National Basketball Association Cluster Analysis

Romith Challa, Carl Ausmees, & Troy Jennings

June 9th, 2021

Contents

1 Background

The dataset used in this analysis was scraped from https://www.basketball-reference.com/. The data contains per-player season statistics for the 2020-2021 NBA season and includes both cumulative and percentage-based statistics.

```r
# Read in basketball data
dat <- read.csv('bballref2.csv')
```

For the purposes of this analysis, we will be using the data which corresponds to a 9-category fantasy basketball league. In this league format, league players compete against 9 categories – field goal percentage, free throw percentage, total three pointers, total rebounds, total assists, total steals, total blocks, total turnovers, and total points. League participants mock draft NBA players to optimize the number of categories they win each week when competing against a fellow league participant.

```r
# Gather only the columns that support clustering and simulation calculation
dat.player <- dat[, c(2, 3, 5, 6, 8:11, 12, 19:21, 24:28, 30)]

# Define new column headings
colnames(dat.player) <- c('player','position','team','games.played',
                          'minutes.played','field.goal.made','field.goal.att',
                          'field.goal.perc','three.pointers','free.throw.made',
                          'free.throw.att','free.throw.perc','rebounds',
                          'assists','steals','blocks','turnovers','points')
```

2 Data Cleaning & Exploratory Data Analysis

First, we'll check the quality of our dataset by looking at "NA" values and looking at basic summary statistics.

```r
# summary(dat.player) # Basic summary statistics
# unique(dat.player$position) # Unique position classifications
# sapply(dat.player, function(y) sum(length(which(is.na(y))))) # Sum the 'NA' values
```

We notice that there are several "NA" values. Since the data represents numerical values (e.g., cumulative and percentage statistics), we will fill "NA" values with 0.

```r
# Resolve "NA" values by replacing with 0
dat.player[is.na(dat.player)] = 0
```

We also notice repeating player names in the 'player' column. This is due to the fact that the dataset includes statistics for players for the team for which they play. Since players may be traded over the course of an NBA season, the dataset includes a row for each team they played for as well as a row for the players total season statistics, denoted with a 'TOT' value in the team column. For this analysis, we will take a naive approach to use player total statistics and we will remove the extraneous rows for each traded player.

```r
# Create boolean mask for filtering out each duplicated player row
trade_mask <- duplicated(dat.player$player)
dat.player <- dat.player[!trade_mask, ]
length(unique(dat.player$player)) # Verify all unique players
```

```
## [1] 540
```

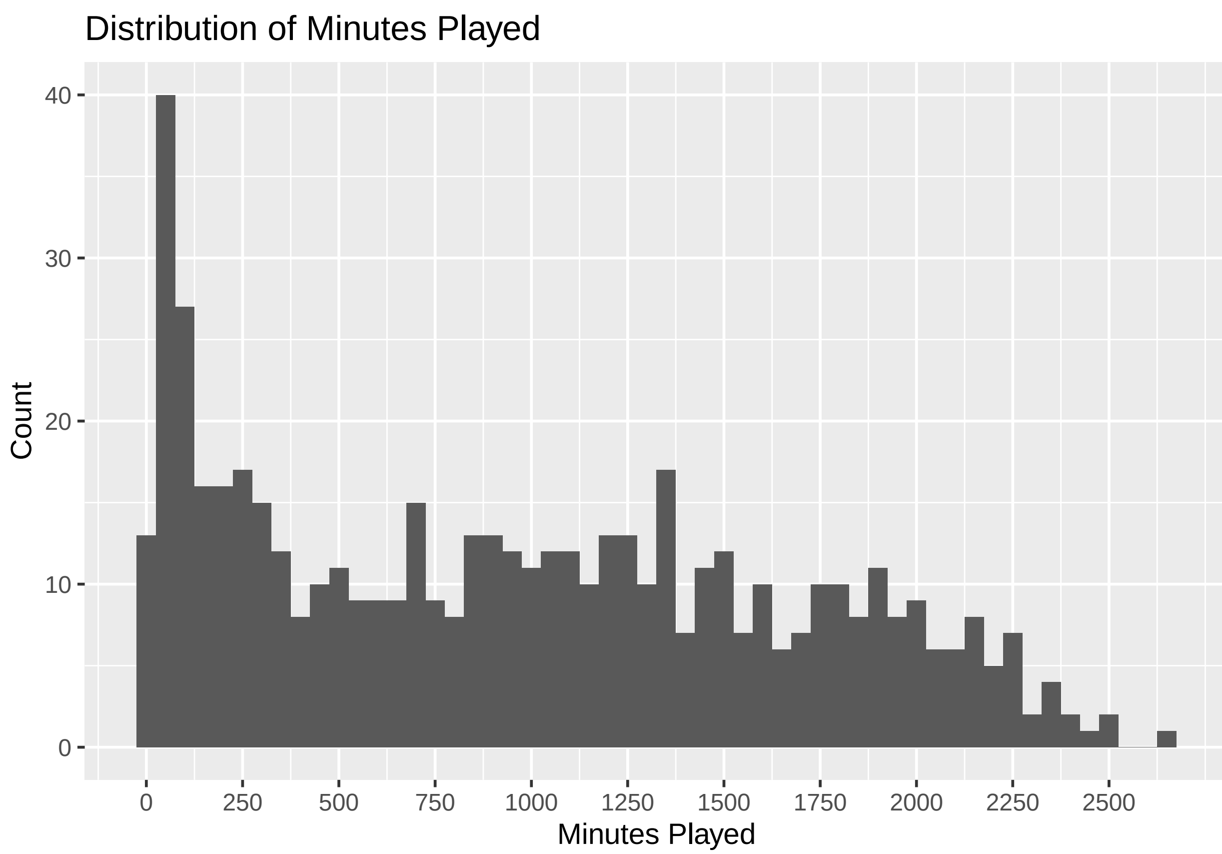
We know that players aren't always assigned a singular position: they can often play multiple positions (i.e., they have utility in more than one position). For purposes of this analysis, we will consolidate player positions into the primary five – point guard (PG), shooting guard (SG), small forward (SF), power forward (PF), and center (C) – by reassigning them to the position the player plays the most.

```
# Consolidate player positions
dat.player$position[dat.player$position == 'C-PF'] <- 'C'
dat.player$position[dat.player$position == 'PG-SG'] <- 'PG'
dat.player$position[dat.player$position == 'SF-SG'] <- 'SF'
dat.player$position[dat.player$position == 'SG-PG'] <- 'SG'
dat.player$position[dat.player$position == 'SG-SF'] <- 'SG'
dat.player$position[dat.player$position == 'SF-PF'] <- 'SF'
dat.player$position[dat.player$position == 'PF-C'] <- 'PF'
dat.player$position[dat.player$position == 'PF-SF'] <- 'PF'
unique(dat.player$position) # Verify all positions consolidated
```

```
## [1] "PF" "PG" "C"  "SG" "SF"
```

From the earlier summary statistics, we notice zero values in all the categories. If we were truly drafting players for a fantasy league, we avoid players with low usage (e.g., low minutes played or low games played), since there will be a direct correlation between minutes played/games played and the category statistics. For purposes of this analysis, we remove players with minutes played less than the mean minutes played for the entire league.

```
# Evaluate the distribution of minutes played across the league
ggplot(data= dat.player, aes(minutes.played)) +
    geom_histogram(binwidth= 50) +
    scale_x_continuous(breaks = seq(0, 2500, 250)) +
    ggtitle('Distribution of Minutes Played') +
    xlab('Minutes Played') + ylab('Count')
```




```r
summary(dat.player)
```

```
##     player            position             team            games.played
##  Length:540         Length:540         Length:540         Min.   : 1.00
##  Class :character   Class :character   Class :character   1st Qu.:26.75
##  Mode  :character   Mode  :character   Mode  :character   Median :46.00
##                                                           Mean   :42.69
##                                                           3rd Qu.:61.00
##                                                           Max.   :72.00
##  minutes.played   field.goal.made field.goal.att   field.goal.perc
##  Min.   :   3.0   Min.   :  0.0   Min.   :   0.0   Min.   :0.0000
##  1st Qu.: 298.5   1st Qu.: 39.0   1st Qu.:  93.0   1st Qu.:0.4027
##  Median : 926.0   Median :128.0   Median : 273.0   Median :0.4415
##  Mean   : 965.7   Mean   :164.9   Mean   : 353.7   Mean   :0.4469
##  3rd Qu.:1505.5   3rd Qu.:253.2   3rd Qu.: 559.2   3rd Qu.:0.4963
##  Max.   :2667.0   Max.   :732.0   Max.   :1396.0   Max.   :1.0000
##  three.pointers   free.throw.made  free.throw.att   free.throw.perc
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :0.0000
##  1st Qu.:  4.00   1st Qu.: 13.00   1st Qu.: 17.00   1st Qu.:0.6705
##  Median : 31.00   Median : 41.00   Median : 55.00   Median :0.7730
##  Mean   : 50.79   Mean   : 67.87   Mean   : 87.29   Mean   :0.7286
##  3rd Qu.: 83.00   3rd Qu.: 94.00   3rd Qu.:119.25   3rd Qu.:0.8430
##  Max.   :337.00   Max.   :484.00   Max.   :581.00   Max.   :1.0000
##     rebounds         assists           steals           blocks
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
```

```
##  1st Qu.: 53.75   1st Qu.: 18.00   1st Qu.:  8.75   1st Qu.:  4.00
##  Median :142.00   Median : 63.00   Median : 26.00   Median : 12.00
##  Mean   :177.20   Mean   : 99.22   Mean   : 30.29   Mean   : 19.49
##  3rd Qu.:256.00   3rd Qu.:129.00   3rd Qu.: 46.00   3rd Qu.: 26.00
##  Max.   :960.00   Max.   :763.00   Max.   :128.00   Max.   :190.00
##    turnovers          points
##  Min.   :  0.0   Min.   :   0.0
##  1st Qu.: 13.0   1st Qu.: 104.5
##  Median : 40.5   Median : 343.5
##  Mean   : 52.9   Mean   : 448.4
##  3rd Qu.: 76.0   3rd Qu.: 687.2
##  Max.   :312.0   Max.   :2015.0
```

