NBA Analysis

Team 10

Contents

Introduction

Project Description

Description

We are team 10 and we want to analyze the underlying trends of the NBA. Our goal is to try to find predictive statistics for predicting the winner of a game, the nost valuable player of the season and the season championship winner. We chose a data set called HoopR since it has all game plays and statistics all the way back to the early 2000's. We want to understand if there are ways of predicting these three goals and if there are we want to evaluate how accurate the predictions are and if they are viable and robust for future NBA seasons.

Goals

MVPs - Find predictive stats for identifying the season MVP

Winning Teams - Find predictive stats for identifying the champion

Championships - Find predictive stats for identifying the winner of a game

Contributions

Andrew Krikorian - Creating and Evaluating Prediction Models

Ehtan Douglas - MVP data exploration and analysis

Harjas Dhaliwal - Championship data exploration and analysis

Justin Reveredo - Winning Teams data exploration and analysis

Ibrahim Ahmad Elhajjmoussa - Data Cleaning, Preparation and Preprocessing

Dataset

Link to Data

Link: https://hoopr.sportsdataverse.org/

Data Overview

HoopR has raw data from 2010 to 2024 from NBA database. This data has over 500,000 rows of data with 57 columns which includes play by plays, team stats, player stats. For each of our goals, we will create subsets of data for analysis and another subset for predictions. Our winning teams dataset will have game statistics over the whole season per team like points, rebounds and steals in each game. Our championship dataset will have the total statistics over the season as well as the average statistics such as total points in the season as well as average points per game. Our MVP dataset will have individual player statistics including totals and averages like total points in a season and average points per game.

Importing Dataset and Dependencies

```
#install.packages("hoopR")
#install.packages('reshape2')
#install.packages("devtools")
#devtools::install_github("ricardo-bion/ggradar", dependencies = TRUE)
library(car)
library(tidyverse)
library("hoopR")
library(ggradar)
library(reshape2)
library(broom)
library(leaps)
library(gridExtra)
library(boot)
```

Aggregating Data

Our first task is to aggregate all the data so that we can do some EDA and start creating the prediciton models. As was mentioned before, we need to create three datasets for each goal's EDA and three datasets for each goal's prediction model.

Gathering all box scores from 2010 to 2024

```
nba_team_box <- load_nba_team_box(seasons = 2010:most_recent_nba_season())
head(nba_team_box, 5)
```

```
## # A tibble: 5 x 57
##     game_id season season_type game_date  game_date_time      team_id team_uid
##       <int>  <int>       <int> <date>     <dttm>                <int> <chr>
## 1 300617013   2010           3 2010-06-17 2010-06-17 21:00:00       2 s:40~l:46~
```

```
## 2 300617013   2010           3 2010-06-17 2010-06-17 21:00:00      13 s:40~l:46~
## 3 300615013   2010           3 2010-06-15 2010-06-15 21:00:00       2 s:40~l:46~
## 4 300615013   2010           3 2010-06-15 2010-06-15 21:00:00      13 s:40~l:46~
## 5 300613002   2010           3 2010-06-13 2010-06-13 20:00:00      13 s:40~l:46~
## # i 50 more variables: team_slug <chr>, team_location <chr>, team_name <chr>,
## #   team_abbreviation <chr>, team_display_name <chr>,
## #   team_short_display_name <chr>, team_color <chr>,
## #   team_alternate_color <chr>, team_logo <chr>, team_home_away <chr>,
## #   team_score <int>, team_winner <lgl>, assists <int>, blocks <int>,
## #   defensive_rebounds <int>, fast_break_points <chr>, field_goal_pct <dbl>,
## #   field_goals_made <int>, field_goals_attempted <int>, ...
```

Cleaning our data by removing all unwanted variables

```
selected_columns <- nba_team_box %>%
  select(team_name, season, team_home_away, team_winner, assists, blocks, defensive_rebounds, fast_breal
```

Using groupby to create our aggregated team stats dataset

```
team_stats <- nba_team_box %>%
  group_by(team_name, season) %>%
  reframe(
    name = team_name,
    abr = team_abbreviation,
    season = season,
    avg_PTS = mean(team_score, na.rm = TRUE),
    avg_ALLPTS = mean(opponent_team_score, na.rm = TRUE),
    avg_FLS = mean(fouls, na.rm = TRUE),
    avg_STLS = mean(steals, na.rm = TRUE),
    avg_BLKS = mean(blocks, na.rm = TRUE),
    avg_FGM = mean(field_goals_made, na.rm = TRUE),
    avg_FGA = mean(field_goals_attempted, na.rm = TRUE),
    avg_3PM = mean(three_point_field_goals_made, na.rm = TRUE),
    avg_3PA = mean(three_point_field_goals_attempted, na.rm = TRUE),
    avg_MINS = mean(minutes, na.rm = TRUE),
    avg_FTM  = mean(free_throws_made, na.rm = TRUE),
    avg_FTA  = mean(free_throws_attempted, na.rm = TRUE),
    avg_OREBS  = mean(offensive_rebounds, na.rm = TRUE),
    avg_DREBS  = mean(defensive_rebounds, na.rm = TRUE),
    avg_REBS = mean(offensive_rebounds+defensive_rebounds, na.rm = TRUE),
    avg_ASTS  = mean(assists, na.rm = TRUE),
    avg_TURN  = mean(turnovers, na.rm = TRUE),
    avg_FBPTS  = mean(fast_break_points, na.rm = TRUE),
    avg_LEAD = mean(largest_lead, na.rm = TRUE),
    avg_PIP = mean(points_in_paint, na.rm = TRUE),
  )
```

```
team_stats_2010_2024 <- team_stats %>%
  distinct(name, season, .keep_all = TRUE)
head(team_stats_2010_2024, 5)
```

```
## # A tibble: 5 x 24
##   team_name season name  abr   avg_PTS avg_ALLPTS avg_FLS avg_STLS avg_BLKS
##   <chr>      <int> <chr> <chr>   <dbl>      <dbl>   <dbl>    <dbl>    <dbl>
## 1 76ers       2010 76ers PHI      97.7      102.     20.5     8.13     5.35
## 2 76ers       2011 76ers PHI      98.3       97.3    19.5     7.51     4.29
## 3 76ers       2012 76ers PHI      92.3       89.0    17.7     7.91     5.18
## 4 76ers       2013 76ers PHI      93.2       96.5    18.4     7.43     4.68
## 5 76ers       2014 76ers PHI      99.5      110.     22.5     9.33     4.02
## # i 15 more variables: avg_FGM <dbl>, avg_FGA <dbl>, avg_3PM <dbl>,
## #   avg_3PA <dbl>, avg_MINS <dbl>, avg_FTM <dbl>, avg_FTA <dbl>,
## #   avg_OREBS <dbl>, avg_DREBS <dbl>, avg_REBS <dbl>, avg_ASTS <dbl>,
## #   avg_TURN <dbl>, avg_FBPTS <dbl>, avg_LEAD <dbl>, avg_PIP <dbl>
```

```
seasons <- c(2010:2024)

for (year in seasons) {
  season_df <- subset(team_stats_2010_2024, season == year)
  season_name <- paste("team_stats", year, sep = "_")
  assign(season_name, season_df)
}
```

