NBA Data Science Project

Dennis Li

2018-08-12

Intro

The 2 questions I hoped to answer that motivated me to do this project were:

1. How well I could detect certain clusters of players (superstars, starters, role players, prospects etc.) from the 2017-18 regular season using dimensionality reduction and k-means cluster analysis
2. How strong of a predictive model I could create to predict an NBA player's position (traditional PG/SG/SF/PF/C) based on per-36 stats, height, weight etc. given that the NBA is transitioning to a more "positionless" league. Also would like to see how strong of a model I could make to predict a binary position classification (guard/wing, big)

Loading Packages

```r
library(ggplot2)
library(kableExtra)
library(corrplot)
library(tidyverse)
library(cluster)
library(factoextra)
library(gridExtra)
library(dplyr)
library(tree)
library(randomForest)
library(xgboost)
library(caret)
library(glmnet)
library(nnet)
library(reshape2)
library(foreign)
library(Metrics)
library(e1071)
```

Data Collection/Cleaning

For each of the past 4 seasons, the "seasonStats" dataset contains individual per-36 minute and per-game player statistics for every player who appeared in a regular season game. Aggregate statistics are used for players traded during that season and the team that the traded player is assigned to is whichever he played more minutes for. Only players who played over 20 games are used. Per-36 stats are included to minimize bias against players whose stats may be inflated due to high playing time.

Additional data, such as height, weight, nationality and experience are also included.

Pos2 is the binary position classification. Guards/SF's are separated from PF/C's.

All data comes from basketball-reference.com.

2017-2018 Regular Season

```r
perGame_17 <- read.csv("perGame17.csv")
perGame_17$Player <- gsub("\\\\.*", "", perGame_17$Player)
perGame_17 <- perGame_17[perGame_17$G >= 20,]

per36_17<- read.csv("per36_17.csv")
per36_17$Player <- gsub("\\\\.*", "", per36_17$Player)

seasonStats17 <- merge(x = perGame_17, y = per36_17, by = "Player")


body17 <- read.csv("nbaBody17.csv")
body17$Player <- gsub("\\\\.*", "", body17$Player)
names(body17)[7] <- "Nationality"
body17 <- body17[,c(2,4,5,7,8)]
body17 <- unique(body17)

#Assign rookies 0 years of experience rather than "R"
body17$Exp <- as.numeric(levels(body17$Exp))[body17$Exp]
for(i in 1:nrow(body17)){
  if(is.na(body17$Exp[i]) == TRUE){
    body17$Exp[i] <- 0
  }
}

#Convert height from feet-inches to inches
input <- body17$Ht
feet <- substr(input, start = 1, stop = 1)
inches <- substr(input, start = 3, stop = 4)
feet <- as.integer(feet)
inches <- as.integer(inches)
output <- feet*12 + inches
body17$Ht <- output


#Joining data
nba17 <- merge(x = seasonStats17, y = body17, by = "Player")




#Add Pos2 column
nba17$Pos2 <- with(nba17, ifelse(nba17$Pos == "PF" | nba17$Pos == "C", "Big", "Guard/Wing"))

#Add US column
nba17$USA <- with(nba17, ifelse(nba17$Nationality == "us", "Yes", "No"))

nba17[, "Pos2"] <- as.factor(nba17[, "Pos2"])
nba17[, "USA"] <- as.factor(nba17[, "USA"])
```

2016-2017 Regular Season

```r
perGame_16 <- read.csv("perGame16.csv")
perGame_16$Player <- gsub("\\\\.*", "", perGame_16$Player)
```

```r
perGame_16 <- perGame_16[perGame_16$G >= 20,]

per36_16<- read.csv("per36_16.csv")
per36_16$Player <- gsub("\\\\.*", "", per36_16$Player)

seasonStats16 <- merge(x = perGame_16, y = per36_16, by = "Player")


body16 <- read.csv("nbaBody16.csv")
body16$Player <- gsub("\\\\.*", "", body16$Player)
names(body16)[7] <- "Nationality"
body16 <- body16[,c(2,4,5,7,8)]
body16 <- unique(body16)

#Assign rookies 0 years of experience rather than "R"
body16$Exp <- as.numeric(levels(body16$Exp))[body16$Exp]
for(i in 1:nrow(body16)){
  if(is.na(body16$Exp[i]) == TRUE){
    body16$Exp[i] <- 0
  }
}

#Convert height from feet-inches to inches
input <- body16$Ht
feet <- substr(input, start = 1, stop = 1)
inches <- substr(input, start = 3, stop = 4)
feet <- as.integer(feet)
inches <- as.integer(inches)
output <- feet*12 + inches
body16$Ht <- output


#Joining data
nba16 <- merge(x = seasonStats16, y = body16, by = "Player")


#Add Pos2 column
nba16$Pos2 <- with(nba16, ifelse(nba16$Pos == "PF" | nba16$Pos == "C", "Big", "Guard/Wing"))

#Add US column
nba16$USA <- with(nba16, ifelse(nba16$Nationality == "us", "Yes", "No"))
nba16[, "Pos2"] <- as.factor(nba16[, "Pos2"])
nba16[, "USA"] <- as.factor(nba16[, "USA"])
```

2015-2016 Regular Season

```r
perGame_15 <- read.csv("perGame15.csv")
perGame_15$Player <- gsub("\\\\.*", "", perGame_15$Player)
perGame_15 <- perGame_15[perGame_15$G >= 20,]

per36_15<- read.csv("per36_15.csv")
per36_15$Player <- gsub("\\\\.*", "", per36_15$Player)
```

```
seasonStats15 <- merge(x = perGame_15, y = per36_15, by = "Player")


body15 <- read.csv("nbaBody15.csv")
body15$Player <- gsub("\\\\.*", "", body15$Player)
names(body15)[7] <- "Nationality"
body15 <- body15[,c(2,4,5,7,8)]
body15 <- unique(body15)

#Assign rookies 0 years of experience rather than "R"
body15$Exp <- as.numeric(levels(body15$Exp))[body15$Exp]
for(i in 1:nrow(body15)){
  if(is.na(body15$Exp[i]) == TRUE){
    body15$Exp[i] <- 0
  }
}

#Convert height from feet-inches to inches
input <- body15$Ht
feet <- substr(input, start = 1, stop = 1)
inches <- substr(input, start = 3, stop = 4)
feet <- as.integer(feet)
inches <- as.integer(inches)
output <- feet*12 + inches
body15$Ht <- output


#Joining data
nba15 <- merge(x = seasonStats15, y = body15, by = "Player")


#Add Pos2 column
nba15$Pos2 <- with(nba15, ifelse(nba15$Pos == "PF" | nba15$Pos == "C", "Big", "Guard/Wing"))

#Add US column
nba15$USA <- with(nba15, ifelse(nba15$Nationality == "us", "Yes", "No"))
nba15[, "Pos2"] <- as.factor(nba15[, "Pos2"])
nba15[, "USA"] <- as.factor(nba15[, "USA"])
```

2014-2015 Regular Season

```
perGame_14 <- read.csv("perGame14.csv")
perGame_14$Player <- gsub("\\\\.*", "", perGame_14$Player)
perGame_14 <- perGame_14[perGame_14$G >= 20,]

per36_14<- read.csv("per36_14.csv")
per36_14$Player <- gsub("\\\\.*", "", per36_14$Player)

seasonStats14 <- merge(x = perGame_14, y = per36_14, by = "Player")


body14 <- read.csv("nbaBody14.csv")
body14$Player <- gsub("\\\\.*", "", body14$Player)
```

```
names(body14)[7] <- "Nationality"
body14 <- body14[,c(2,4,5,7,8)]
body14 <- unique(body14)

#Assign rookies 0 years of experience rather than "R"
body14$Exp <- as.numeric(levels(body14$Exp))[body14$Exp]
for(i in 1:nrow(body14)){
  if(is.na(body14$Exp[i]) == TRUE){
    body14$Exp[i] <- 0
  }
}

#Convert height from feet-inches to inches
input <- body14$Ht
feet <- substr(input, start = 1, stop = 1)
inches <- substr(input, start = 3, stop = 4)
feet <- as.integer(feet)
inches <- as.integer(inches)
output <- feet*12 + inches
body14$Ht <- output


#Joining data
nba14 <- merge(x = seasonStats14, y = body14, by = "Player")


#Add Pos2 column
nba14$Pos2 <- with(nba14, ifelse(nba14$Pos == "PF" | nba14$Pos == "C", "Big", "Guard/Wing"))

#Add US column
nba14$USA <- with(nba14, ifelse(nba14$Nationality == "us", "Yes", "No"))
nba14[, "Pos2"] <- as.factor(nba14[, "Pos2"])
nba14[, "USA"] <- as.factor(nba14[, "USA"])
```

Final 4-season data

```
nba1 <- rbind(nba14, nba15)
nba2 <- rbind(nba1, nba16)
nba <- rbind(nba2, nba17)
nba$Pos <- factor(nba$Pos,c("PG","SG","SF","PF","C"))
#Add BMI variable
nba <- nba %>%
  mutate(bmi = Wt / (Ht^2)*703)
```