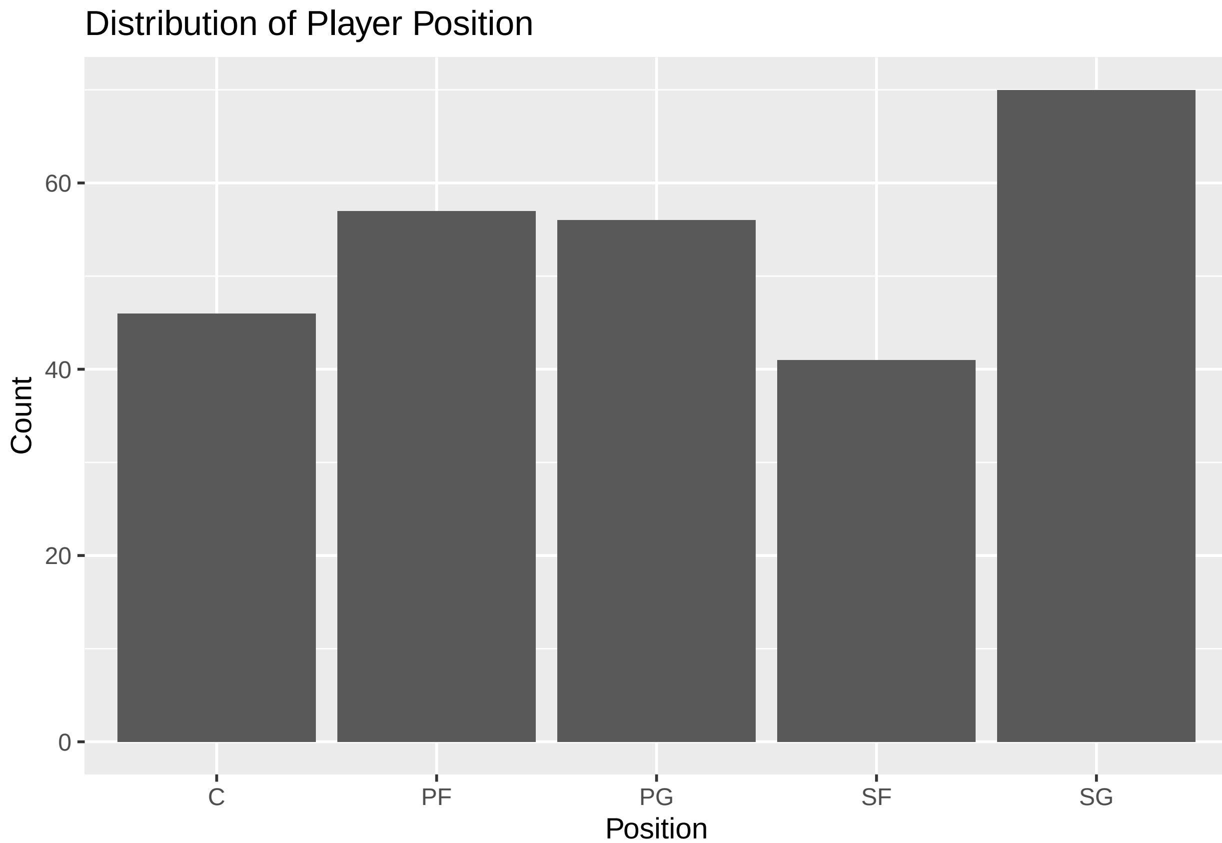
```
# Remove players with minutes played less than the mean minutes played
dat.player <- filter(
    dat.player,
    minutes.played >= median(dat.player$minutes.played))

length(unique(dat.player$player))
```

```
## [1] 270
```

After removing players with low minutes played, we notice there are still sufficient remaining players for 10-player rosters for a 14-team fantasy league.

```
ggplot(data= dat.player,
       aes(x= position)) +
    geom_histogram(stat= 'count') +
    ggtitle('Distribution of Player Position') +
    xlab('Position') + ylab('Count')
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



Finally, we will split our data so that we can: * Maintain player positions data * Split the category statistics data for scaling and clustering analysis * Maintain free throw and field goal actuals to calculate true free throw and field goal percentages for a team

```
# Set the player names as the row name
row.names(dat.player) <- dat.player[, 1]

# Split data into separate data frames
dat.positions <- dat.player[, c(1:2)]
dat.stats.raw <- dat.player[, c(8:9, 12:18)]
dat.stats <- data.frame(scale(dat.player[, c(8:9, 12:18)])) # Scale stats data
dat.ft.fg <- data.frame(dat.player[, c(1, 6:7, 10:11)])
```

3 K-Means Analysis

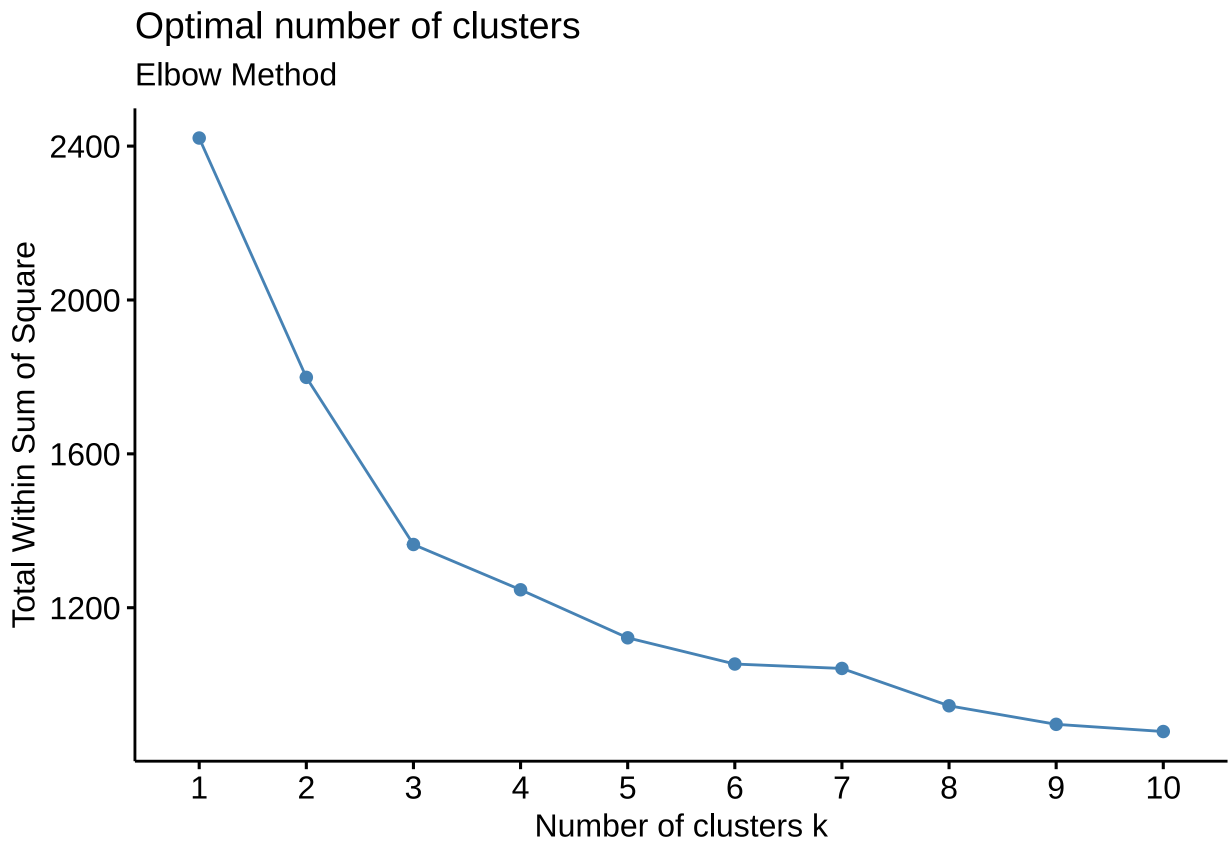
With data cleaning and processing complete, we can turn our attention to analyzing the optimal number of clusters to use in the model.