Gathering data for our aggregated player dataset

```
player_box <- load_nba_player_box(seasons = 2010:most_recent_nba_season())
head(player_box, 5)
```

```
## # A tibble: 5 x 57
##     game_id season season_type game_date  game_date_time      athlete_id
##       <int>  <int>       <int> <date>     <dttm>                   <int>
## 1 300617013   2010           3 2010-06-17 2010-06-17 21:00:00        261
## 2 300617013   2010           3 2010-06-17 2010-06-17 21:00:00        883
## 3 300617013   2010           3 2010-06-17 2010-06-17 21:00:00        662
## 4 300617013   2010           3 2010-06-17 2010-06-17 21:00:00       3026
## 5 300617013   2010           3 2010-06-17 2010-06-17 21:00:00          9
## # i 51 more variables: athlete_display_name <chr>, team_id <int>,
## #   team_name <chr>, team_location <chr>, team_short_display_name <chr>,
## #   minutes <dbl>, field_goals_made <int>, field_goals_attempted <int>,
## #   three_point_field_goals_made <int>,
## #   three_point_field_goals_attempted <int>, free_throws_made <int>,
## #   free_throws_attempted <int>, offensive_rebounds <int>,
## #   defensive_rebounds <int>, rebounds <int>, assists <int>, steals <int>, ...
```

Getting average stats per game by calculating the means of aggregated data

```
player_stats <- player_box %>%
  group_by(athlete_display_name, season) %>%
  reframe(
    name = athlete_display_name,
    short_name = athlete_short_name,
```

```
    season = season,
    avg_PTS = mean(points, na.rm = TRUE),
    avg_FLS = mean(fouls, na.rm = TRUE),
    avg_STLS = mean(steals, na.rm = TRUE),
    avg_REBS = mean(rebounds, na.rm = TRUE),
    avg_BLKS = mean(blocks, na.rm = TRUE),
    avg_FGM = mean(field_goals_made, na.rm = TRUE),
    avg_FGA = mean(field_goals_attempted, na.rm = TRUE),
    avg_3PM = mean(three_point_field_goals_made, na.rm = TRUE),
    avg_3PA = mean(three_point_field_goals_attempted, na.rm = TRUE),
    avg_MINS = mean(minutes, na.rm = TRUE),
    avg_FTM  = mean(free_throws_made, na.rm = TRUE),
    avg_FTA  = mean(free_throws_attempted, na.rm = TRUE),
    avg_OREBS  = mean(offensive_rebounds, na.rm = TRUE),
    avg_DREBS  = mean(defensive_rebounds, na.rm = TRUE),
    avg_ASTS  = mean(assists, na.rm = TRUE),
    avg_TURN  = mean(turnovers, na.rm = TRUE),
  )
```

```
player_stats_2010_2024 <- player_stats %>%
  distinct(name, season, .keep_all = TRUE)
head(player_stats_2010_2024, 5)
```

```
## # A tibble: 5 x 20
##   athlete_display_name season name  short_name avg_PTS avg_FLS avg_STLS avg_REBS
##   <chr>                 <int> <chr> <chr>        <dbl>   <dbl>    <dbl>    <dbl>
## 1 A.J. Lawson            2023 A.J.~ A.J. Laws~    3.73   0.733    0.133      1.4
## 2 A.J. Lawson            2024 A.J.~ A.J. Laws~    2.9    0.46     0.2        1.06
## 3 A.J. Price             2010 A.J.~ A.J. Price    7.32   0.946    0.625      1.57
## 4 A.J. Price             2011 A.J.~ A.J. Price    6.64   1.2      0.582      1.44
## 5 A.J. Price             2012 A.J.~ A.J. Price    3.78   0.652    0.435      1.37
## # i 12 more variables: avg_BLKS <dbl>, avg_FGM <dbl>, avg_FGA <dbl>,
## #   avg_3PM <dbl>, avg_3PA <dbl>, avg_MINS <dbl>, avg_FTM <dbl>, avg_FTA <dbl>,
## #   avg_OREBS <dbl>, avg_DREBS <dbl>, avg_ASTS <dbl>, avg_TURN <dbl>
```

```
seasons <- c(2010:2024)

for (year in seasons) {
  season_df <- subset(player_stats_2010_2024, season == year)
  season_name <- paste("player_stats", year, sep = "_")
  assign(season_name, season_df)
}
```

```
all_players <- nba_commonallplayers(league_id = '00', season = year_to_season(most_recent_nba_season() -
head(all_players, 5)
```

```
## $CommonAllPlayers
## # A tibble: 4,920 x 16
##    PERSON_ID DISPLAY_LAST_COMMA_FIRST DISPLAY_FIRST_LAST  ROSTERSTATUS FROM_YEAR
##    <chr>     <chr>                    <chr>               <chr>        <chr>
##  1 76001     Abdelnaby, Alaa          Alaa Abdelnaby      0            1990
##  2 76002     Abdul-Aziz, Zaid         Zaid Abdul-Aziz     0            1968
```