Exploratory Data Analysis

Some basic summary statistics and visuals to help get a high level understanding of what features are important.

```
head(nba)[,1:10]
```

```
##          Player Pos Age  Tm  G GS   MP  FG  FGA FGpct
## 1    A.J. Price  PG  28 IND 26  0 12.5 2.0  5.3 0.372
## 2   Aaron Brooks  PG  30 CHI 82 21 23.0 4.2 10.0 0.421
## 3   Aaron Gordon  PF  19 ORL 47  8 17.0 2.0  4.4 0.447
## 4 Adreian Payne  PF  23 MIN 32 22 23.1 2.8  6.9 0.414
## 5     Al Horford   C  28 ATL 76 76 30.5 6.8 12.7 0.538
## 6   Al Jefferson   C  30 CHO 65 61 30.6 7.5 15.5 0.481
```

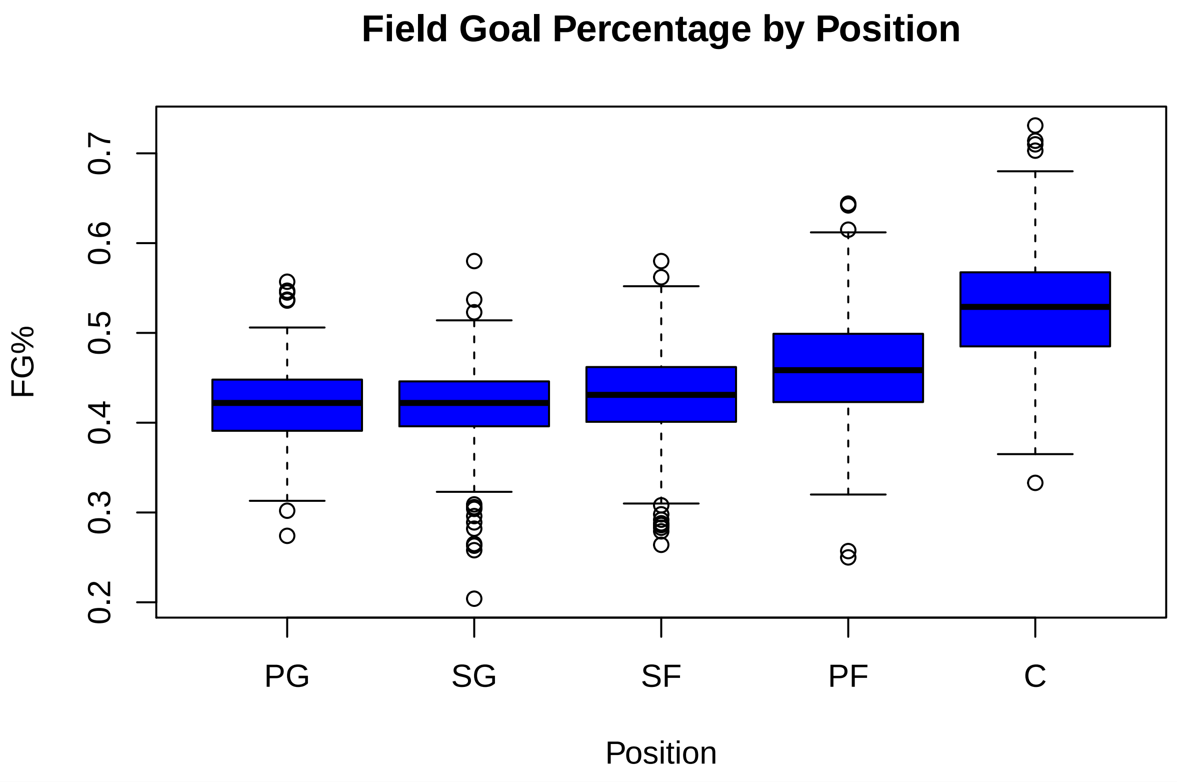
```r
dim(nba)
```

```
## [1] 1664   53
```

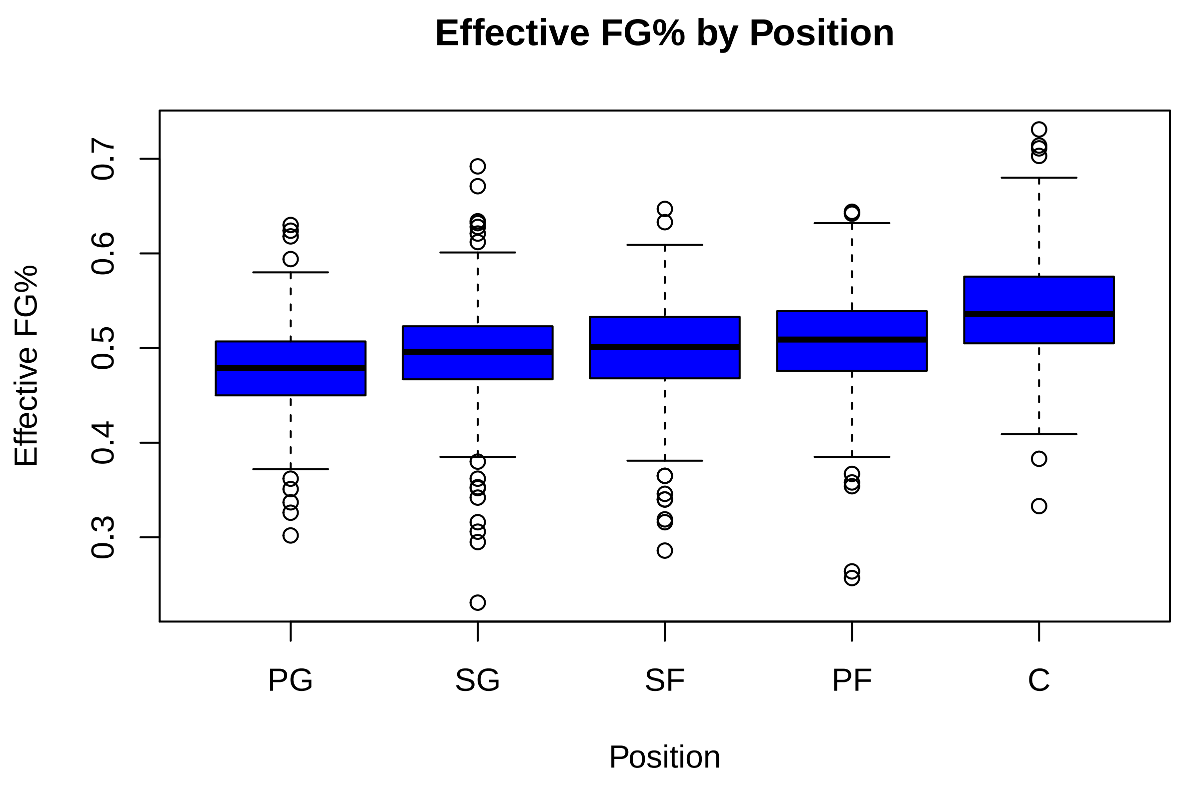
```r
summary(nba$Pos)
```

```
##  PG  SG  SF  PF   C
## 334 359 305 342 324
```

```r
boxplot(nba$FGpct~nba$Pos, col = "blue", ylab = "FG%", xlab = "Position", main = "Field Goal Percentage
```



```r
boxplot(nba$eFGpct~nba$Pos, col = "blue", ylab = "Effective FG%", xlab = "Position", main = "Effective
```