```
# Define a function to analyze the optimal number of clusters on the dataset
run.cluster.analyses <- function(data){
    set.seed(1651654)

    # Elbow Method
    wss <- fviz_nbclust(
```

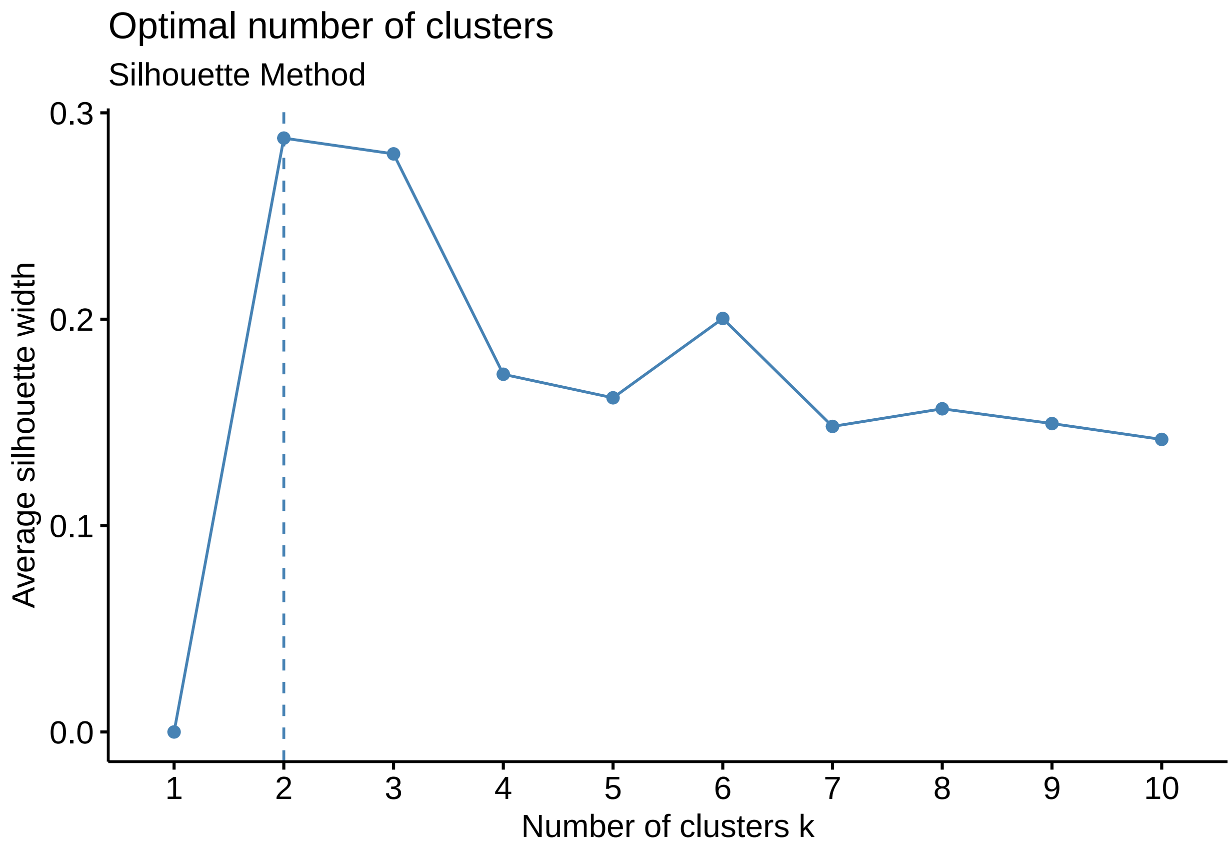
```
            data,
            FUN= kmeans,
            method= "wss") +
        labs(subtitle= "Elbow Method")
    plot(wss)

    # Silhouette Method
    avg.sil <- fviz_nbclust(
            data,
            FUN= kmeans,
            method= "silhouette")+
        labs(subtitle = "Silhouette Method")
    plot(avg.sil)

    # Gap Statistic
    gap.stat <- fviz_nbclust(
            data,
            FUN= kmeans,
            nstart= 25,
            method = "gap_stat",
            nboot = 500,
            verbose= FALSE) +
        labs(subtitle = "Gap Statistic Method")
    plot(gap.stat)
}
```

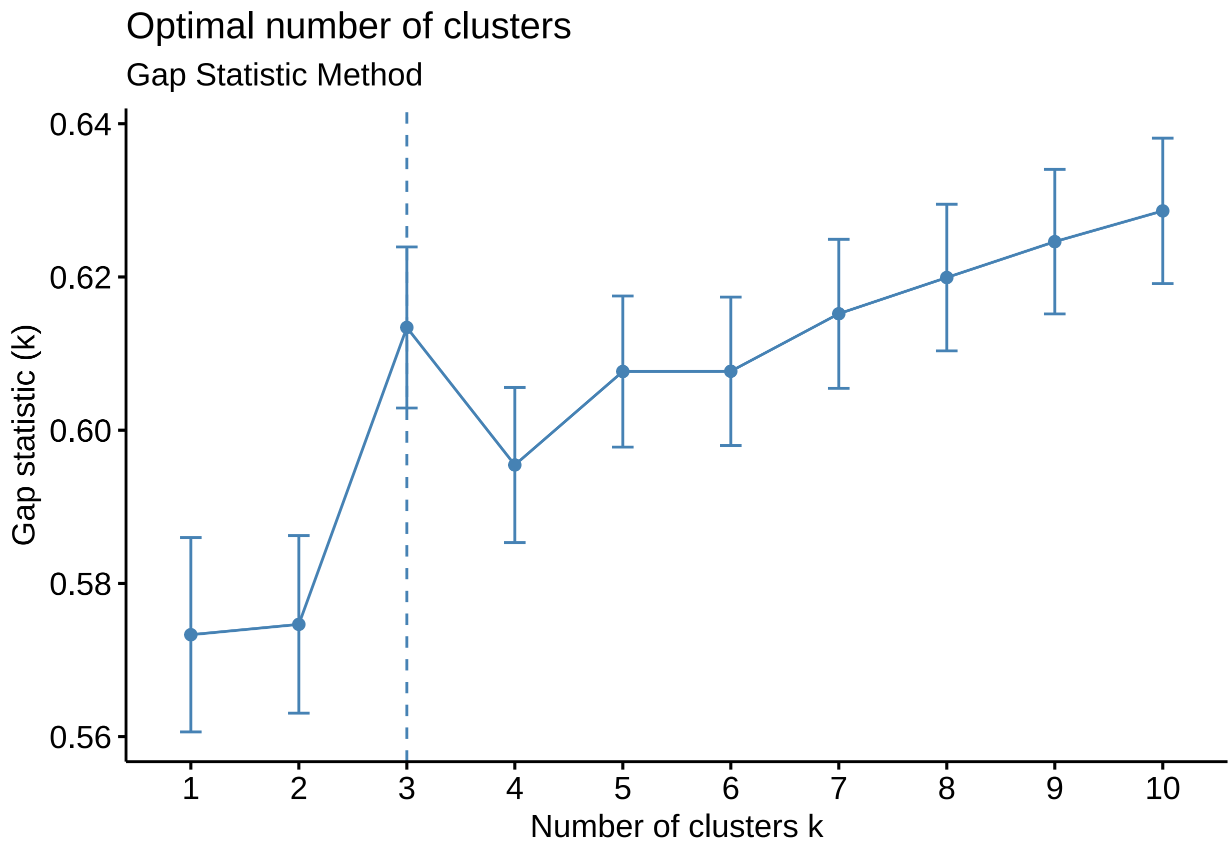
We run an analysis of the clustering model to determine the optimal number of clusters. This analysis triangulates the optimal number of clusters using three methods – elbow method, silhouette method, and gap statistic.