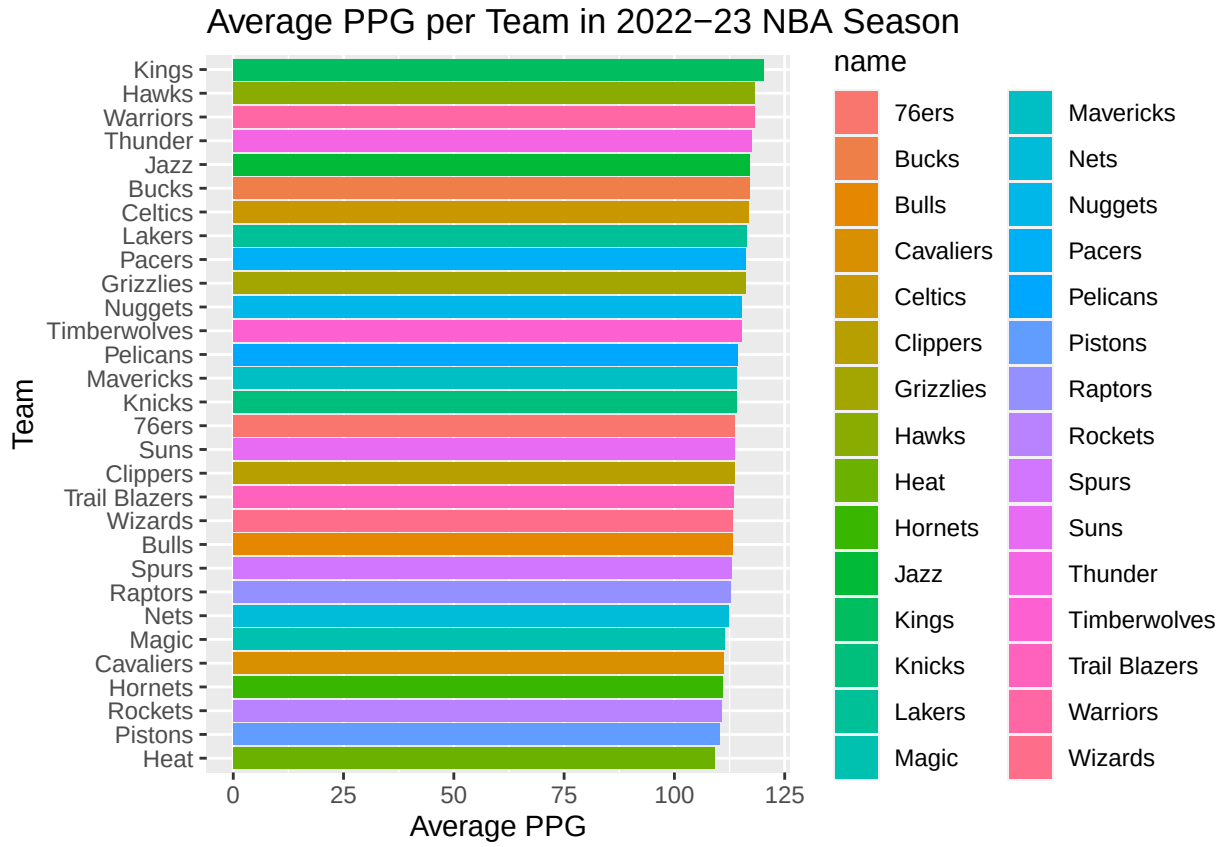
```
##  3 76003     Abdul-Jabbar, Kareem     Kareem Abdul-Jabbar 0            1969
##  4 51        Abdul-Rauf, Mahmoud      Mahmoud Abdul-Rauf  0            1990
##  5 1505      Abdul-Wahad, Tariq       Tariq Abdul-Wahad   0            1997
##  6 949       Abdur-Rahim, Shareef     Shareef Abdur-Rahim 0            1996
##  7 76005     Abernethy, Tom           Tom Abernethy       0            1976
##  8 76006     Able, Forest             Forest Able         0            1956
##  9 76007     Abramovic, John          John Abramovic      0            1946
## 10 203518    Abrines, Alex            Alex Abrines        0            2016
## # i 4,910 more rows
## # i 11 more variables: TO_YEAR <chr>, PLAYERCODE <chr>, PLAYER_SLUG <chr>,
## #   TEAM_ID <chr>, TEAM_CITY <chr>, TEAM_NAME <chr>, TEAM_ABBREVIATION <chr>,
## #   TEAM_CODE <chr>, TEAM_SLUG <chr>, GAMES_PLAYED_FLAG <chr>,
## #   OTHERLEAGUE_EXPERIENCE_CH <chr>
```

Exploratory Data Analysis

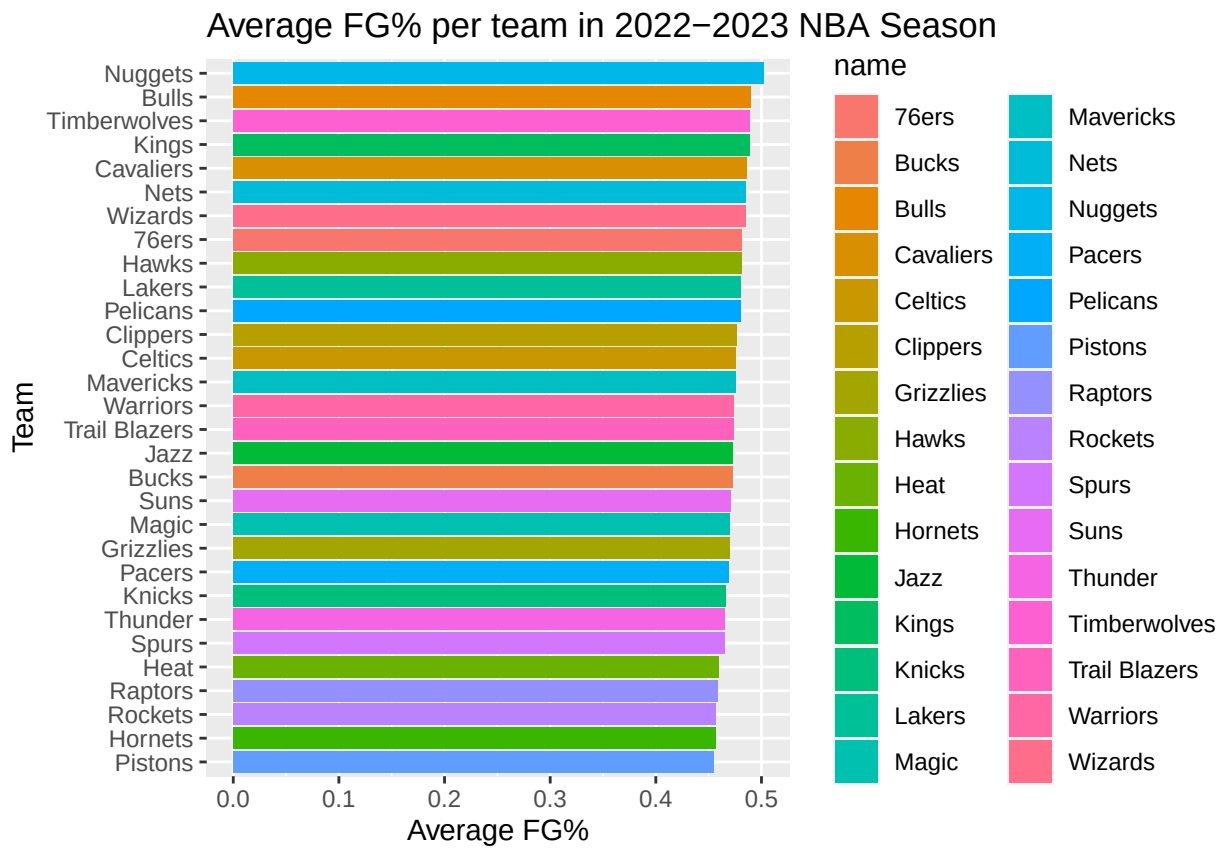
Championships

For the championship analysis, we analyzed the team statistics from the 2022-2023 NBA season to visualize how each team's attributes varies. The attributes that were examined were team averages of points per game (PPG), field goal percentage (FG%), rebounds per game (RPG), assists per game (APG), steals per game (SPG), blocks per game (BPG), and free throw percentage (FT%). We also filtered out all-star teams (Team Lebron, Team Giannis) to concentrate on the regular NBA teams to identify patterns associated with championship success. We know that the Denver Nuggets won the championship in the 2022-2023 season, therefore with that in mind, we examined what attributes that the Nuggets excelled in to see if a certain attribute may be more significant to winning a championship than others. We noticed that the Nuggets were #1 in average FG% and #2 in average APG. To try to identify trends to see if certain attributes are more significant than others, we can change the year in filter(season) and compare it with the championship winner of that year.