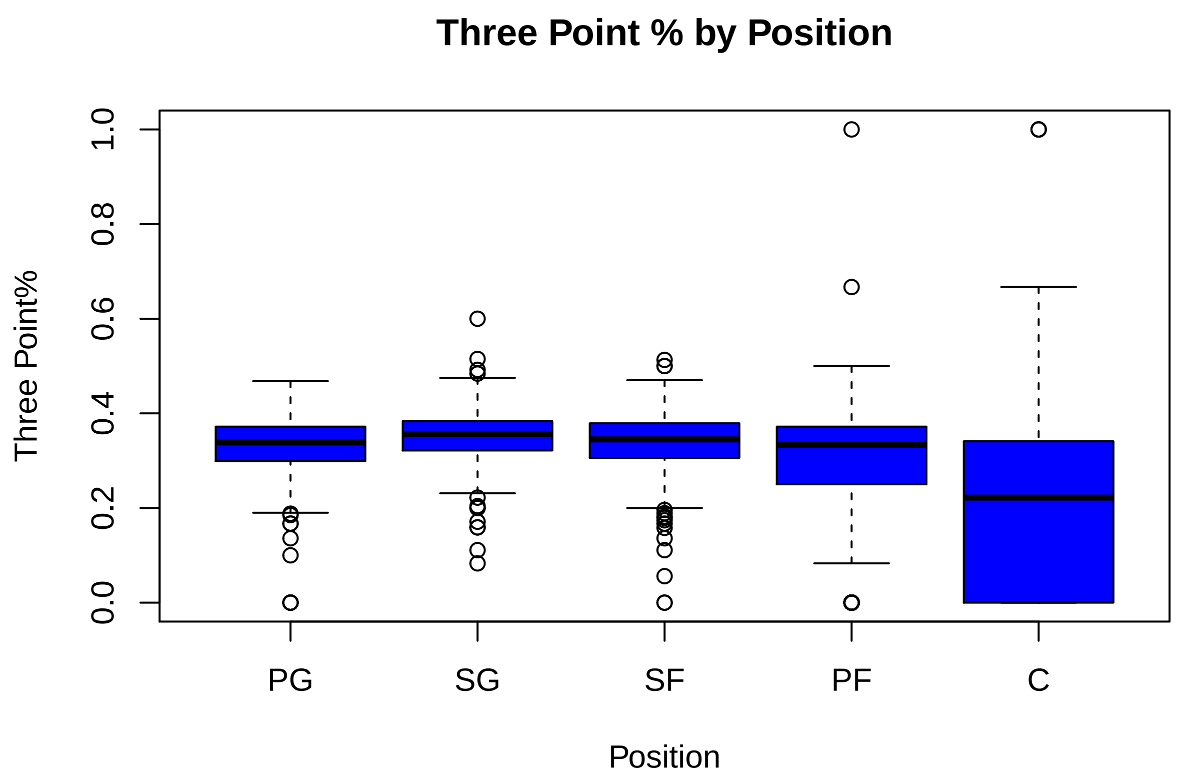


Effective FG% is a metric that takes into account that three pointers are worth more than 2 pointers (used heavily in the modern "moneyball" approach pioneered by Daryl Morey):

eFG% = (FGM + (0.5 x 3PTM)) / FG

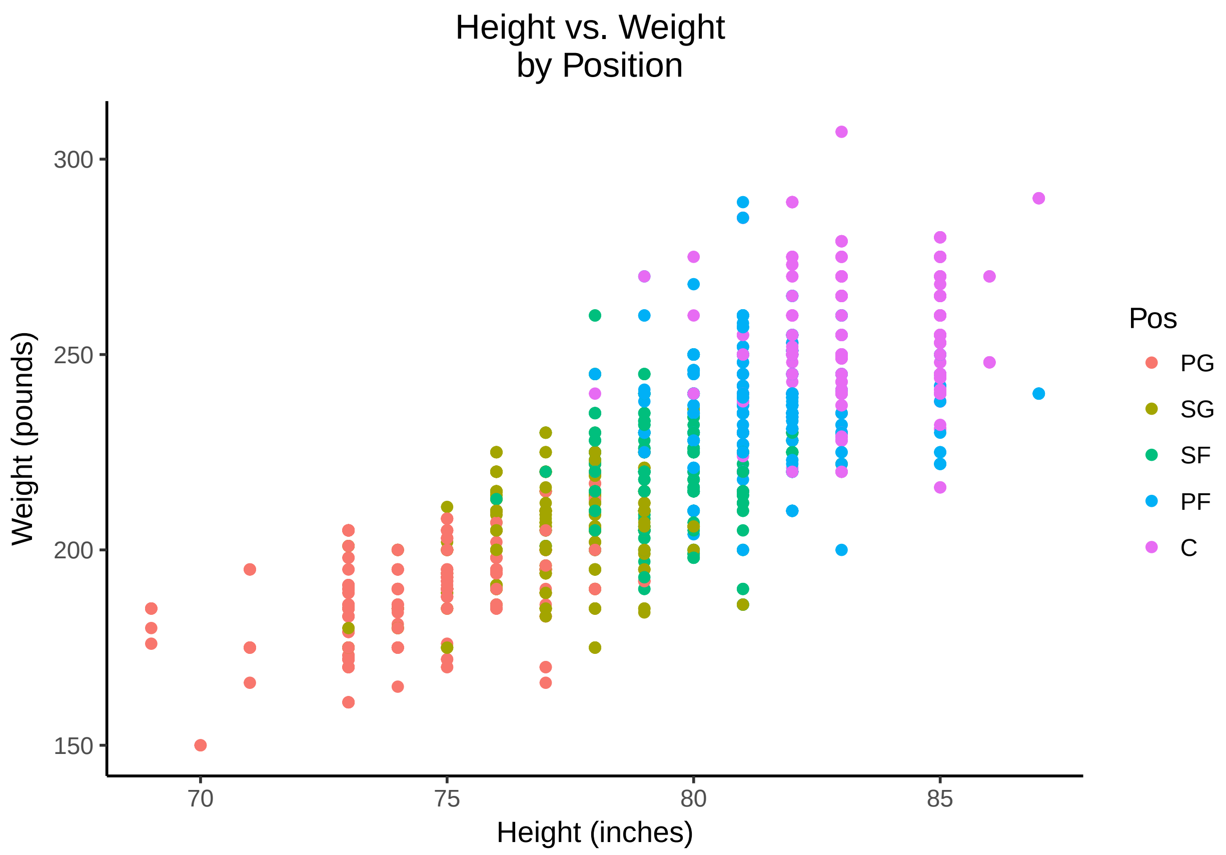
As expected, centers on average have the highest FG% and eFG%, but the edge is reduced when weighing 3's into the metric.

```
boxplot(nba$threePct~nba$Pos, col = "blue", ylab = "Three Point%", xlab = "Position", main = "Three Poi
```

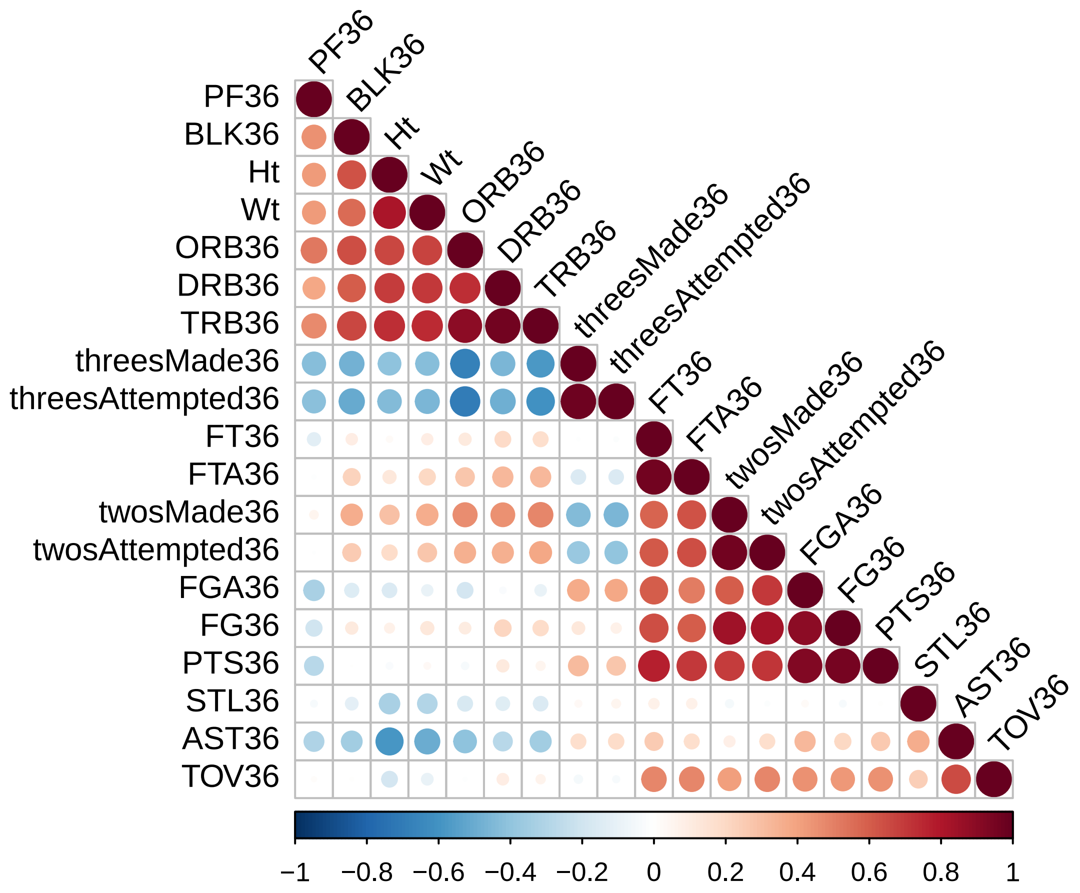


The stretch 4 is in full effect, although the variability is still larger than PG/SG/SF's as seen in the IQR. On average, it is hard to distinguish positions 1 through 4 based on 3P% in today's NBA.

```
p <- ggplot(nba, aes(x = Ht, y = Wt, color = Pos)) + geom_point() +
  labs(title="Height vs. Weight \n by Position",
       x="Height (inches)", y = "Weight (pounds)")
p + theme_classic() + theme(plot.title = element_text(hjust=0.5))
```



```r
nbaCorr <- nba[,30:48]
source("http://www.sthda.com/upload/rquery_cormat.r")
rquery.cormat(nbaCorr)
```



An easy-to-interpret correlation plot of the per-36 features. Certain features are highly correlated with many other features (such as TRB36), while others like STL36 are not highly correlated with any other features. As you can tell from this subset of the data, high correlation between variables exist. Let's move to PCA to try to transform this dataset.