```
# Run cluster analysis across scaled statistics
run.cluster.analyses(dat.stats)
```



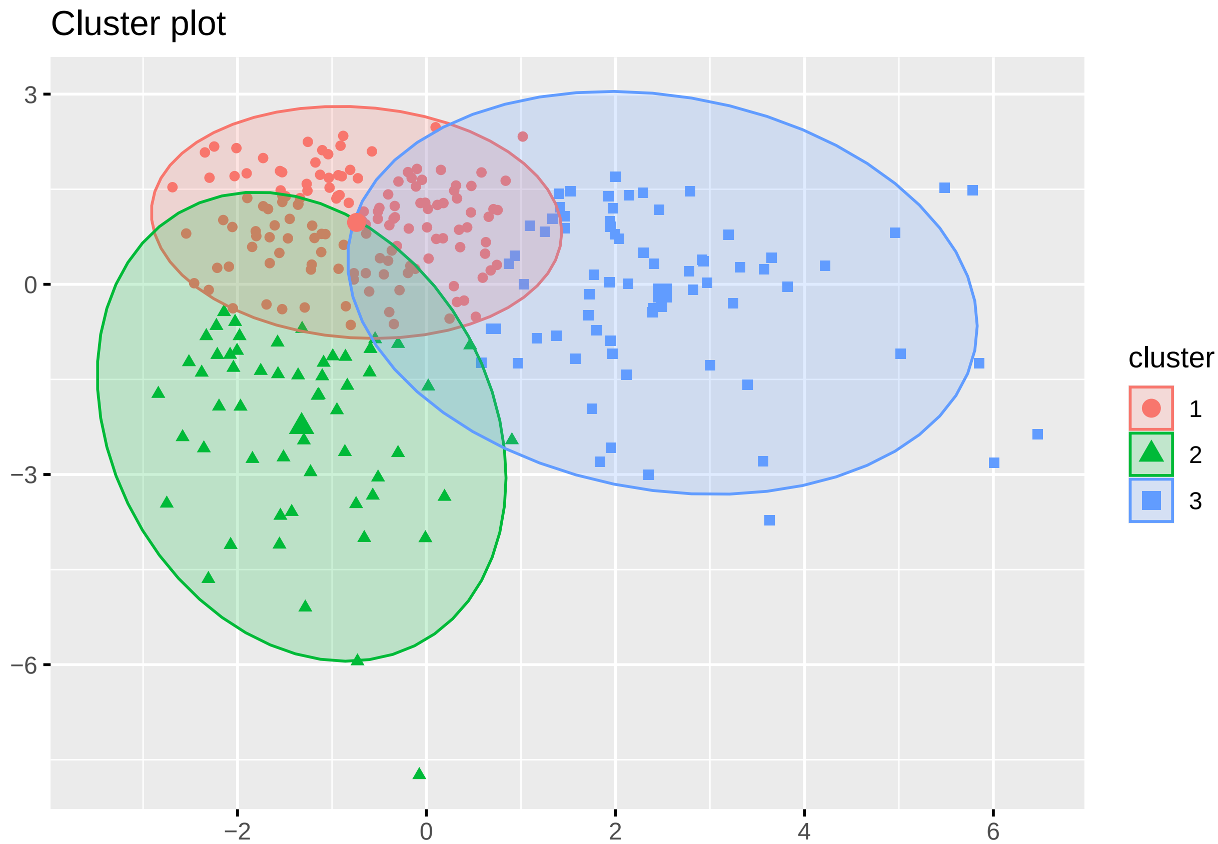
Optimal number of clusters
Elbow Method
Total Within Sum of Square
2400
2000
1600
1200
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Optimal number of clusters
Silhouette Method
Average silhouette width
0.0
0.1
0.2
0.3
1
2
3
4
5
6
7
8
9
10
Number of clusters k



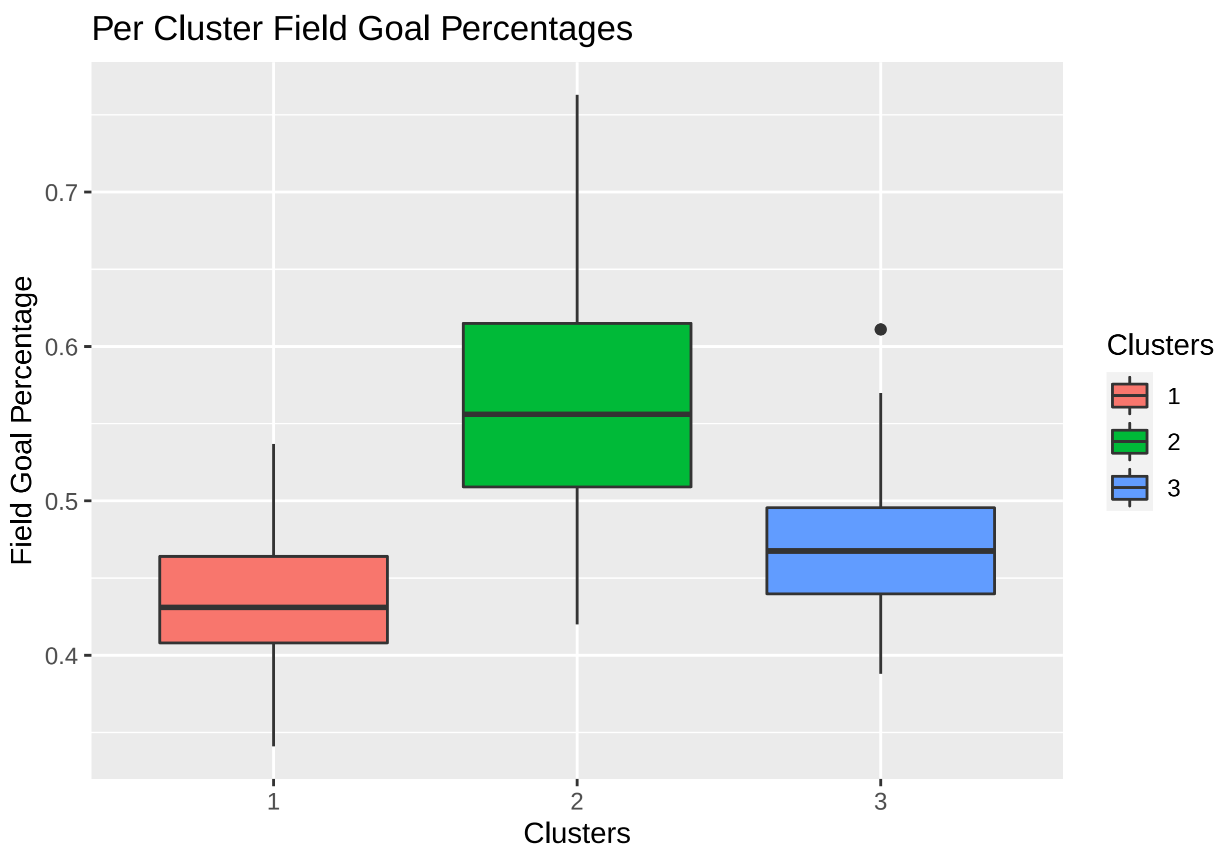
From this analysis, we will use 3 clusters in our model.