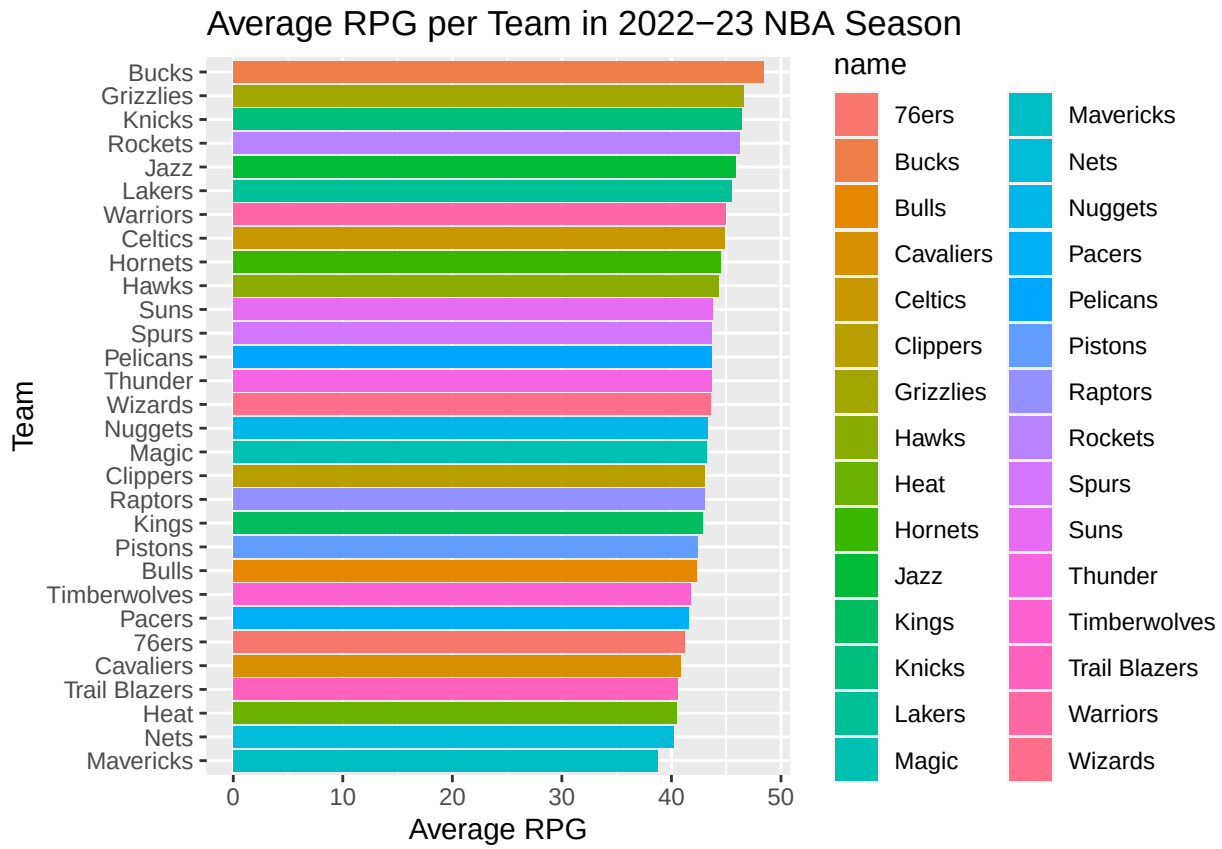
```
cleaned_team_stats_2010_2024 <- subset(team_stats_2010_2024, team_name != "Team LeBron" & team_name !=
# PPG
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_PTS), y = avg
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average PPG", title = "Average PPG per Team in 2022-23 NBA Season")
```



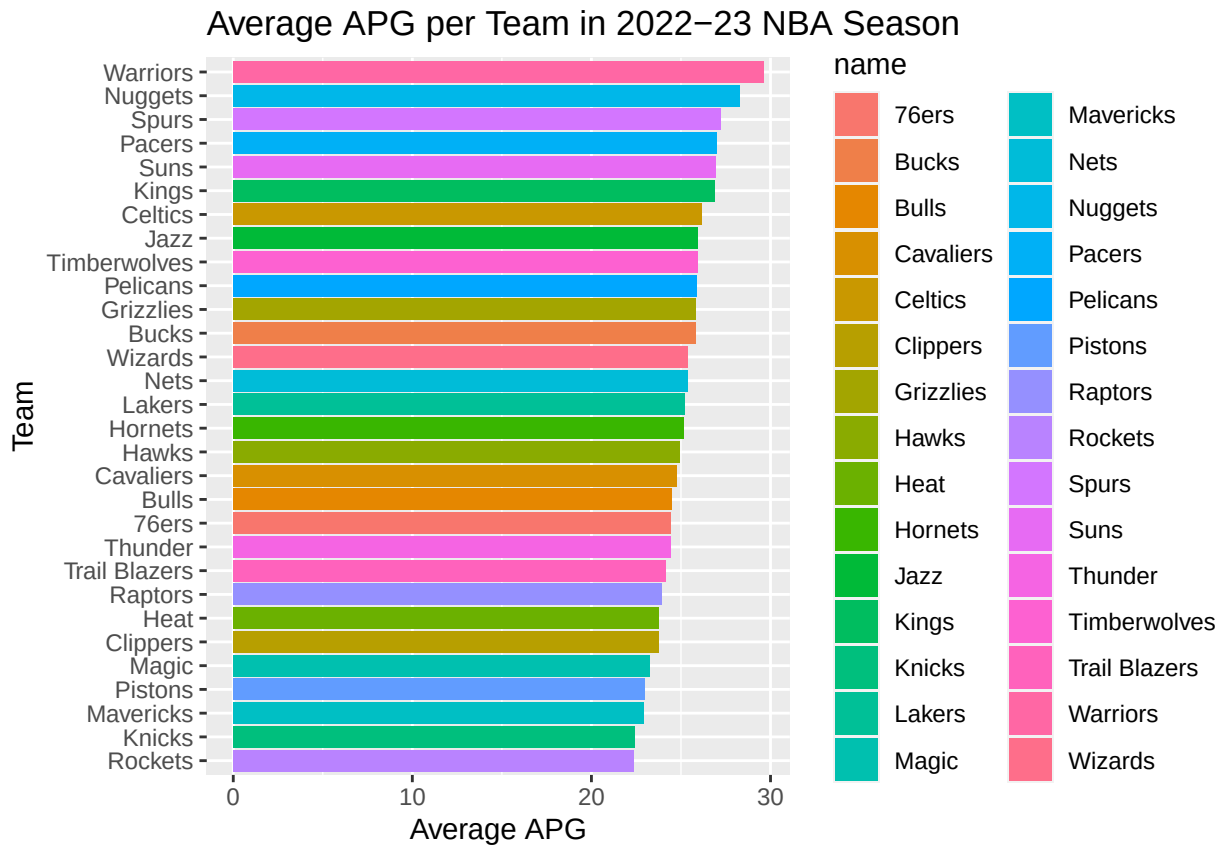
```r
# FG%
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_FGM / avg_FGA
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average FG%", title = "Average FG% per team in 2022-2023 NBA Season")
```