K-Means Cluster Analysis and PCA

Using data from the 2017-2018 regular season, I want to see how well I can cluster the players into different categories to judge skill level.

The intuition behind using PCA and k-means clustering is that principal component analysis is a dimensionality reduction technique which does not actually perform feature selection/removal, but rather performs orthogonal transformations to project the data on a lower dimension, resulting in less multicollinearity and more interpretability without loss of information.

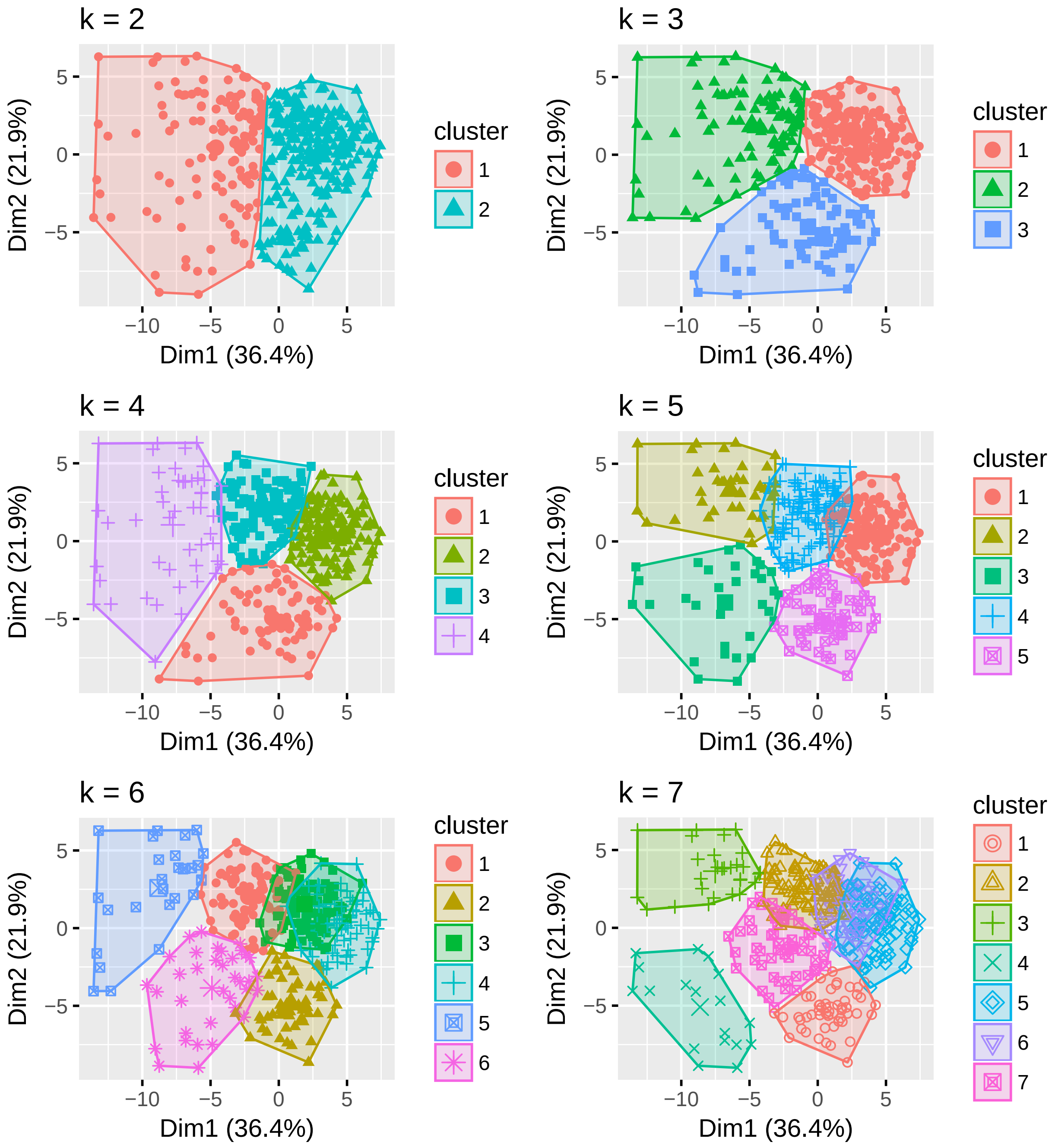
In a similar dynamic, k-means is a non-parametric algorithm to represent the data into a small number of cluster centroids. Applying PCA before k-means is (hopefully) a way to reduce the noise and improve clustering results.

I'll be using the "cluster" and "factoextra" packages to apply these techniques and cluster with 2 principal components.

```
#remove categorical predictors
nbaCluster <- nba17[,-c(2,4,49,51,52)]
nbaCluster <- na.omit(nbaCluster)
index <- sapply(nbaCluster, is.numeric)
```

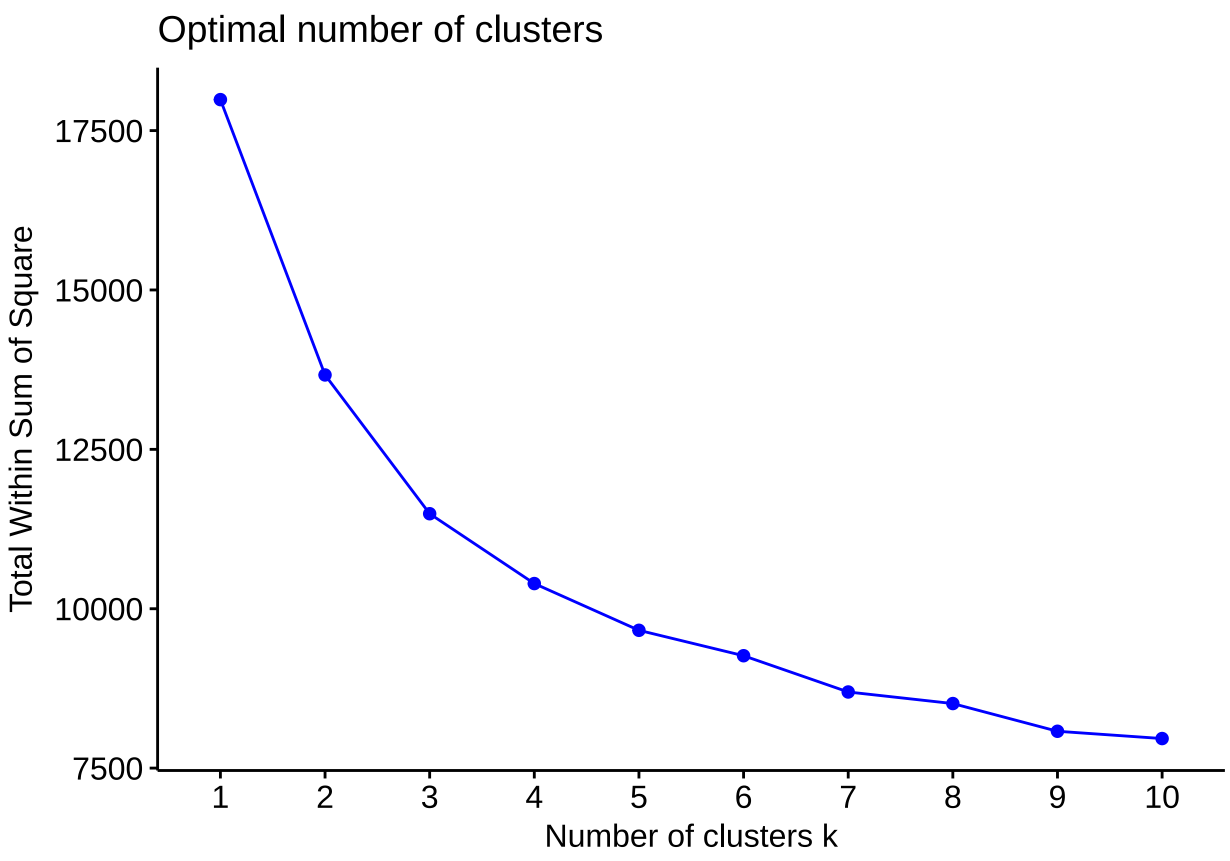
```r
nbaCluster[index] <- lapply(nbaCluster[index], scale)
#we scale because k-means and other cluster algorithms are based on distances between points (euclidean
```

```r
set.seed(999)
k2 <- kmeans(nbaCluster[,-1], centers = 2)
k3 <- kmeans(nbaCluster[,-1], centers = 3)
k4 <- kmeans(nbaCluster[,-1], centers = 4)
k5 <- kmeans(nbaCluster[,-1], centers = 5)
k6 <- kmeans(nbaCluster[,-1], centers = 6)
k7 <- kmeans(nbaCluster[,-1], centers = 7)


# Roughly comparing the clusters, seeing if there's any delinations
p2 <- fviz_cluster(k2, geom = "point", data = nbaCluster[,-1]) + ggtitle("k = 2")
p3 <- fviz_cluster(k3, geom = "point",  data = nbaCluster[,-1]) + ggtitle("k = 3")
p4 <- fviz_cluster(k4, geom = "point",  data = nbaCluster[,-1]) + ggtitle("k = 4")
p5 <- fviz_cluster(k5, geom = "point",  data = nbaCluster[,-1]) + ggtitle("k = 5")
p6 <- fviz_cluster(k6, geom = "point",  data = nbaCluster[,-1]) + ggtitle("k = 6")
p7 <- fviz_cluster(k7, geom = "point",  data = nbaCluster[,-1]) + ggtitle("k = 7")



grid.arrange(p2, p3, p4, p5, p6, p7, nrow = 3)
```



This visual shows us that k = 2, 3, 4 and 5 all do a pretty good job at making distinct clusters. However, if we want to see what the optimal number of clusters is, we can use the elbow method. This is a good way to achieve the goal of clustering the data, which is to minimize the within-cluster sum of squares variation.