```
set.seed(123456)
km_res <- kmeans(dat.stats, centers= 3, nstart= 20)
fviz_cluster(km_res, dat.stats, ellipse.type = 'norm',
             xlab= FALSE, ylab= FALSE, geom= 'point') +
    ggsave('all_position_3_cluster_scaled_all.png', width= 5, height= 3)
```

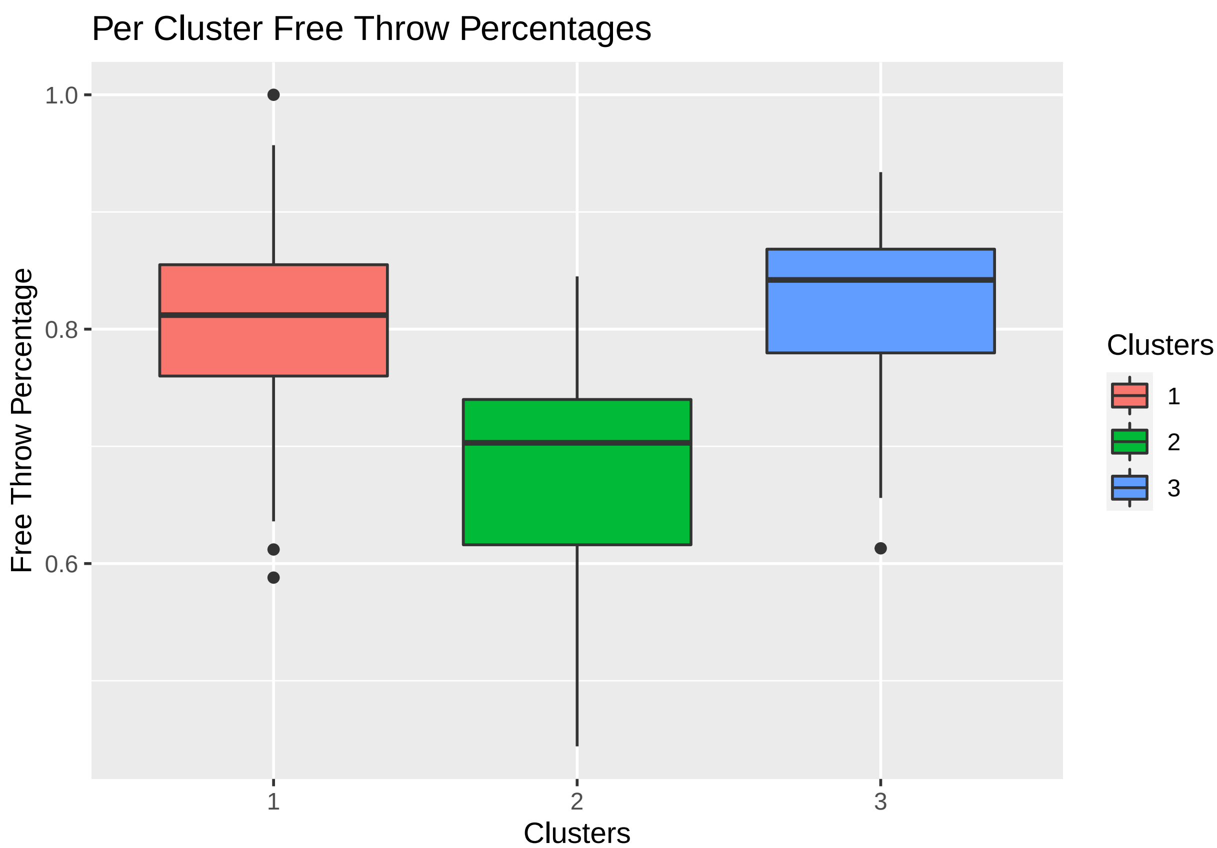


4 Differences in 3-Cluster Model

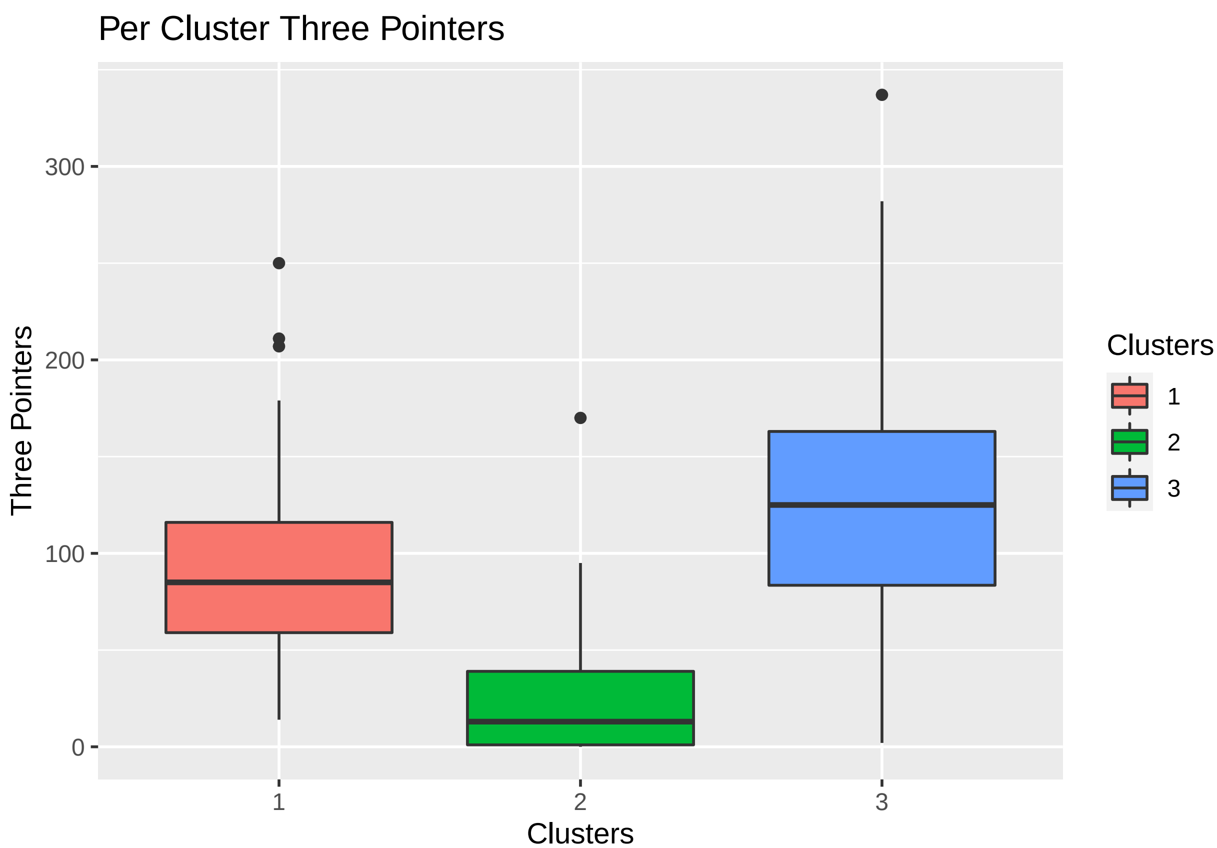
```
# Combine raw stats and the clusters for cross-comparisons
three.cluster <- cbind(dat.stats.raw, 'cluster'= km_res$cluster)

# Plot the differences between clusters over each category
ggplot(data= three.cluster[, c('field.goal.perc', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= field.goal.perc,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Field Goal Percentages') +
    xlab('Clusters') + ylab('Field Goal Percentage') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



