3,] 0.000749801 0.9992502
## [4,] 0.023695956 0.9763040
## [5,] 0.010788723 0.9892113
## [6,] 0.014633031 0.9853670

head(p_hat[,2])

## [1] 0.9299743 0.7688608 0.9992502 0.9763040 0.9892113 0.9853670

#Cut off

predicted_y = ifelse(p_hat > 0.5, 1, 0)
```

```
head(predicted_y)
```

```
##      0 1
## [1,] 0 1
## [2,] 0 1
## [3,] 0 1
## [4,] 0 1
## [5,] 0 1
## [6,] 0 1
```

```
sum(predicted_y[,1])
```

```
## [1] 410
```

```
sum(predicted_y[,2])
```

```
## [1] 151
```

```
#The predict function returns values close to 1 or 0, so it's difficult to de
termine an opticmal cutoff - this model also predicts primarily ones, so it's
unclear whether this model will be optimal in terms of predictive power.


#Predictions for Naive Bayes for X 2

pred_nb <- predict(nb_X, newdata=test_scaled_cluster)

#calculating error

NBErrorX <- mean(response_test != pred_nb)
nb_cmX <- confusionMatrix(pred_nb, reference = as.factor(response_test),
positive = "1")


nb_cmX
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 391  19
##          1 146   5
##
##                Accuracy : 0.7059
##                  95% CI : (0.6663, 0.7433)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.018
##
```

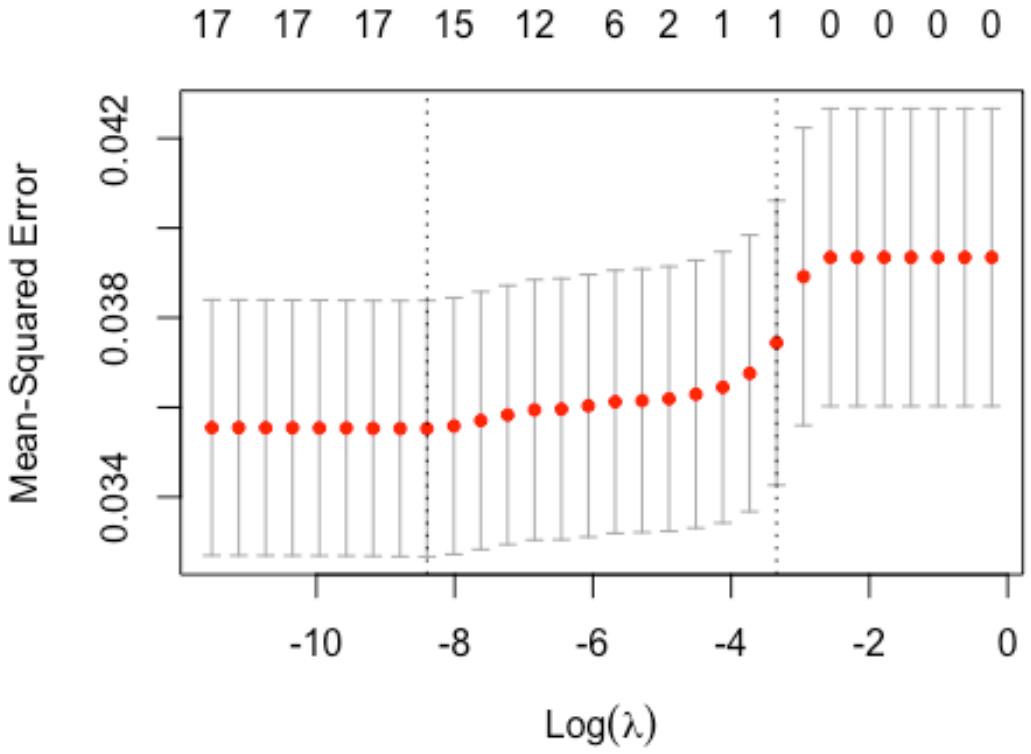
```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.208333
##             Specificity : 0.728119
##          Pos Pred Value : 0.033113
##          Neg Pred Value : 0.953659
##              Prevalence : 0.042781
##          Detection Rate : 0.008913
##    Detection Prevalence : 0.269162
##       Balanced Accuracy : 0.468226
##
##        'Positive' Class : 1
##