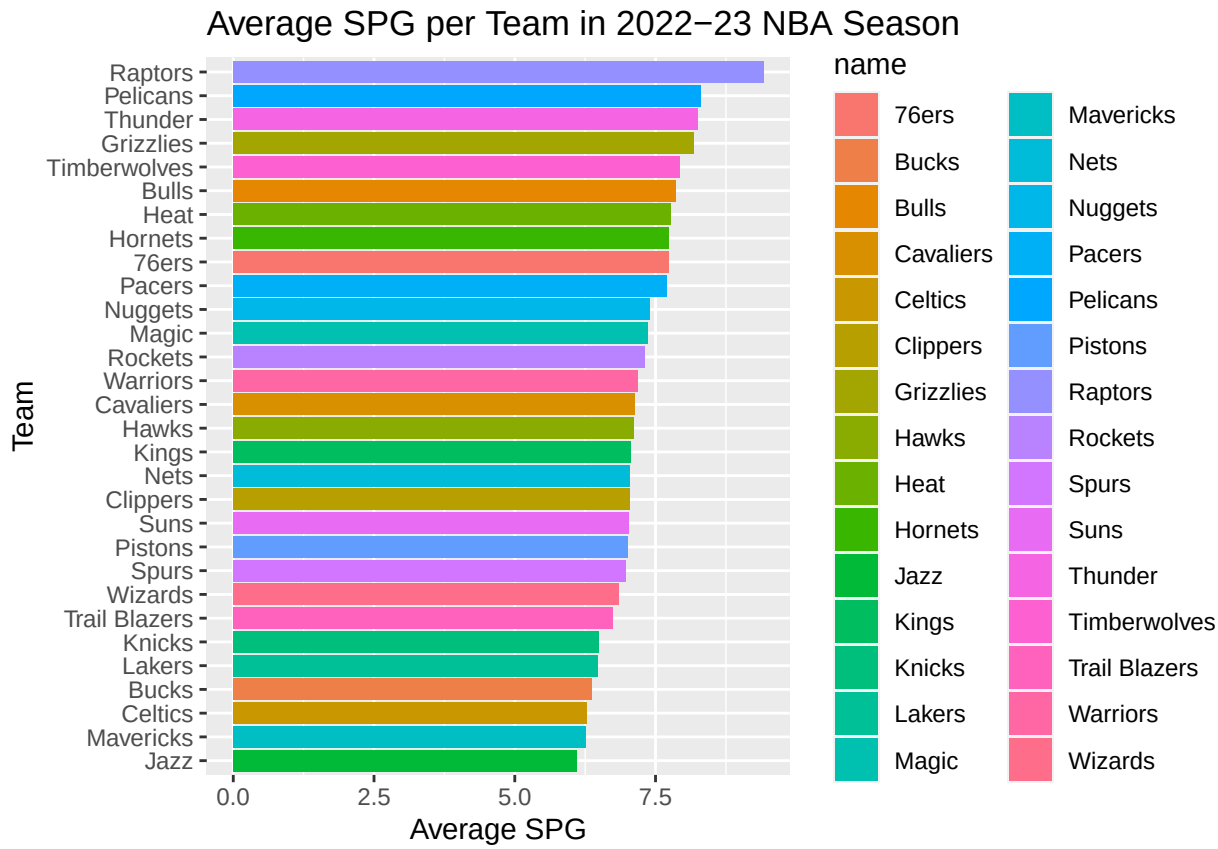

```r
# RPG
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_REBS), y = avg
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average RPG", title = "Average RPG per Team in 2022-23 NBA Season")
```



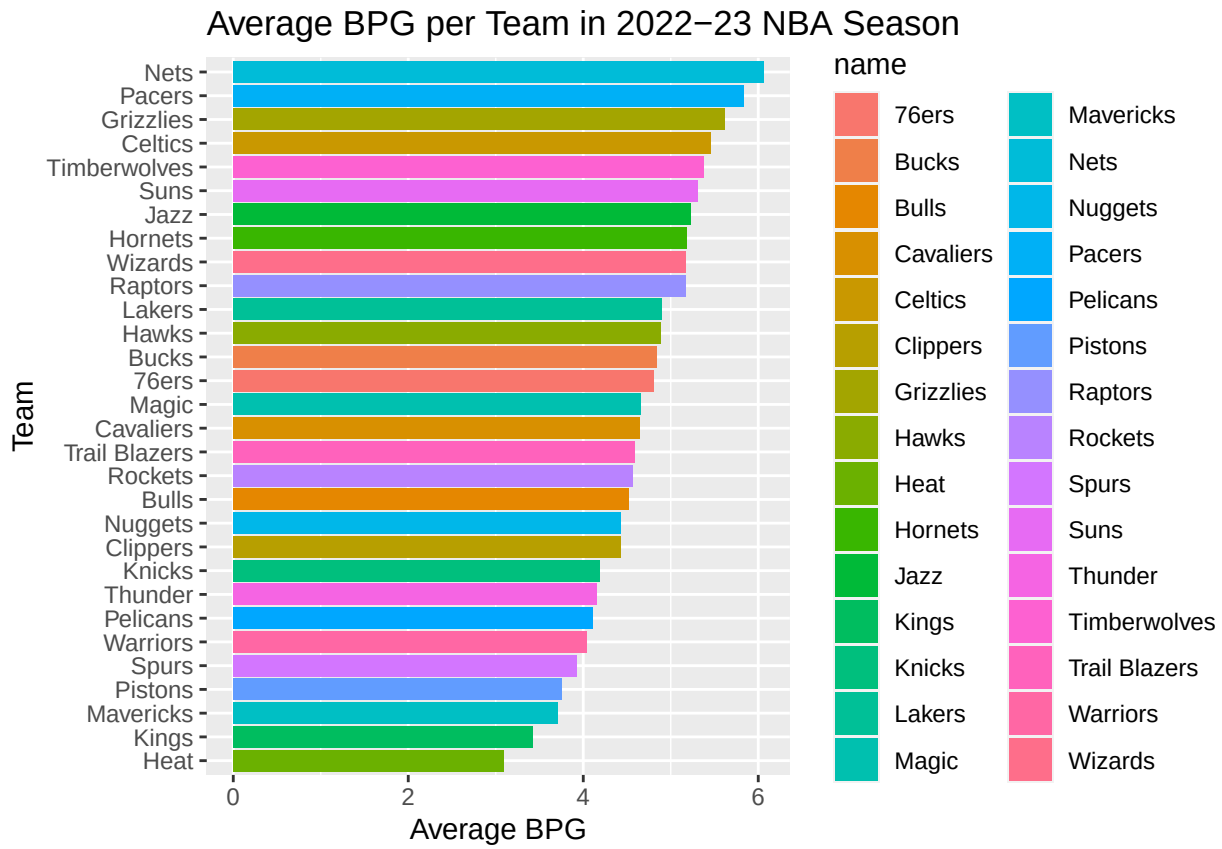

```r
# APG
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_ASTS), y = avg
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average APG", title = "Average APG per Team in 2022-23 NBA Season")
```