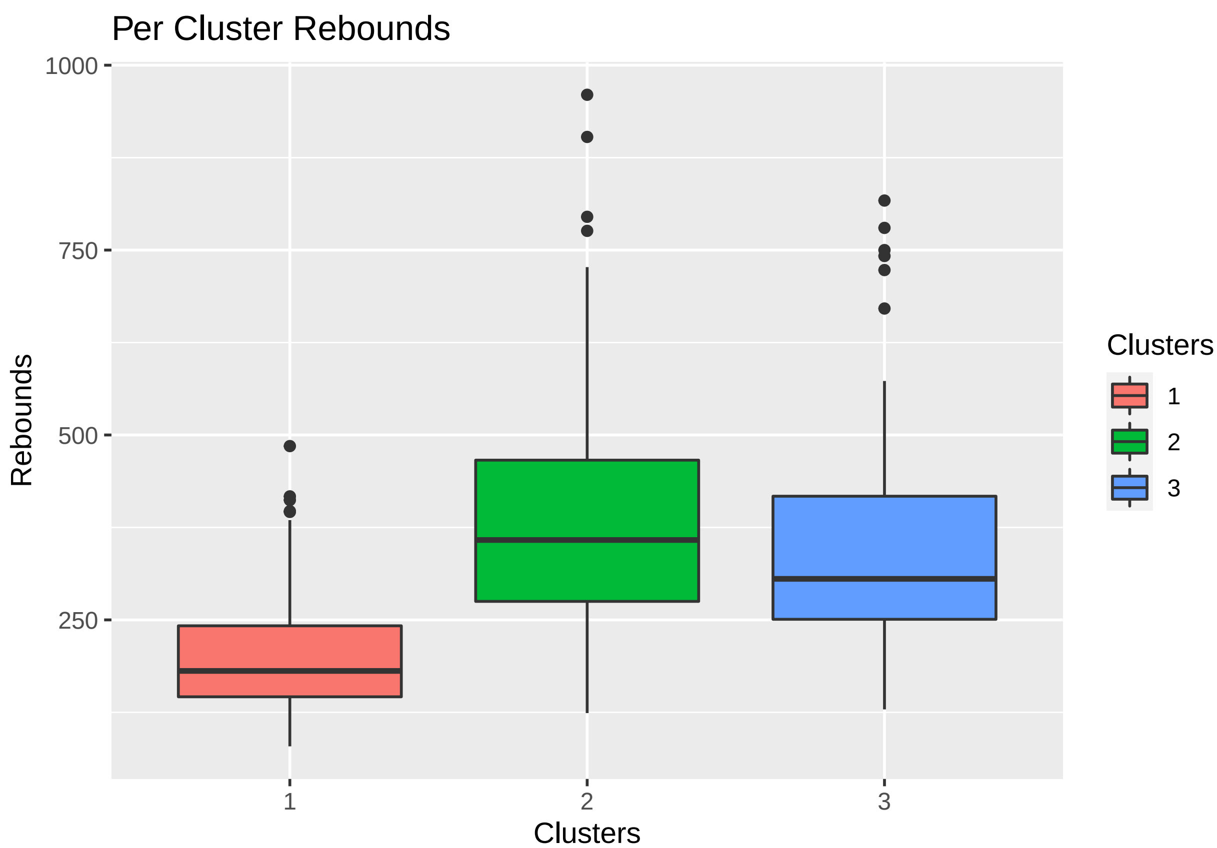

```r
ggplot(data= three.cluster[, c('free.throw.perc', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= free.throw.perc,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Free Throw Percentages') +
    xlab('Clusters') + ylab('Free Throw Percentage') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



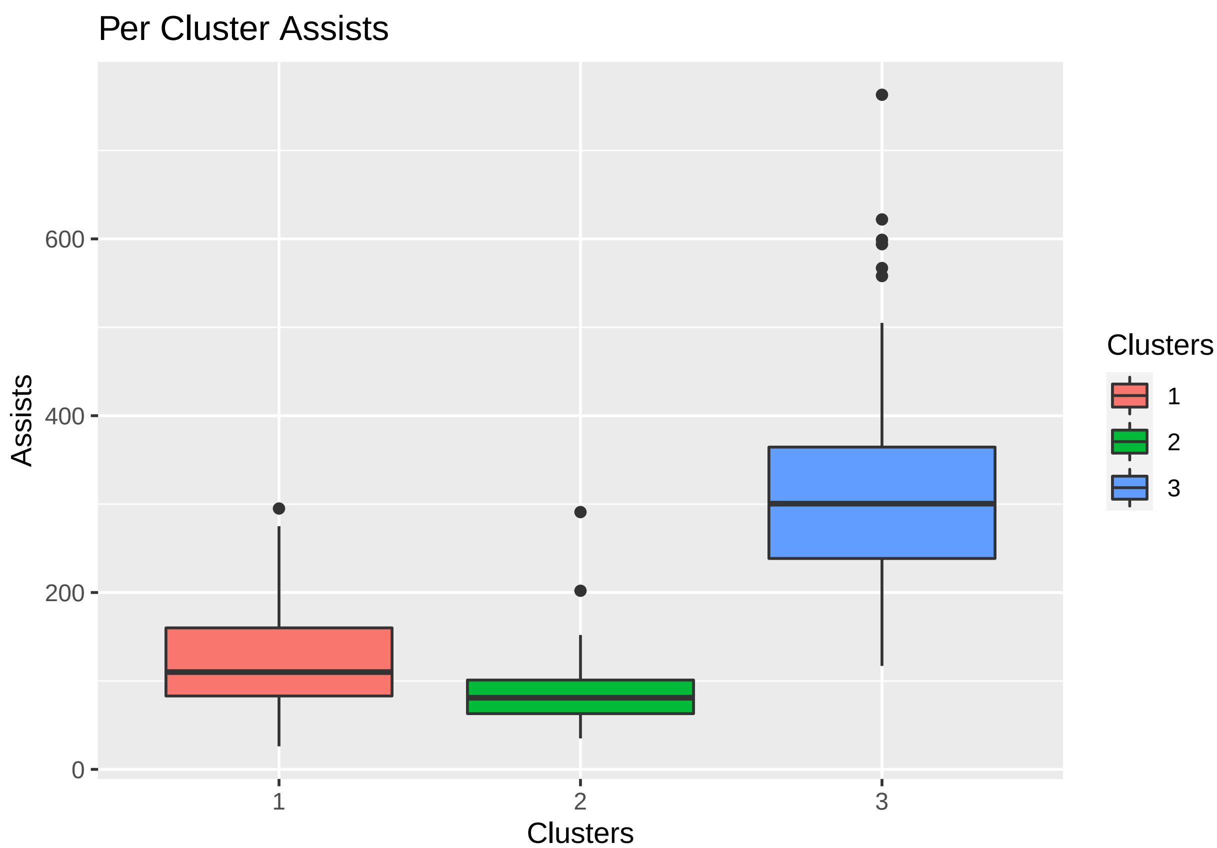

```
ggplot(data= three.cluster[, c('three.pointers', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= three.pointers,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Three Pointers') +
    xlab('Clusters') + ylab('Three Pointers') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



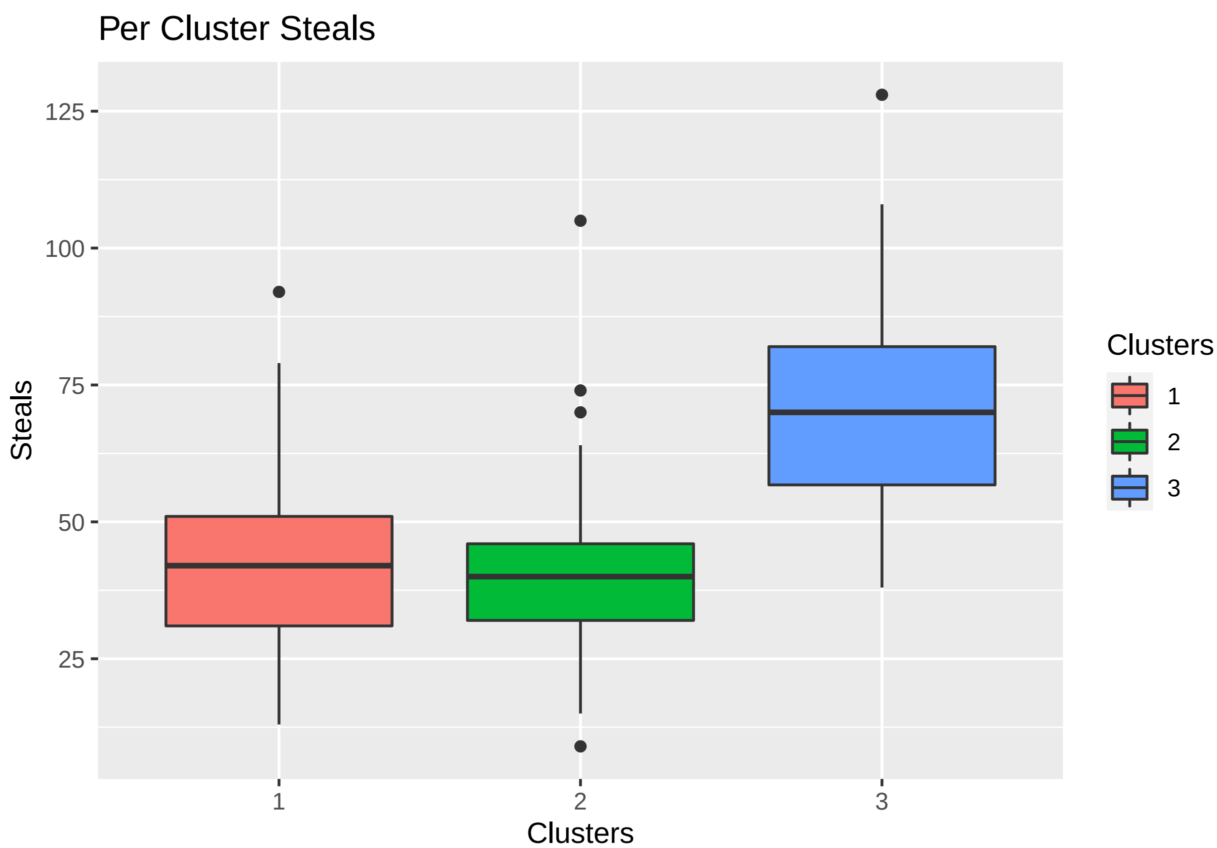

```
ggplot(data= three.cluster[, c('rebounds', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= rebounds,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Rebounds') +
    xlab('Clusters') + ylab('Rebounds') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



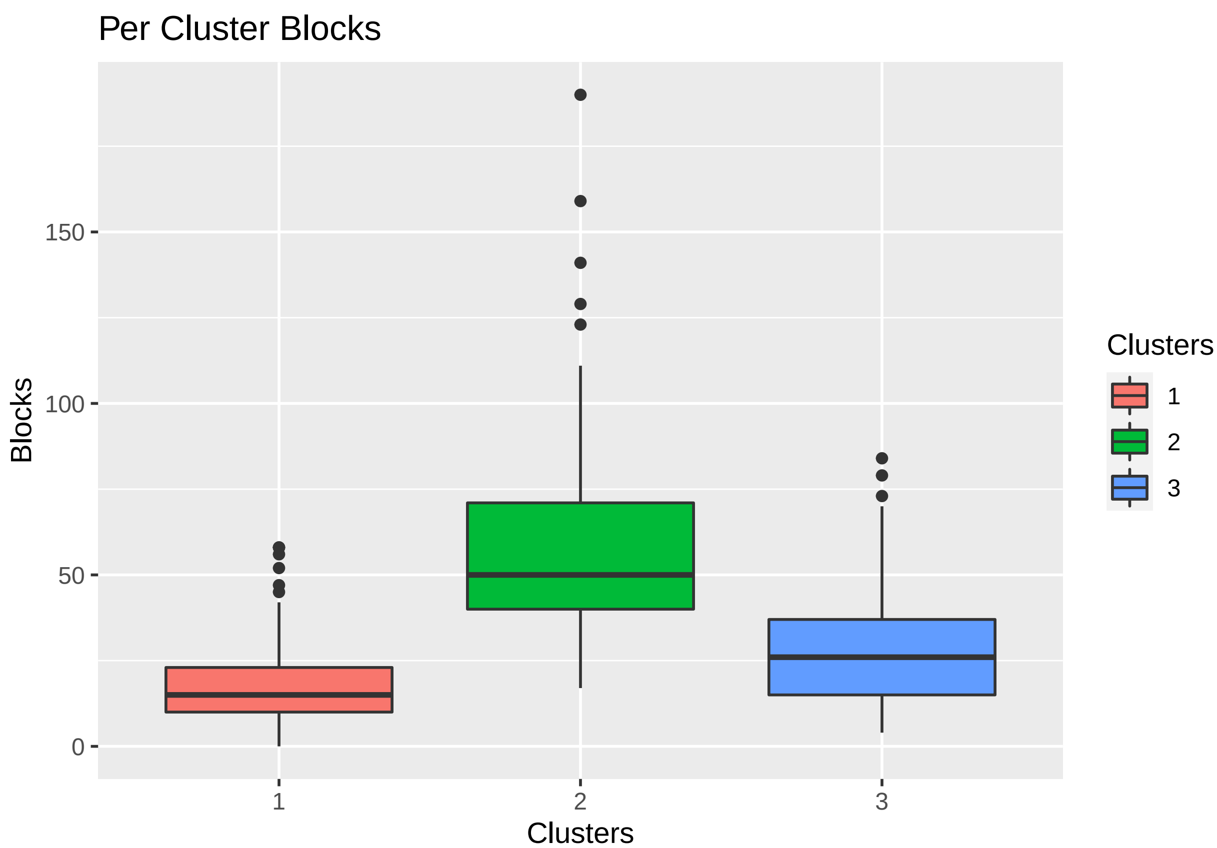