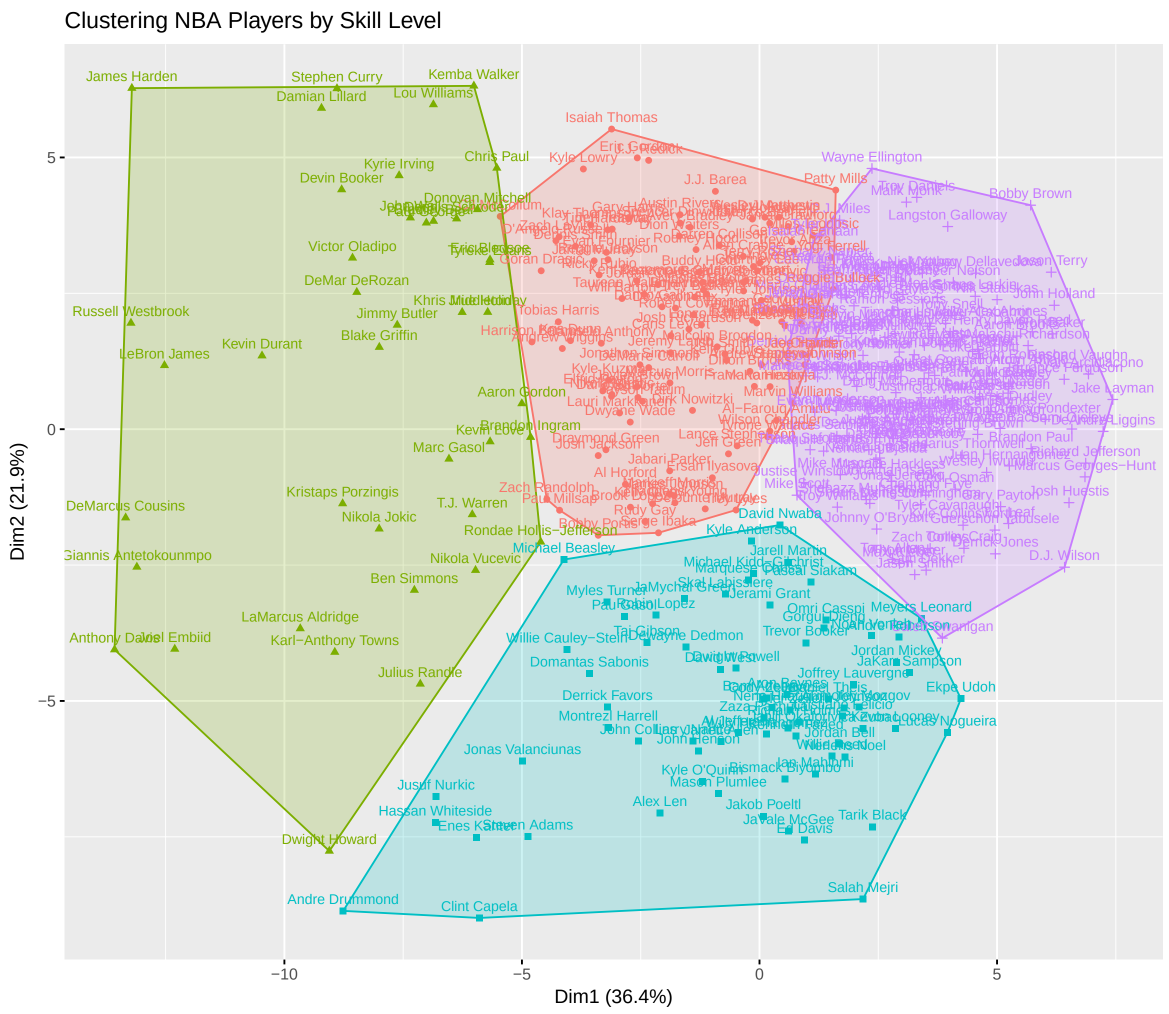
```r
set.seed(999)

fviz_nbclust(nbaCluster[,-1], kmeans, method = "wss", k.max = 10, linecolor = "blue", print.summary = T
```



The elbow plot suggests that a value around k = 4 is a good (yet arbitrary) cut-off point. The subsequent decreases in the within-cluster sum of squares don't seem to be worth the risk of overfitting. This uses factoextra's built-in function.

Optimal Clusters with Player Labels

```
optimalCluster <- kmeans(nbaCluster[,-1], 4)
nbaCluster$Cluster <- optimalCluster$cluster
rownames(nbaCluster) <- nbaCluster$Player
fviz_cluster(optimalCluster, data = nbaCluster[,-c(1,48)], labelsize = 8, show.clust.cent = FALSE, main
```



The clustering results show some interesting patterns in the data for the 2017-2018 regular season. Just from a rough visual analysis of where the players are placed, it appears that the first principal component (Dim 1, which explains 36.4% of the total variance) is a measure relating to **scoring prowess** of the players. The second principal component (Dim 2, which explains 21.9% of the total variance) is a little less clear, but it seems that guards populate the top half of the plot while big men dominate the bottom half. The k-means algorithm does not take Position (or any categorical variable) into consideration so this is probably some measure of **ball-handling ability** and **3 point abiliity**.

Some big names pop up when examining the green. Most of the players here are superstars and all-star level players. The few exceptions that jump out are Rondae Hollis-Jefferson and TJ Warren (although he has proven to be a very effective scorer, but not much else). Some others, like Julius Randle and Brandon Ingram, may make that jump next year. Mitchell and Simmons both get the nod here. Lets call this cluster **“Star”**.

The red and blue clusters which occupy the middle section of the plot are mainly filled with role players. Specifically, the red cluster is populated with ball-handling role players such as Lonzo Ball and the blue cluster is non-ball handling role players such as Robin Lopez. Some players who were categorized in these 2 clusters may not come to mind as role players, but also probably don’t deserve to be in the blue cluster (Andrew Wiggins, Isaiah Thomas, Jusuf Nurkic). Others, such as Andre Drummond and Klay Thompson were all-stars this past season and are clearly snubbed. Let’s call these players **“Ball Handling Starters/Role Player”** and **“Off Ball Starter/Role Player”**.

The purple cluster is filled with players who spent most of their time during the 2017-18 regular season riding the bench or are prospects. Many of these players are young players who have a lot of room to grow, but for the time being, players like D.J. Wilson have a lot to prove. Let's call these players **"Prospect/Bench"**.

Let's take a deeper dive into how last year's all-stars were clustered, and who was snubbed based off of this particular algorithm.

Which All Stars Were Snubbed?