NBErrorX

## [1] 0.2941176

nb_cmX$byClass[c(5,6)]

##  Precision     Recall
## 0.03311258 0.20833333
```

Comparative to our random guesses, our precision using the Naïve Bayes model is less than random selection, but our recall is nearly twice as high! However, our PCA + OLS model is still preferable to this one.

LASSO

The next method we'll be using to model our data is LASSO. LASSO (Least Absolute Shrinkage and Selection Operator) encourages sparse models and "shrinks" data values towards a centralized point. LASSO tends to create models with less predictors by penalizing the least important features based on their magnitude.

```
lasso.cv <- cv.glmnet(as.matrix(train_scaled_variables), response_train,
                      lambda = 10^seq(-5, -0.1, length.out = 30),
                      alpha=1, standardize=T)
plot(lasso.cv)
```




```
which_lambda <- which(lasso.cv$lambda == lasso.cv$lambda.1se)
best <- lasso.cv$glmnet.fit$beta[, which_lambda]
vector <- as.vector(best)
head(vector)

## [1] 0 0 0 0 0 0

lasso.cv$glmnet.fit$beta[, which_lambda]

##               Hits         RunsScored        WalksBatted               Save
s
##          0.0000000          0.0000000          0.0000000          0.000000
0
## FieldingPercentage         Strikeouts        RunsAllowed         EarnedRun
s
##          0.0000000          0.0000000          0.0000000          0.000000
0
##  AverageEarnedRuns            Triples              AtBats           HomeRun
s
##          0.0000000          0.0000000          0.0000000          0.000000
0
##        HitsAllowed        OutsPitched             Errors          HitsRati
o
##          0.0000000          0.0000000          0.0000000          0.000000
```

```
0
##          RunsRatio
##          0.2330872
```

```
prediction <- predict(lasso.cv, s=lasso.cv$lambda.1se, newx=as.matrix(test_sc
aled), standardize=T)
lasso_error <- mean(response_test != round(prediction))
lasso_confusion <- confusionMatrix(factor(round(prediction)), reference = as.
factor(response_test), positive ="1")
```

```
lasso_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 537  24
##          1   0   0
##
##                Accuracy : 0.9572
##                  95% CI : (0.937, 0.9724)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.554
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 2.668e-06
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.95722
##              Prevalence : 0.04278
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
lasso_error
```

```
## [1] 0.04278075
```

Unfortunately, our LASSO model penalized our other predictors so heavily that the only one it included was the Runs Ratio variable. In addition, our output did not closely match the test output. Because our LASSO model didn't correctly predict any of the World Series Winners (it didn't predict that any of the test teams would be World Series Winners), we are unable to calculate precision due to not having a positive prediction proportion.

However, our model predicted the negative values (people who did not win the World Series) with 95.722% accuracy.

Since our LASSO didn't end up working the way we wanted it to, let's try multivariate logistic regression.

Logistic Regression

```
drop <- c("row_num")
df = train_scaled[,!(names(train_scaled) %in% drop)]

logreg <- glm(ws_win ~ ., data=df, family=binomial(link="logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logreg)

##
## Call:
## glm(formula = ws_win ~ ., family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8570  -0.2129  -0.0781  -0.0158   3.3381
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         50.7842    17.3084   2.934  0.00335 **
## Hits                41.0658    24.7414   1.660  0.09695 .
## RunsScored          29.3939    13.9015   2.114  0.03448 *
## WalksBatted         -1.3880     2.1374  -0.649  0.51610
## Saves                1.3811     1.2112   1.140  0.25418
## FieldingPercentage -10.5675     8.4566  -1.250  0.21144
## Strikeouts          -0.1091     1.5248  -0.072  0.94295
## RunsAllowed        -32.8052    20.4549  -1.604  0.10876
## EarnedRuns          49.3741    24.8865   1.984  0.04726 *
## AverageEarnedRuns  -50.3204    24.4677  -2.057  0.03972 *
## Triples              4.0147     1.5860   2.531  0.01136 *
## AtBats             -15.2537    13.4439  -1.135  0.25654
## HomeRuns             0.5563     1.4769   0.377  0.70644
## HitsAllowed        -59.7521    29.7907  -2.006  0.04489 *
## OutsPitched         -2.7897    14.9641  -0.186  0.85211
## Errors              -8.1180     5.6753  -1.430  0.15260
## HitsRatio          -48.5833    28.4336  -1.709  0.08751 .
## RunsRatio           -9.2048    14.3236  -0.643  0.52046
## Cluster2            -0.4715     0.5979  -0.789  0.43030
## Cluster3            -0.3275     0.8427  -0.389  0.69754
## Cluster4            -1.0335     0.8121  -1.273  0.20315
```