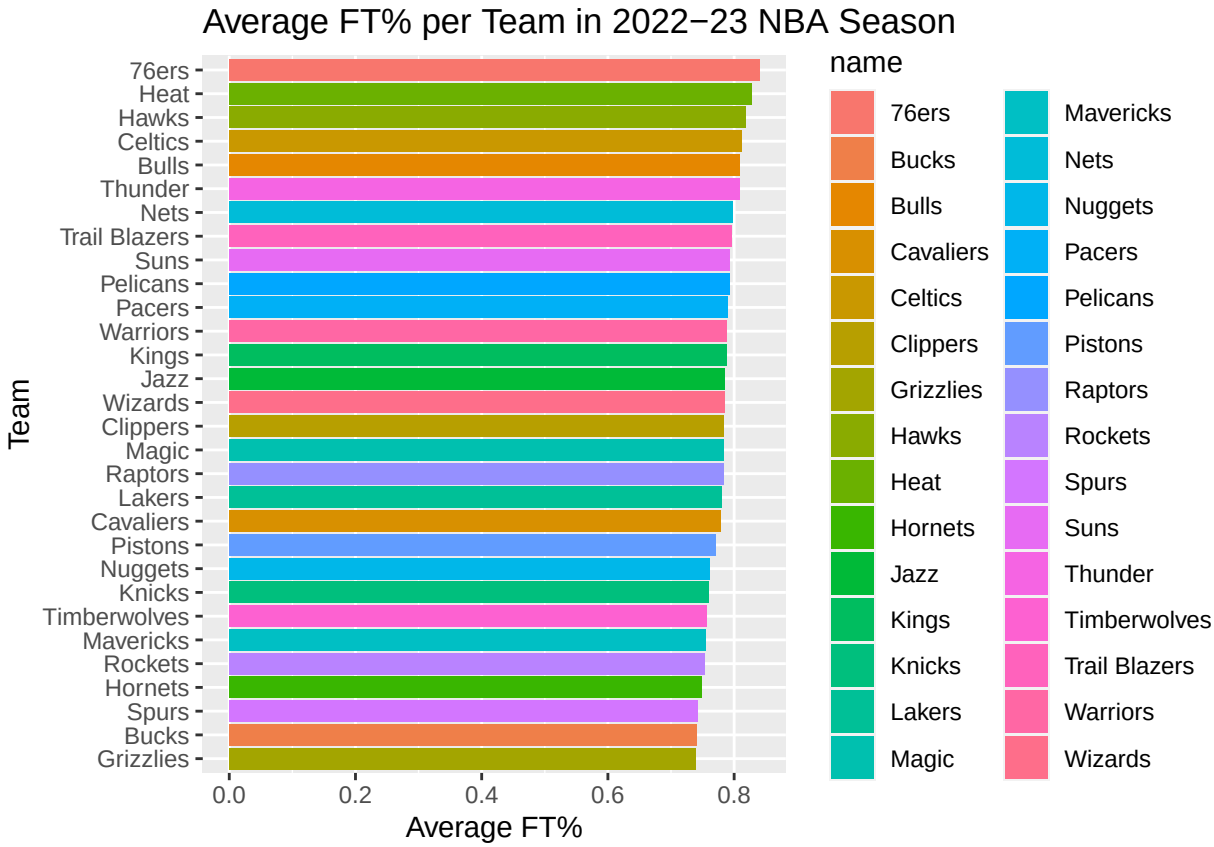

```r
# SPG
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_STLS), y = avg
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average SPG", title = "Average SPG per Team in 2022-23 NBA Season")
```




```r
# BPG
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_BLKS), y = avg
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average BPG", title = "Average BPG per Team in 2022-23 NBA Season")
```




```r
# FT%
ggplot(cleaned_team_stats_2010_2024 %>% filter(season == 2023), aes(x = reorder(name, avg_FTM / avg_FTA)
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Team", y = "Average FT%", title = "Average FT% per Team in 2022-23 NBA Season")
```


Average FT% per Team in 2022−23 NBA Season
Team
76ers
Heat
Hawks
Celtics
Bulls
Thunder
Nets
Trail Blazers
Suns
Pelicans
Pacers
Warriors
Kings
Jazz
Wizards
Clippers
Magic
Raptors
Lakers
Cavaliers
Pistons
Nuggets
Knicks
Timberwolves
Mavericks
Rockets
Hornets
Spurs
Bucks
Grizzlies
0.0
0.2
0.4
0.6
0.8
Average FT%
name
76ers
Bucks
Bulls
Cavaliers
Celtics
Clippers
Grizzlies
Hawks
Heat
Hornets
Jazz
Kings
Knicks
Lakers
Magic
Mavericks
Nets
Nuggets
Pacers
Pelicans
Pistons
Raptors
Rockets
Spurs
Suns
Thunder
Timberwolves
Trail Blazers
Warriors
Wizards

MVPs

We created a dataframe of each MVP and their corresponding stats from the season they won MVP. We gathered this data from basketballreference.com, which is one of the most used NBA statistics websites. After verifying the correctness of their data, we then input the data into our project.

```
name <- c("Nikola Jokic", "Joel Embiid", "Nikola Jokic", "Nikola Jokic", "Giannis Antetokounmpo", "Gian
season <- C(2024:2010)
position <- c("C", "C", "C", "C", "F", "F", "G", "G", "G", "G", "F", "F", "SF", "PG", "SF")
fg_pct <- c(0.583, 0.548, 0.583, 0.566, 0.553, 0.578, 0.449, 0.425, 0.504, 0.487, 0.503, 0.565, 0.531, 
ppg <- c(26.4, 33.1, 27.1, 26.4, 29.5, 27.7, 30.4, 31.6, 30.1, 23.8, 32, 26.8, 27.1, 25, 29.7)
rpg <- c(12.4, 10.2, 13.8, 10.8, 13.6, 12.5, 5.4, 10.7, 5.4, 4.3, 7.4, 8, 7.9, 4.1, 7.3)
apg <- c(9, 4.2, 7.9, 8.3, 5.6, 5.9, 8.8, 10.4, 6.7, 7.7, 5.5, 7.3, 6.2, 7.7, 8.6)
blks <- c(0.9, 1.7, 0.9, 0.7, 1, 1.5, 0.7, 0.4, 0.2, 0.2, 0.7, 0.9, 0.8, 0.6, 1)

mvp_data <- cbind.data.frame(name, season, position, fg_pct, ppg, rpg, apg, blks)
mvp_data
```