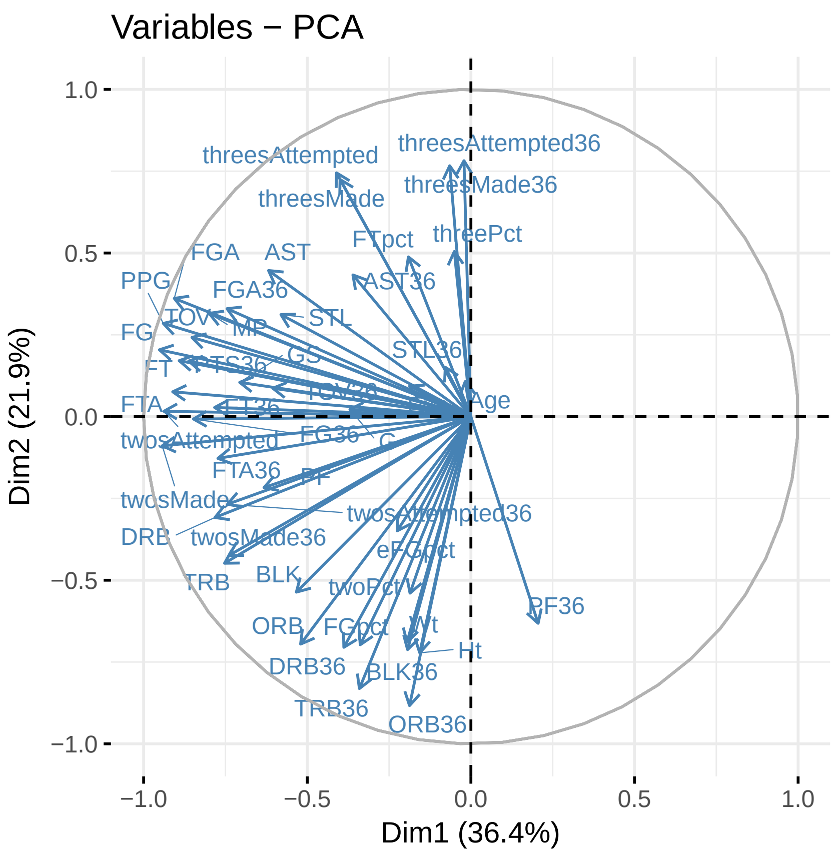
```r
allStarList <- c("LeBron James", "Kevin Durant", "Russell Westbrook", "Kyrie Irving", "Anthony Davis",
nbaCluster$AllStar <- ifelse(nbaCluster$Player %in% allStarList,"Yes", "No")
nbaCluster$Cluster <- as.factor(nbaCluster$Cluster)
#Rename clusters
levels(nbaCluster$Cluster)[levels(nbaCluster$Cluster)=="1"] <- "BallHandlingStarterRolePlayer"
levels(nbaCluster$Cluster)[levels(nbaCluster$Cluster)=="3"] <- "OffBallStarterRolePlayer"
levels(nbaCluster$Cluster)[levels(nbaCluster$Cluster)=="2"] <- "Star"
levels(nbaCluster$Cluster)[levels(nbaCluster$Cluster)=="4"] <- "ProspectBench"

ClusterTable <- nbaCluster[,c(1, 48, 49)]
AllStarClusterTable <- ClusterTable[ClusterTable$AllStar == "Yes",]


rownames(AllStarClusterTable) <- NULL
kable(AllStarClusterTable[,c(1,2)]) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Player	Cluster
Al Horford	BallHandlingStarterRolePlayer
Andre Drummond	OffBallStarterRolePlayer
Anthony Davis	Star
Bradley Beal	Star
Damian Lillard	Star
DeMar DeRozan	Star
DeMarcus Cousins	Star
Draymond Green	BallHandlingStarterRolePlayer
Giannis Antetokounmpo	Star
Goran Dragic	BallHandlingStarterRolePlayer
James Harden	Star
Jimmy Butler	Star
Joel Embiid	Star
John Wall	Star
Karl-Anthony Towns	Star
Kemba Walker	Star
Kevin Durant	Star
Kevin Love	Star
Klay Thompson	BallHandlingStarterRolePlayer
Kristaps Porzingis	Star
Kyle Lowry	BallHandlingStarterRolePlayer
Kyrie Irving	Star
LaMarcus Aldridge	Star
LeBron James	Star
Paul George	Star
Russell Westbrook	Star
Stephen Curry	Star
Victor Oladipo	Star

Out of the 28 all-stars, 22 of them were placed in the “Star” cluster. Kyle Lowry, Klay Thompson, Goran Dragic, Draymond Green, Andre Drummond and Al Horford (point forward?) were snubbed. Besides Klay and Drummond, who was a monster pre-Griffin trade, the rest may not have deserved it based on numbers.

Understanding the Principal Components

```r
set.seed(999)
nbaPCA <- prcomp(nbaCluster[,-c(1,48, 49)],  scale = TRUE)
fviz_pca_var(nbaPCA, col.var = "steel blue",labelsize = 3,
          repel = TRUE,
   ggtheme = theme_minimal())
```



This plot helps understand how the original features are transformed into the 2 principal components used in the k-means clustering analysis. The longer vectors indicate features that contribute more heavily to the resulting components. This confirms with our optimal clusters that the features most associated with the first principal component are those that have to do with "scoring prowess", such as FG, PPG, FT, twosMade etc. Those most strongly associated with principal component 2 are statistics dealing with threes (near the top of the plot, think of players playing further from the rim and dribble more), and statistics dealing with rebounds/blocks/height/weight (near the bottom of the plot, think of players playing closer to the rim, dribbling less).

Supervised Learning to Predict Positions

Here I'll try to use data from the past 4 regular seasons to predict the positions of players who played at least 20 games. First I'll try to predict the traditional 5-class positions, then the binary classification of guard/wing or big. As Brad Stevens intelligently said, the game is now composed of 3 "real" positions: ball-handler, wing, and big. However, if I were to incorporate this into my model, I would have to manually label every player since not every SG is a ball handler/SF is a wing etc.. I'll live with the 5-class and 2-class labels for now (for 2-class, I categorize PGs, SGs, and SFs into the first class).

Logistic Regression for 5 Position Classification

Logistic regression is a classic way of performing binary classification. Every multiclass problem can be simplified into several binary classification problems, and taking the average or majority vote of them to arrive at a final classification. The two common methods are "one vs. one" and "one vs. rest".

One vs. rest takes a multiclass problem with k classes and splits it into k classifiers. Each of the k classifiers contains a positive (representing the specified class) and a negative (representing the rest of the classes).

After running an observation through each of the k classifiers, it is placed into the class corresponding to the highest score (which could be a distance score if it is a linear classifier). One possible downside of this approach is class imbalance in each binary classifier, even if the original classes are balanced.

One vs. one creates k(k-1)/2 classifiers, one for each pair of classes. The algorithm runs an observation through each of the classifiers and places it in the class that receives the most votes. This method is more computationally expensive because it goes through more classifiers.

The default for nnet and other packages that perform multinomial classification is usually one vs. rest.

We use nnet which is a great tool to fit multinomial logistic models via neural networks.

```r
nba[is.na(nba)] <- 0
#only consider per36 stats, height, weight, and percentages to reduce multicollinearity
#since logistic regression is a linear model
nba2 <- nba[, c(2,10,13,16,17, 20,30:48)]

#Split into train/test data using 2/3 vs. 1/3 split
set.seed(123)
n <- dim(nba)[1]
index <- 1:n
i.train <- sample(index, n/1.5, replace = FALSE)
nba.train <- nba2[i.train,]
nba.test <- nba2[-i.train,]


logistic1 <- multinom(Pos~., data = nba.train, maxit = 500, trace = T)


#Predictions on testing data
logisticPreds1 <- predict(logistic1, type= "class", newdata = nba.test)
#Measure accuracy of model using caret function
logisticAccuracyMulti <- postResample(nba.test$Pos,logisticPreds1)[[1]]
logisticAccuracyMulti #0.8144144

#Error rate using base R
mean(nba.test$Pos != logisticPreds1)

#Confusion matrix, diagonals are correct predictions
test.classes <- logisticPreds1
truth <- factor(nba.test$Pos)
```

```r
table(test.classes, truth)
```

```
##             truth
## test.classes  PG  SG  SF  PF   C
##           PG  99  11   0   0   0
##           SG   8  84  17   0   0
##           SF   0   9  78  14   0
##           PF   1   0  13  87  15
##           C    0   0   1  14 104
```

```r
#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))
```

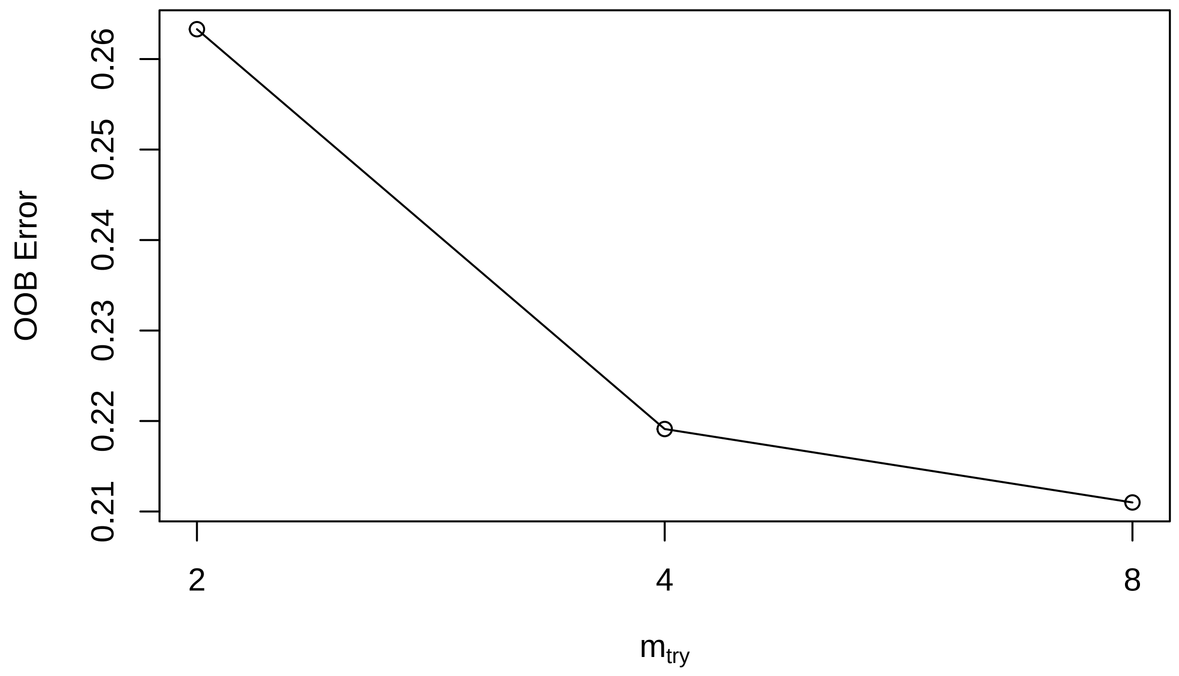
```r
for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba2[index,]
  # everything else to train
  training <- nba2[-index,]

  logisticCV <- multinom(Pos~., data=training, maxit=1000, trace=T)

  pred <- predict(logisticCV, newdata=testing, type="class")