```
## Cluster5           -32.0768   634.3615  -0.051  0.95967
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.91  on 2243  degrees of freedom
## Residual deviance: 484.03  on 2222  degrees of freedom
## AIC: 528.03
##
## Number of Fisher Scoring iterations: 19
```

As we can see, at an alpha level of 0.1, the only predictors that are statistically significant are HitsRatio, HitsAllowed, Triples, AverageEarnedRuns, EarnedRuns, RunsScored, and Hits. I'll make a model that includes only those variables.

```
logreg2 <- glm(ws_win ~ HitsRatio + HitsAllowed + Triples+ AverageEarnedRuns+
EarnedRuns+ RunsScored + Hits, data=df, family=binomial(link="logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logreg2)

##
## Call:
## glm(formula = ws_win ~ HitsRatio + HitsAllowed + Triples + AverageEarnedRu
ns +
##     EarnedRuns + RunsScored + Hits, family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7030  -0.2646  -0.1236  -0.0451   3.6290
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        18.6958     4.9628   3.767 0.000165 ***
## HitsRatio         -46.3247     9.6228  -4.814 1.48e-06 ***
## HitsAllowed       -65.7864    10.9135  -6.028 1.66e-09 ***
## Triples            -0.2127     1.0560  -0.201 0.840380
## AverageEarnedRuns -34.2501     6.7876  -5.046 4.51e-07 ***
## EarnedRuns         23.6016     7.1693   3.292 0.000995 ***
## RunsScored         12.6637     2.2668   5.587 2.32e-08 ***
## Hits               48.4681     8.5497   5.669 1.44e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.91  on 2243  degrees of freedom
## Residual deviance: 550.69  on 2236  degrees of freedom
```

```
## AIC: 566.69
##
## Number of Fisher Scoring iterations: 8
```

Now Triples isn't statistically significant! Let's re-form our model.

```
logreg3 <- glm(ws_win ~ HitsRatio + HitsAllowed + AverageEarnedRuns+ EarnedRu
ns+ RunsScored + Hits, data=df, family=binomial(link="logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logreg3)

##
## Call:
## glm(formula = ws_win ~ HitsRatio + HitsAllowed + AverageEarnedRuns +
##     EarnedRuns + RunsScored + Hits, family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6978  -0.2646  -0.1236  -0.0451   3.6459
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         18.873      4.937   3.823 0.000132 ***
## HitsRatio          -46.575      9.640  -4.831 1.36e-06 ***
## HitsAllowed        -66.384     10.599  -6.263 3.77e-10 ***
## AverageEarnedRuns  -34.834      6.163  -5.652 1.58e-08 ***
## EarnedRuns          24.465      5.771   4.239 2.24e-05 ***
## RunsScored          12.625      2.259   5.589 2.29e-08 ***
## Hits                48.532      8.615   5.633 1.77e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 767.91  on 2243  degrees of freedom
## Residual deviance: 550.73  on 2237  degrees of freedom
## AIC: 564.73
##
## Number of Fisher Scoring iterations: 8
```

We finally have a model where all predictors are statistically significant. The low probability values for each variable tests the null hypothesis that the coefficient for that variable is equal to zero, and the p-value tells us that the probability of those variables having no effect on the response variable is extremely low. In terms of magnitude, it seems that hits allowed and Hits have the largest effect on the response variable. Now, let's use this model to predict the outcomes of the test set.