```r
ggplot(data= three.cluster[, c('assists', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= assists,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Assists') +
    xlab('Clusters') + ylab('Assists') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



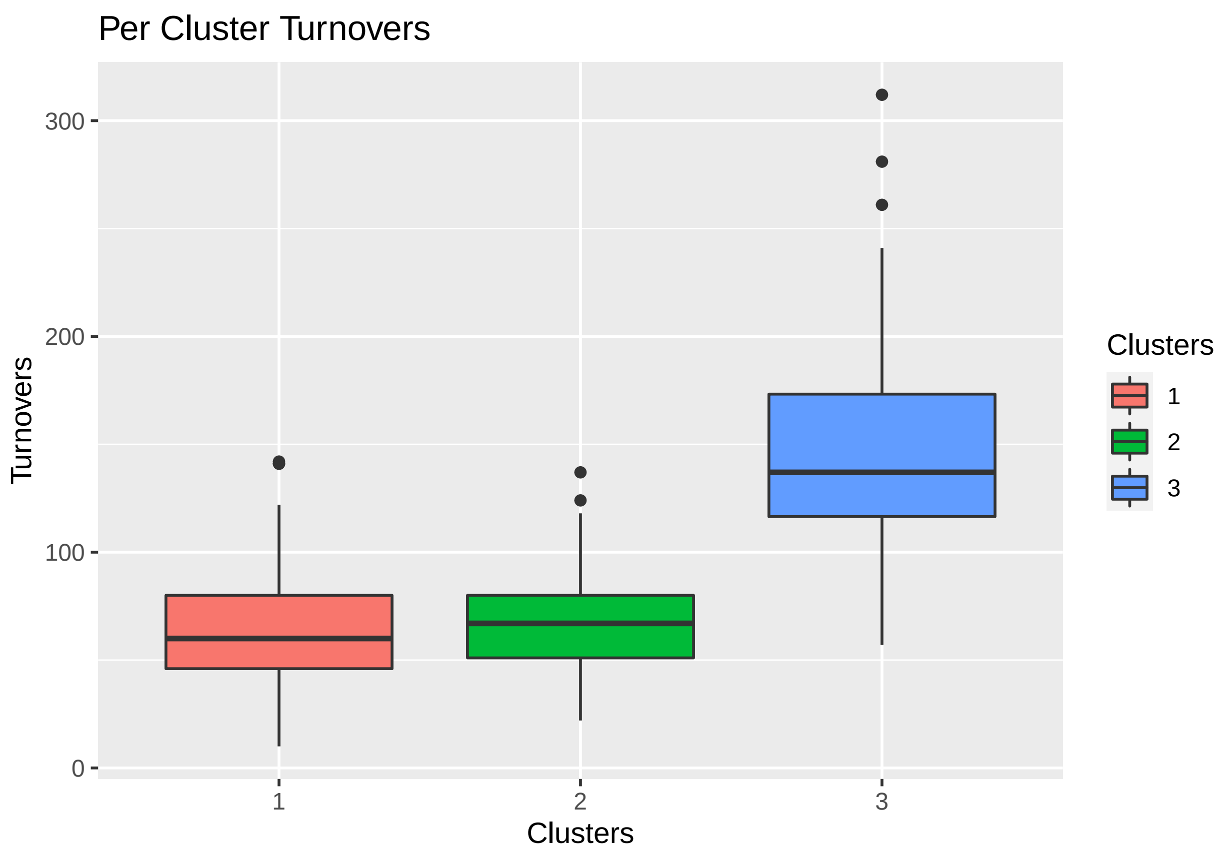

```
ggplot(data= three.cluster[, c('steals', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= steals,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Steals') +
    xlab('Clusters') + ylab('Steals') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



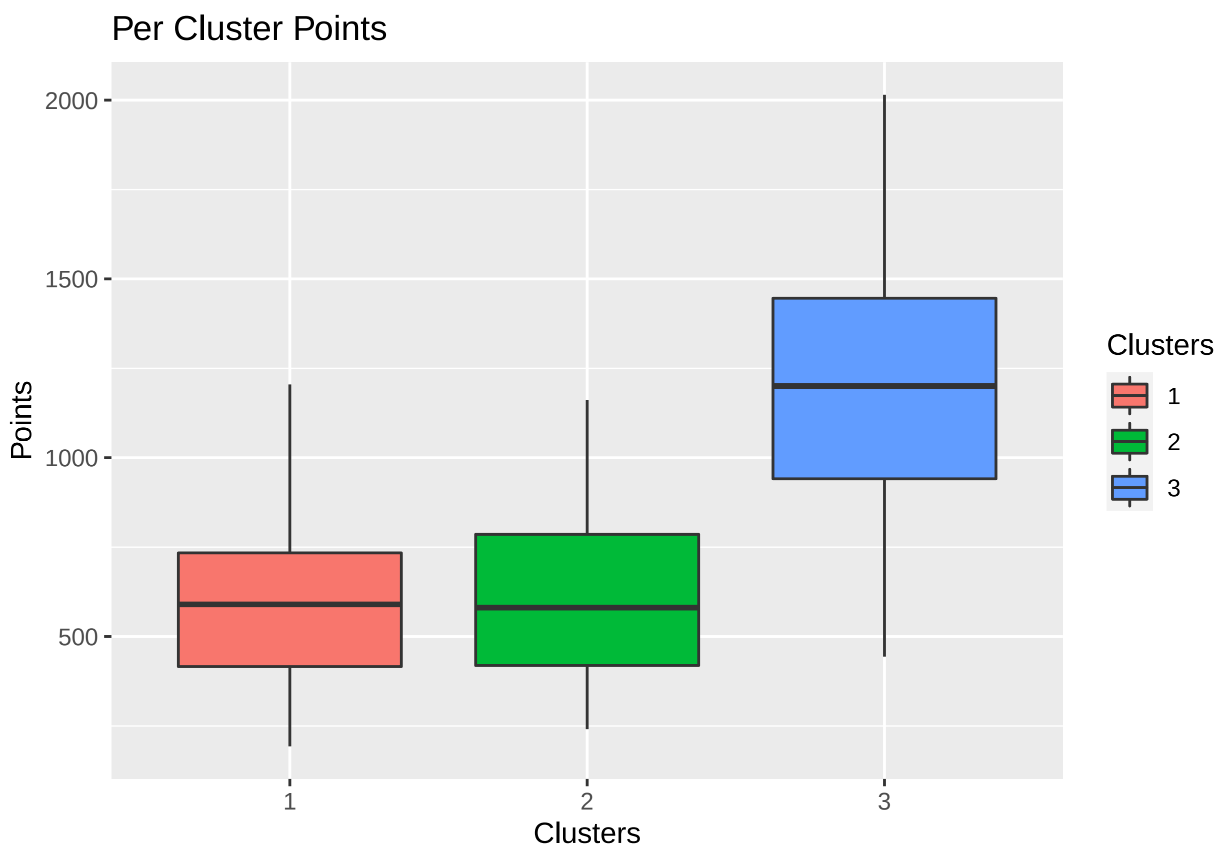

```
ggplot(data= three.cluster[, c('blocks', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= blocks,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Blocks') +
    xlab('Clusters') + ylab('Blocks') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```




```
ggplot(data= three.cluster[, c('turnovers', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= turnovers,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Turnovers') +
    xlab('Clusters') + ylab('Turnovers') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```




```
ggplot(data= three.cluster[, c('points', 'cluster')]) +
    geom_boxplot(
        aes(
            x= factor(cluster),
            y= points,
            fill= factor(cluster))) +
    ggtitle('Per Cluster Points') +
    xlab('Clusters') + ylab('Points') +
    scale_fill_discrete(name= 'Clusters', labels= c(1, 2, 3))
```



5 Research Question

Is there a significant difference in categories won, based on drafting strategies that selects either guards or big-men in the later rounds.