  #  classification error
  accuracy <- postResample(testing$Pos, pred)[[1]]
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}

logisticCVaccuracy <- mean(accuracyList)
logisticCVaccuracy #0.7993421
```

The logistic model for multiclass prediction yielded a cross-validated accuracy score of 0.7993421.

Random Forests for 5 Position Classification

Random forest is a popular ensemble method which uses bagging (bootstrap aggregating) to prevent overfitting in CART (classification and regression trees), but at each split only considers a certain subset of predictors to reduce multicollinearity. In other words, random forests work by reducing the variance of fully grown decision trees. Random Forests/trees are also very easy to interpret compared to other "black box" methods like SVM's.

```r
set.seed(123)

tuneRF(nba.train[,-1], nba.train$Pos)
```



```r
#mtry = 8 gives the lowest out of bag error rate

forest1 <- randomForest(Pos ~ .,
              data = nba.train, mtry = 8, importance = TRUE)
pred1 = as.vector(predict(forest1, newdata = nba.test, type = "class"))
#Confusion matrix, diagonals are correct predictions
pred1 <- factor(pred1, levels=c("PG", "SG", "SF", "PF", "C") )

randomForestAccuracyMulti <- 1 - mean(nba.test$Pos != pred1)
randomForestAccuracyMulti #0.8288288

test.classes <- pred1
truth <- factor(nba.test$Pos)
```

```r
table(test.classes, truth)
```

```
##              truth
## test.classes  PG  SG  SF  PF   C
##           PG  98   4   0   0   0
##           SG   9  91  18   0   0
##           SF   0   9  82  14   0
##           PF   1   0   9  89  19
##           C    0   0   0  12 100
```

```r
#try mtry = 5. sqrt(p) is usually a benchmark, where p is the # of predictors (25)
#forest2 <- randomForest(Pos ~ ., data = nba.train, mtry = 5, importance = TRUE)
#pred2 = as.vector(predict(forest2, newdata = nba.test, type = "class"))

#Confusion matrix, diagonals are correct predictions
#pred2 <- factor(pred2, levels=c("PG", "SG", "SF", "PF", "C") )
#test.classes <- pred2
#truth <- factor(nba.test$Pos)
#table(test.classes, truth)
```

```
#randomForestAccuracy2 <- 1 - mean(nba.test$Pos != pred2)
#randomForestAccuracy2 #0.8234234


#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))

for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba2[index,]
  # everything else to train
  training <- nba2[-index,]

  randomForestCV <- randomForest(Pos~., data=training, mtry = 8, importance = TRUE)

  pred <- predict(randomForestCV, newdata=testing, type="class")

  #  classification error
  accuracy <- postResample(testing$Pos, pred)[[1]]
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}

randomForestCVaccuracy <- mean(accuracyList)
randomForestCVaccuracy #0.8171053
```

The random forest model for multiclass prediction yielded a cross-validated accuracy score of 0.8171053, higher than the logistic model's cross-validated score.

Support Vector Machines for 5 Position Classification

The e1071 package uses the one vs. one method for multiclass classification with support vector machines.

```
set.seed(123)
svmModel <- svm(Pos~., data =  nba.train)
svmPred <- predict(svmModel, newdata=nba.test)
#Confusion matrix, diagonals are correct predictions
svmPred <- factor(svmPred, levels=c("PG", "SG", "SF", "PF", "C") )

svmMultiAccuracy <- 1 - mean(nba.test$Pos != svmPred)
svmMultiAccuracy #0.8126126

test.classes <- svmPred
truth <- factor(nba.test$Pos)
```

```
table(test.classes, truth)
```

```
##             truth
## test.classes  PG  SG  SF  PF   C
##           PG  96   6   0   0   0
##           SG  11  90  20   0   0
##           SF   0   8  74  16   0
##           PF   1   0  14  81   9
##           C    0   0   1  18 110
```

```
#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))

for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba2[index,]
  # everything else to train
  training <- nba2[-index,]

  svmCV <- svm(Pos~., data=training)

  pred <- predict(svmCV, newdata=testing, type="class")

  #  classification error
  accuracy <- postResample(testing$Pos, pred)[[1]]
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}

svmCVaccuracy <- mean(accuracyList)
svmCVaccuracy #0.7875
```

The SVM model for multiclass prediction yielded a cross-validated accuracy score of 0.7875, which is the lowest among the three multiclass models.

Logistic Regression for 2 Position Classification

```
#Exclude Pos, include Pos2
nba3 <- nba[, c(10,13,16,17, 20,30:48, 51)]

#Split into train/test data using 2/3 vs. 1/3 split
set.seed(123)
n <- dim(nba)[1]
index <- 1:n
i.train <- sample(index, n/1.5, replace = FALSE)
nba.train <- nba3[i.train,]
nba.test <- nba3[-i.train,]


glm.fit <- glm(factor(Pos2)~., data = nba.train, family = binomial)
summary(glm.fit)

trainPreds <- predict(glm.fit, type = "response")
trainPreds <- ifelse(trainPreds > 0.5, "Guard/Wing", "Big")
trainError <- 1 - mean(nba.train$Pos2 != trainPreds)
trainError

#Predictions on testing data
preds <- predict(glm.fit, type =  "response", newdata = nba.test)
```

```
glm.pred <- ifelse(preds > 0.5, "Guard/Wing", "Big")


#Accuracy on test data
binarylogisticAccuracy <- 1 - mean(nba.test$Pos2 != glm.pred)
binarylogisticAccuracy #0.9459459
#Error rate using base R
mean(nba.test$Pos2 != glm.pred)

#Confusion matrix, diagonals are correct predictions
test.classes <- glm.pred
truth <- factor(nba.test$Pos2)
```

```
table(test.classes, truth)
```

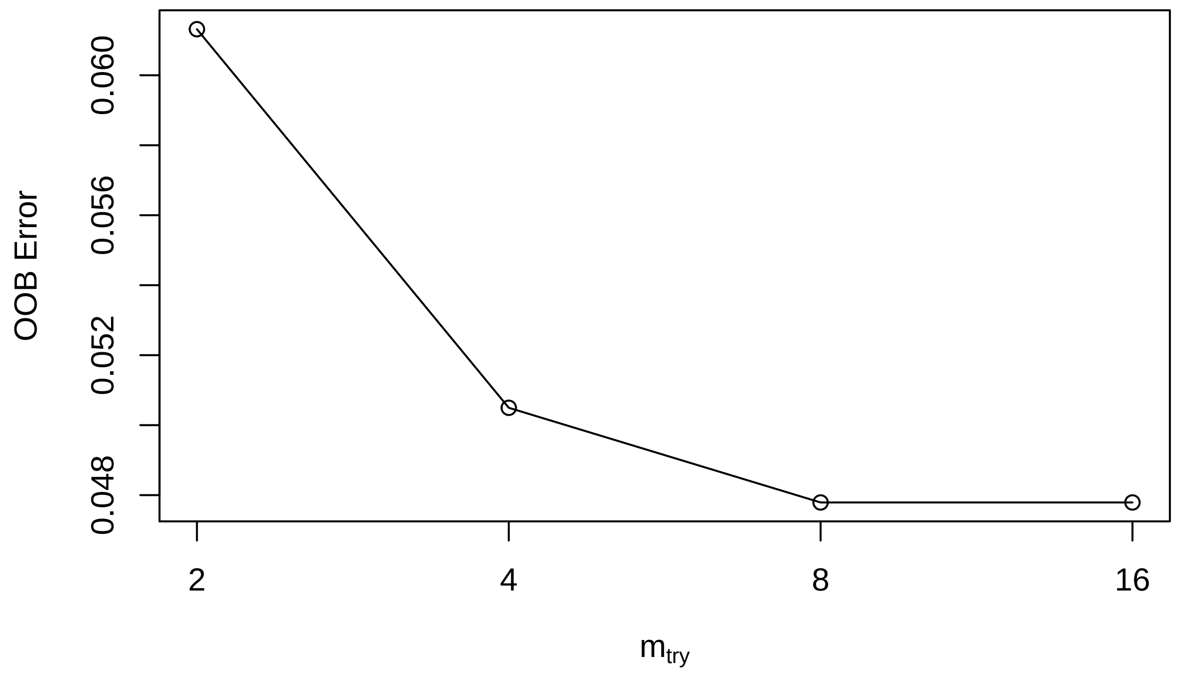
```
##             truth
## test.classes Big Guard/Wing
##   Big        220         16
##   Guard/Wing  14        305
```