```r
logpredval <- predict(logreg3,test_scaled,type="response")
mean(logpredval)
```

```
## [1] 0.04245714
```

```r
logpredrounded <- logpredval
for (i in 1:length(logpredrounded)) {
  if (logpredrounded[i] < 0.15) { logpredrounded[i] <- 0 }
    else { logpredrounded[i] <- 1 }
}
mean(logpredrounded)
```

```
## [1] 0.07130125
```

```r
#Let's pick this optimal cutoff value because it closely mirrors the distribu
tion of victories in the overall dataset.


logpredtest <- predict(logreg3,test_scaled,type="response")
mean(logpredtest)
```

```
## [1] 0.04245714
```

```
## [1] 0.1012888
```

```r
#rounding the predictions based on the optimal cutoff found earlier
logtestrounded <- logpredtest
for (i in 1:length(logtestrounded)) {
  if (logtestrounded[i] < 0.215) { logtestrounded[i] <- 0 }
    else { logtestrounded[i] <- 1 }
}
mean(logtestrounded)
```

```
## [1] 0.04278075
```

```
## [1] 0.10576
```

```r
#Calculating Classification Error
mean(logtestrounded != response_test)
```

```
## [1] 0.08199643
```

```r
conf_log <- table(logtestrounded,response_test)
print(conf_log)
```

```
##               response_test
## logtestrounded   0   1
##              0 514  23
##              1  23   1
```

```r
#Calculating Precision
conf_log[2,2]/(conf_log[2,1]+conf_log[2,2])
```

```
## [1] 0.04166667

#Calculating Recall
conf_log[2,2]/(conf_log[1,2]+conf_log[2,2])