The first step we took in our EDA for MVP winners was to look at how much better they are at each facet of the game than their peers. The method we used to quantify this was to use percentile rankings (99th percentile means 99% of players got less of a certain stat than out MVP winner). In order to calculate the percentile of all MVPs, we counted how many total players there were and then counted how many total players scored lower in each stat than our MVPS. Then, by dividing the total number lower by the total count, we obntained our percentile.

```
player_stats_2010_2024 <- na.omit(player_stats_2010_2024)

#The get_percentile function allows us to run this code snippet as many times as needed
get_percentile <- function(year, a) {
  current_year <- filter(player_stats_2010_2024, season == year)

  if(a == "ppg")
  {
    mvp_stat <- filter(mvp_data, season == year)$ppg
    more_than_mvp <- nrow(filter(current_year, avg_PTS >= mvp_stat))
    percentile <- round((nrow(current_year) - more_than_mvp) / nrow(current_year), 2) * 100
  } else if(a == "rpg")
  {
    mvp_stat <- filter(mvp_data, season == year)$rpg
    more_than_mvp <- nrow(filter(current_year, avg_REBS >= mvp_stat))
    percentile <- round((nrow(current_year) - more_than_mvp) / nrow(current_year), 2) * 100
  } else if(a == "apg")
  {
    mvp_stat <- filter(mvp_data, season == year)$apg
    more_than_mvp <- nrow(filter(current_year, avg_ASTS >= mvp_stat))
    percentile <- round((nrow(current_year) - more_than_mvp) / nrow(current_year), 2) * 100
  } else if(a == "blks")
  {
    mvp_stat <- filter(mvp_data, season == year)$blks
    more_than_mvp <- nrow(filter(current_year, avg_BLKS >= mvp_stat))
    percentile <- round((nrow(current_year) - more_than_mvp) / nrow(current_year), 2) * 100
  }
```

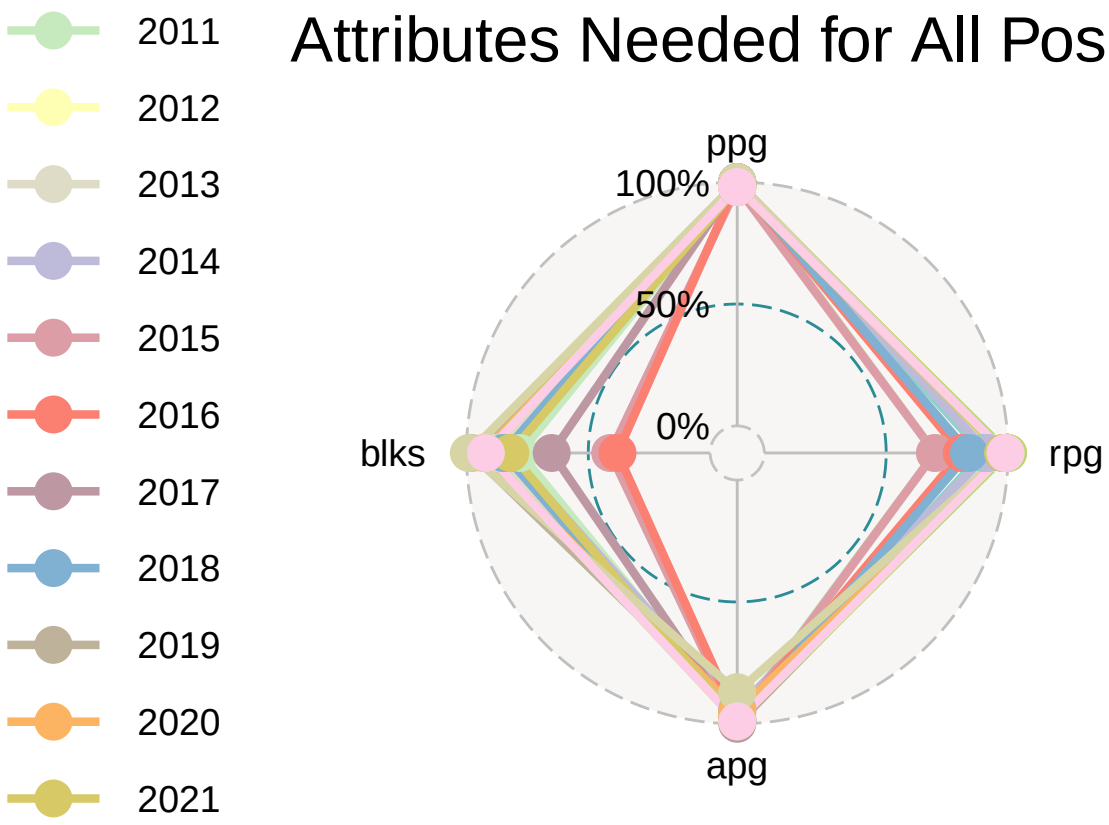
```
  return(percentile)
}
```

Calculating percentiles in each stat for each MVP

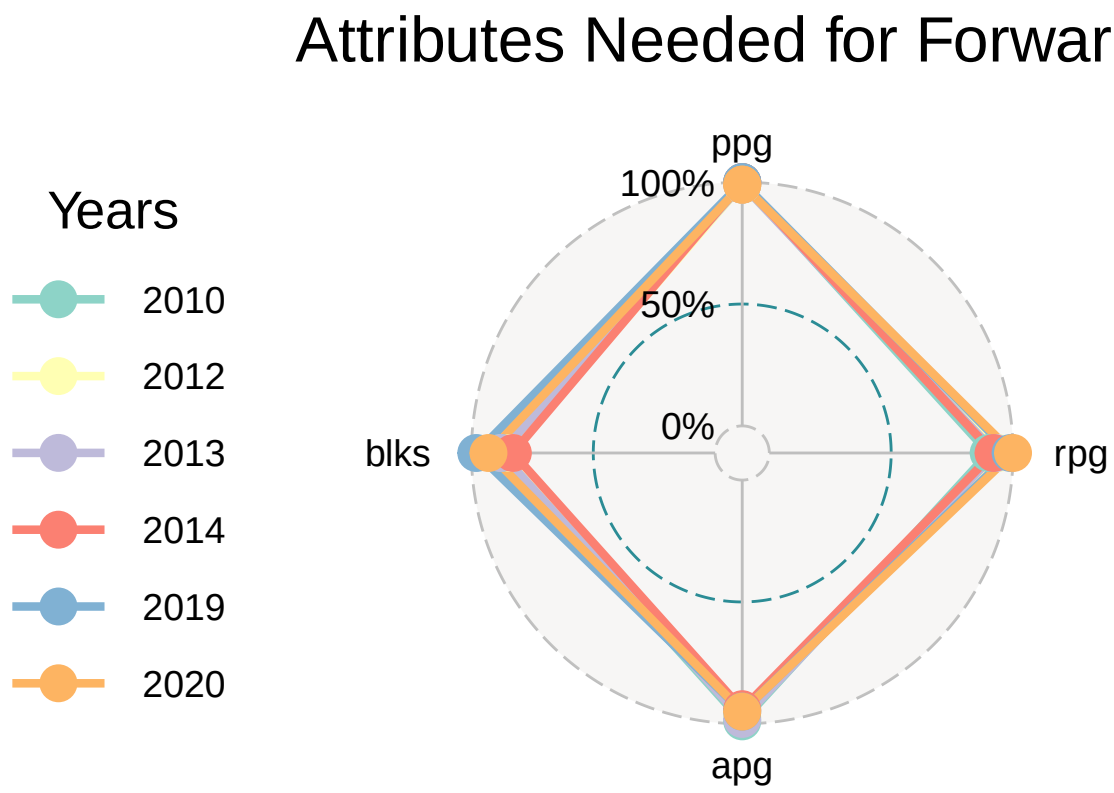
```
percentile_ppg <- c(get_percentile(2024, "ppg"), get_percentile(2023, "ppg"), get_percentile(2022, "ppg

percentile_rpg <- c(get_percentile(2024, "rpg"), get_percentile(2023, "rpg"), get_percentile(2022, "rpg

percentile_blks <- c(get_percentile(2024, "blks"), get_percentile(2023, "blks"), get_percentile(2022, "

percentile_apg <- c(get_percentile(2024, "apg"), get_percentile(2023, "apg"), get_percentile(2022, "apg

all_percentiles <- cbind.data.frame(season, percentile_ppg, percentile_rpg, percentile_apg, percentile_
names(all_percentiles) <- c("season", "ppg", "rpg", "apg", "blks")

center_percentiles <- filter(all_percentiles, season %in% c("2021", "2022", "2023", "2024"))
guard_percentiles <- filter(all_percentiles, season %in% c("2011", "2015", "2016", "2017", "2018"))
forward_percentiles <- filter(all_percentiles, season %in% c("2010", "2012", "2013", "2014", "2019", "20
```