```
#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))

for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba3[index,]
  # everything else to train
  training <- nba3[-index,]

  logisticCV2 <- glm(factor(Pos2)~., data=training, family = binomial)

  cvpred <- predict(logisticCV2, newdata=testing, type="response")
  predictions <- ifelse(cvpred > 0.5, "Guard/Wing", "Big")


  #  classification error
  accuracy <- 1 - mean(testing$Pos2 != predictions)
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}

binarylogisticCVaccuracy <- mean(accuracyList)
binarylogisticCVaccuracy #0.9467105
```

The logistic model for binary prediction yielded a cross-validated accuracy score of 0.9467105.

Random Forests for 2 Position Classification

```
set.seed(123)

tuneRF(nba.train[,-25], nba.train$Pos2)
```



```
#mtry = 8 has lowest out of bag error rate estimate

forest1 <- randomForest(Pos2 ~ .,
              data = nba.train, mtry = 8, importance = TRUE)
pred1 = as.vector(predict(forest1, newdata = nba.test, type = "class"))

randomForestBinaryAccuracy <- 1 - mean(nba.test$Pos2 != pred1)
randomForestBinaryAccuracy #0.9531532

#Confusion matrix, diagonals are correct predictions
test.classes <- pred1
truth <- factor(nba.test$Pos2)
```

```
table(test.classes, truth)
```

```
##              truth
## test.classes Big Guard/Wing
##   Big        217          9
##   Guard/Wing  17        312
```

```
#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))

for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba3[index,]
  # everything else to train
  training <- nba3[-index,]

  randomForestCV2 <- randomForest(Pos2~., data=training, mtry = 4, importance = TRUE)
  pred2 <- predict(randomForestCV2, newdata=testing, type="class")

  #  classification error
  accuracy <- 1 - mean(testing$Pos2 != pred2)
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}

randomForestBinaryCVaccuracy <- mean(accuracyList)
randomForestBinaryCVaccuracy #0.9585526
```

The random forest model for binary prediction yielded a cross-validated accuracy score of 0.9585526, higher than the logistic model for binary prediction (0.9467105).

Support Vector Machines for 2 Position Classification

```
set.seed(123)
svmModel1 <- svm(Pos2~., data =  nba.train)
svmPred <- predict(svmModel1, newdata=nba.test)

svmBinaryAccuracy <- 1 - mean(nba.test$Pos2 != svmPred)
svmBinaryAccuracy #0.9531532

#Confusion matrix, diagonals are correct predictions
test.classes <- svmPred
truth <- factor(nba.test$Pos2)
```

```
table(test.classes, truth)
```

```
##             truth
## test.classes Big Guard/Wing
##   Big        218         10
##   Guard/Wing  16        311
```

```
#Cross validation to obtain a more comprehensive accuracy score of how the model will perform on indepe
accuracyList <- c()
cv <- 10
cvDivider <- floor(nrow(nba) / (cv+1))

for(cv in seq(1:cv)) {
  # assign chunk to data test
  index <- c((cv * cvDivider):(cv * cvDivider + cvDivider))
  testing <- nba3[index,]
  # everything else to train
  training <- nba3[-index,]

  svmCV2 <- svm(Pos2~., data=training)

  pred2 <- predict(svmCV2, newdata=testing, type="class")

  #  classification error
  accuracy <- 1 - mean(testing$Pos2 != pred2)
  print(paste('Current Accuracy:',accuracy,'for CV:',cv))
  accuracyList <- c(accuracyList, accuracy)
}
```

```
svmBinaryCVaccuracy <- mean(accuracyList)
svmBinaryCVaccuracy #0.95
```

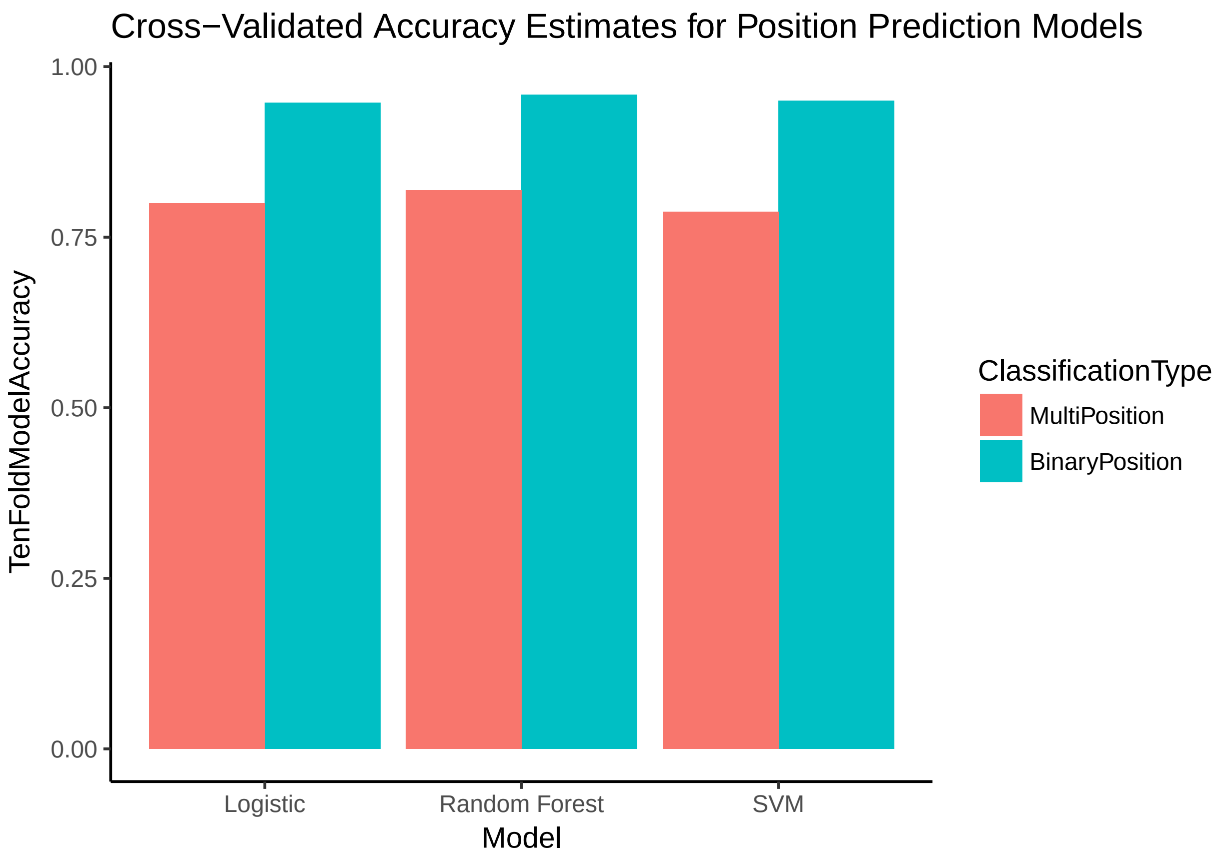
The SVM model for binary prediction yielded a cross-validated accuracy score of 0.95, which finishes between the random forest and logistic models.

Model Cross-Validated Accuracy Comparisons

```
multinomialAccuracies <- data.frame("Logistic" = c(logisticCVaccuracy, binarylogisticCVaccuracy), "Rand


multinomialAccuracies <- melt(data.frame(MultiPosition=c(logisticCVaccuracy, randomForestCVaccuracy, svm
          Model=c("Logistic", "Random Forest", "SVM")),
          variable_name="Classification Type")
```

```
## Using Model as id variables
```

```
names(multinomialAccuracies)[3] <- "TenFoldModelAccuracy"
names(multinomialAccuracies)[2] <- "ClassificationType"

ggplot(multinomialAccuracies, aes(x = Model, y = TenFoldModelAccuracy, fill = ClassificationType))+
       geom_bar(stat = "identity", position = "dodge") + theme_classic() +
  ggtitle("Cross-Validated Accuracy Estimates for Position Prediction Models")
```



Conclusion

For the k-means cluster analysis, it would be interesting to see if I could produce more clusters (without overlapping) to represent more specific player archetypes such as 3 and D, back to the basket big man, and defensive minded guard. This would likely require more data as well as more attributes to distinguish the principal components. For both multiclass classification and binary classification, the random forest models yielded the highest accuracy score (number of correct classifications/total inputs) at 0.8171053 and 0.9585526 for multiclass and binary, respectively. For future renditions of this project, I'll look into collecting more historical data beyond these 4 seasons and considering different combinations of variables, which could include eliminating height/weight. It'll be interesting to see if it becomes more and more difficult to predict the 5 positions as the NBA continues to evolve, and if the NBA will ever officially adopt a less segregated position classification.