```
# Define variables for the tiers so we don't have to
top.tier = 3
small.heavy = 1
big.heavy = 2
```

```
# All position types for simulation of drafting all primary positions
positionals <- c('PG', 'SG', 'SF', 'PF', 'C')

# Recombine positions, with scaled player stats, and cluster index
master <- cbind(dat.positions, dat.stats, 'cluster'= km_res$cluster)

# Join the attempts and makes for free throws and field goals
master <- full_join(x= master, y= dat.ft.fg, by= 'player')

# Define a function that drafts all positional players from top-tier players
draft_positional <- function() {
    # Empty data frame for appending
    first_five <- data.frame()
    # Draft each position
```

```r
    for (p in positionals) {
        # Create local data frame filtered to position and cluster
        this.position <- filter(master, position== p, cluster== top.tier)
        # Sample from the data frame
        this.sample <- sample_n(tbl= this.position, size= 1, replace= FALSE)
        # Bind sample to create the team
        first_five <- rbind(first_five, this.sample)
    }
    return(first_five)
}
```

Run simulations to simulate n NBA drafts

```r
# Define number of simulations
n <- 10000
# Create vector for storing each simulation result (won or lost (tie))
wins.vector <- c(rep(0, n))
total.wins <- 0

# Run simulations
for (sim in 1:n) {
    # Initialize empty data frames for our individual teams
    bigs <- data.frame()
    smalls <- data.frame()

    # Draft all required position and add to both teams
    ff <- draft_positional()
    smalls <- rbind(smalls, ff)
    bigs <- rbind(bigs, ff)

    # Remove previously drafted players to avoid "over-drafting"
    remaining_pool <- anti_join(master, ff, by= 'player')

    # Draft the 6th man for each team (Forward for smalls and Guard for bigs)
    this.forward <- sample_n(
        tbl= filter(
            remaining_pool,
            cluster== top.tier, position== 'SF' | position== 'PF'),
        size= 1, replace= FALSE)
    this.guard <- sample_n(
        tbl= filter(
            remaining_pool,
            cluster== top.tier, position== 'PG' | position== 'SG'),
        size= 1, replace= FALSE)

    # Add drafted to teams
    smalls <- rbind(smalls, this.forward)
    bigs <- rbind(bigs, this.guard)

    # Remove the 6th men to avoid "over-drafting"
    remaining_pool <- anti_join(remaining_pool, this.forward, by= 'player')
    remaining_pool <- anti_join(remaining_pool, this.guard, by= 'player')
```

```r
# Draft the remaining utility players and another center for the bigs
small.utility <- sample_n(
    tbl= filter(
        remaining_pool,
        cluster== small.heavy, position!= 'PF', position!= 'C'),
    size= 4, replace= FALSE)

big.non.center <- sample_n(
    tbl= filter(
        remaining_pool,
        cluster== big.heavy, position != 'C'),
    size= 1, replace= FALSE)
bigs <- rbind(bigs, big.non.center)
remaining_pool <- anti_join(remaining_pool, big.non.center, by= 'player')

big.utility <- sample_n(
    tbl= filter(
        remaining_pool,
        cluster== big.heavy, position!= 'PG', position!= 'SG'),
    size= 3, replace= FALSE)

# Add drafted to teams
smalls <- rbind(smalls, small.utility)
bigs <- rbind(bigs, big.utility)

if (length(unique(smalls$player)) != 10 |
    length(unique(smalls$player)) != 10) {
    print('Duplicate players!')
}

# Sum the totals for each team
small.totals <- c(
    sum(smalls$field.goal.made) / sum(smalls$field.goal.att),
    sum(smalls$three.pointers),
    sum(smalls$free.throw.made) / sum(smalls$free.throw.att),
    sum(smalls$rebounds),
    sum(smalls$assists),
    sum(smalls$steals),
    sum(smalls$blocks),
    sum(smalls$points),
    sum(smalls$turnovers))

big.totals <- c(
    sum(bigs$field.goal.made) / sum(bigs$field.goal.att),
    sum(bigs$three.pointers),
    sum(bigs$free.throw.made) / sum(bigs$free.throw.att),
    sum(bigs$rebounds),
    sum(bigs$assists),
    sum(bigs$steals),
    sum(bigs$blocks),
    sum(bigs$points),
    sum(bigs$turnovers))
```

```
    # Take the vector differences
    diff <- small.totals - big.totals

    # Calculate won or lost for all
    small.won.cats <- c(0, 0)
    for (k in 1:8) {
        # Add wins for positive differences
        if (diff[k] > 0) {
            small.won.cats[1] <- small.won.cats[1] + 1
        # Add ties
        } else if (diff[k] == 0) {
            small.won.cats[2] == small.won.cats[2] + 1
        }
    }

    # Add wins for positive differences on TURNOVERS
    if (diff[9] < 0) {
        small.won.cats[1] <- small.won.cats[1] + 1
    # Add ties for TURNOVERS
    } else if (diff[9] == 0) {
        small.won.cats[2] == small.won.cats[2] + 1
    }

    # Check for explicit winning strategy
    if (small.won.cats[1] >= 5) {
        total.wins <- total.wins + 1
        wins.vector[sim] <- 1
    }
}
```

```
# Output the win percentage for the small strategy
print(total.wins / n)
```

```
## [1] 0.4418
```

6 Hypothesis Testing

- Z-Score Approximation (with large sample size of 10000):
- Sample Mean: $\bar{X}$ = total.wins
- Sample Std Dev: $sigma = \sqrt{P(1-P) \cdot n} = \sqrt{0.5\,(1-0.5) \cdot 10000} = 50$
- Under null hypothesis, mu = 0.50*10000 = 5000
- z-score: $\frac{\bar{X}-\mu}{\sigma} = \frac{4417.5-5000}{50}$ = -11.6
- -11.6 < -1.96 (95% confidence interval z-score under normal approximation; observed z-score is outside of the range from -1.96 to 1.96)

Alternatively, in the code block below, we can under the assumption of the null hypothesis conduct qbinom calculations to get the range of expected values (based on 95% and 99% confidence intervals) for wins across 10000 simulations. Based on this, we can assess if our observed value is within the range. Based on this computation, we can see that our observed outcome of 4418 wins is not within that range, enabling us to confidently reject the null hypothesis. Through this, we can conclude there is indeed a significant change in win percentage based on the two drafting strategies.

```r
# qbinom outputs the expected range of small-strategy wins under the null
# hypothesis, for 95% confidence interval
l95 <- qbinom(0.025, 10000, 0.5)
u95 <- qbinom(0.975, 10000, 0.5)
cat("For a 95% confidence interval, under assumption of Null H, we expect wins
    to range from", l95, "to", u95, "\n")
```

```
## For a 95% confidence interval, under assumption of Null H, we expect wins
##     to range from 4902 to 5098
```

```r
# qbinom outputs the expected range of small-strategy wins under the null
# hypothesis, for 95% confidence interval
l99 <- qbinom(0.005, 10000, 0.5)
u99 <- qbinom(0.995, 10000, 0.5)
cat("For a 99% confidence interval, under assumption of Null H, we expect wins
    to range from", l99, "to", u99, "\n")
```

```
## For a 99% confidence interval, under assumption of Null H, we expect wins
##     to range from 4871 to 5129
```

```r
# Probability of getting 4418 in a binomial distribution, if under assumption of
# the null, should yield 5000
cat("Under assumption of Null H,the probability of getting 4418 wins out of
    10000 simulations is: ", dbinom(4418, 10000, 0.5))
```

```
## Under assumption of Null H,the probability of getting 4418 wins out of
##     10000 simulations is:  2.611548e-32
```