## [1] 0.04166667

## [1]0.04166667
```

In this case, we still have an extremely low recall and precision rate. Once again, this is likely due to the extremely low accuracy with which this model accurately predicted World Series Winners to be World Series Winners. Due to the fact that an extremely small fraction of the population ends up winning the World Series, this model and LASSO struggled with correct positive identifications but excelled at correct negative identifications (that is, they were extremely good at finding the World Series losers but not skilled at finding the World Series Winners).

Ultimately, based on the criterion of Precision and Recall, we would ultimately proceed with the model that combined Principal Component Analysis with Ordinary Least Squares regression. Below is the performance of all of our models side-by-side:

Random Choice:

Precision: 0.09523810

Recall: 0.08333333

PCA + OLS:

Precision: 0.9572193

Recall: 1.0000000

Naïve Bayes:

Precision: 0.03311258
Recall: 0.20833333

Lasso:

Unable to calculate precision and recall due to no correct positive identifications.

Logarithmic Model:

Precision: 0.04166667
Recall: 0.04166667

As we can see, the clear winner based on both metrics is the PCA + OLS classifier, which massively outperformed all other classifiers. Therefore, this algorithm has the highest probability of identifying Major League World Series Winners.

Conclusion

After cleaning our data, we decided to engineer two features into our data, one which analyzed the ratio of hits scored to hits allowed, and another which analyzed the ratio of runs scored to runs allowed. We also scaled our data using the minimum, maximum, and range of each variable. Scaling our data allows us to bring the features closer together, which is helpful in machine learning because models are able to learn and train faster when the different data points are close to and resemble each other, especially since many algorithms make decisions based on how far away different data points are. After scaling our data, we began with some exploratory analysis in the form of clustering.

With this exploratory analysis, we were attempting to uncover hidden patterns and associations between the different observations that weren't immediately obvious. Our cluster analysis, which resulted in 5 clusters, showed clear and distinct differences in performance between the clusters, which potentially indicated that clusters were representative of skill, or likelihood to perform well during a given season. However, after realizing that the proportion of World Series winners appeared to stay constant across clusters, we realized that clusters were largely distributed by year, even though year was removed from the dataset that the clusters were originally built on. The clusters represented different decades of history throughout the formation of the MLB until the present day. Following our cluster analysis, we decided to perform principal component

analysis, another type of exploratory analysis that seeks to uncover patterns, but this time between the variables rather than the observations. Essentially, principal component analysis tries to group together our different variables our features into distinct principal components, limiting the number of features and reducing the number of dimensions present in our dataset. The four factors we ultimately decided on explained more than 90% of the variance within the dataset. The associations between the variables in each factor weren't immediately obvious, since most factors had a mix of both offensive and defensive skill metrics. Principal Component 2, which explained 17.81% of the variance within the data, was the only component to contain the two engineered features. We also theorized that Principal Component 3 could be thought of as "game-ending" statistics, statistics that were especially important because they could single-handedly win a game for your team. The creation of these factors is especially important for baseball stakeholders like coaches and franchise owners since it helps them organize their chances of winning the World Series around a particular grouping of skills and statistics, so they know which metrics to focus on in order to maximize their chances of bringing home a championship. They can be cost-efficient in terms of their coaching staff, since they know what skills specifically to hone and the types of practices and drills to engage in to improve performance in those fields.

After conducting our PCA analysis and splitting our data up into training and test sets, we started with the process of using algorithms to explore the relationship of these different variables to the probability that a team would win the World Series. In order to evaluate whether these models were successful or not, we use the test set in order to use our model in an objective setting, against data it hasn't already seen. In order to evaluate

the performance of these models in those neutral settings, we used recall and precision as our two metrics. Precision tells us what proportion of our guesses for World Series winners was correct , while recall asks us what proportion of World Series Winners were correctly identified as World Series winners. We compared the precision and recall of these models to the precision and recall of randomly selecting which teams would win based on the base rate of winning the World Series (i.e, blindly picking around 5% of the observations without paying any attention to their statistics.

The types of algorithms we used included ordinary-least-squares regression using the output of our principal components analysis, Naïve Bayes classification, LASSO regression, and logarithmic regression. Basically, what we attempted to do with these algorithms was use them to classify whether or not a team would win the World Series. These algorithms studied the training data in different ways, observing the relationships of our variables or principal components to the proportion of World Series wins before fitting a model around the data. Afterwards, they would be given the test data (without knowing which of the teams represented were actually World Series winners) and then had to try to classify which teams were World Series winners using the models fit to the training data. Using our metrics of precision and recall, our ordinary least squares model fit using the principal components from earlier massively outperformed the other models. Ordinary least squares regressions attempt to minimize the sum of the squared differences between the observed variable and the estimated variable across all observations. Essentially, it tries to fit a model that minimizes the difference between our predictions and the actual value. If we were to try to predict whether or not a team would be able to win the super bowl based on a dataset similar to this one, we would use this model in order to predict

their performance. We can trust this model to perform similarly well on a new set of data, since the test dataset we used to evaluate its performance had never been seen by our model throughout the course of its training.

Critique

I critiqued Devansh Taori's (drt2131) final project, which was about classifying and predicting the quality of different types of wine. Devansh's project was similar to mine in terms of the methods he used, and the ways he played around with and manipulated his data, but the metrics he was trying to measure and predict were fairly different from mine (his response variable was on an ordinal scale.) It's possible that some of these critiques have already been accounted for in his project, since we were pretty early on in our projects when we compared.

One critique I had was regarding the PCA analysis Devansh did, which was originally eyeballed. In the case of a situation where we had many more components (like in the case of homework 3), I argued that it was generally best practice to calculate which principal component gave us a value of above 0.9, as we did in class. I thought Devansh did a really fantastic job with the clusters, and the explanation of what they meant, but I would've liked to see a breakdown of red vs white wines in the different clusters, or whether some clusters saw higher quality wine values, etc.

My other comment was primarily about many of the data visualizations in the project – given that he was analyzing two separate groups (red and white wines) to determine whether they were crucial differences in their properties, many of the data visualizations showed general spread and distribution for all values. In this case, it would be potentially interesting to use color or transparency to distinguish between red and white wines (using some sort of identifier when he merged the two datasets) to distinguish between the two groups. This would be helpful both in proving some of his arguments about in-group and out-of-group differences, and could also potentially have exposed some patterns that were not originally seen.

My last comment was just that I think more background information on what the different variables were and meant would have been helpful, potentially as an appendix item or at the very beginning of the document, since I am not super familiar with the terminology behind wine (although I do enjoy drinking it.) At times, I had to Google what different words meant in the context of wine, which could defeat the purpose of making the report readable to an outsider.