In order to visualize the percentile rankings among MVP winners, we grouped them by position. This is because different positions require different skills in order to be considered an elite player. For example, point guards do not usually possess the rebounding ability of a center, but most centers do not rack up assists like point guards do. We chose to use radar plots to display this data because it shows what aspects of the game each player is dominant at. The way to read these charts is the closer to a percent diamond shape, the more well rounded a player is.

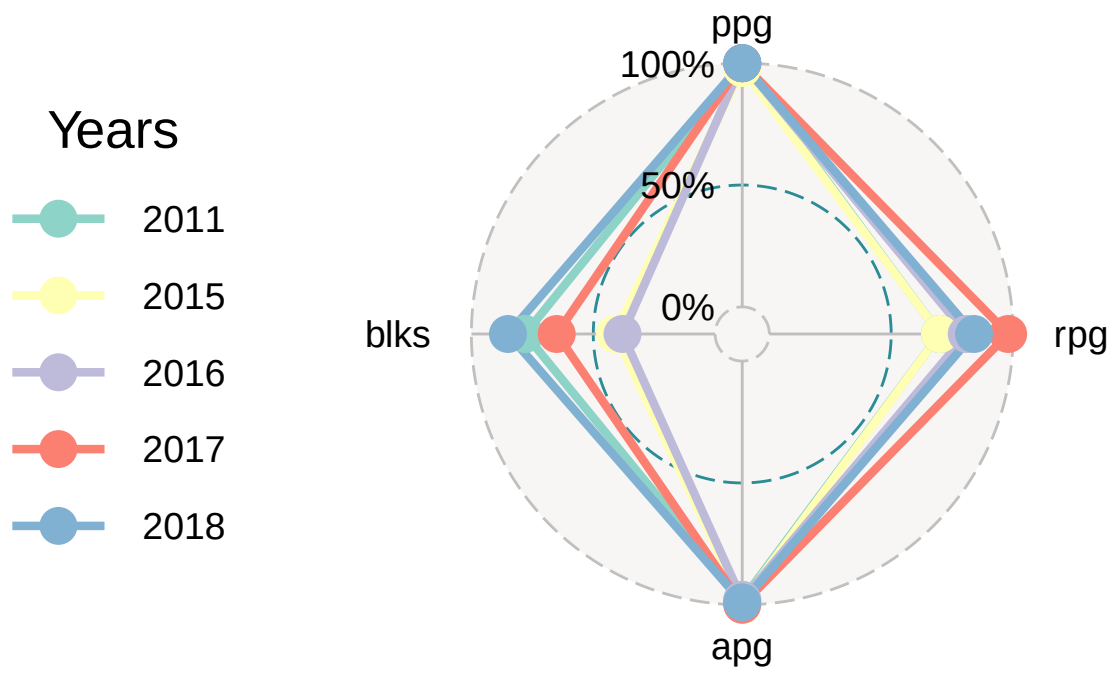
```
ggradar(all_percentiles, plot.title = "Attributes Needed for All Positions", legend.title = "Years")
```




```
#The radar plot with all MVP winners was too crowded, so display by position group
ggradar(forward_percentiles, plot.title = "Attributes Needed for Forwards", legend.title = "Years")
```

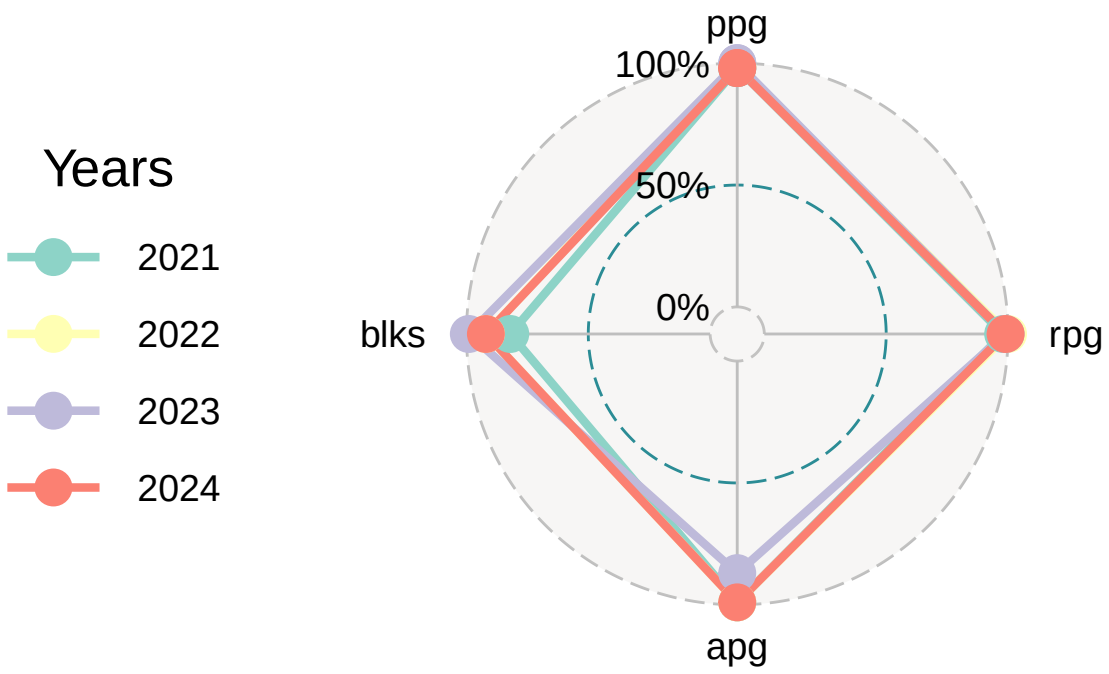



```
ggradar(guard_percentiles, plot.title = "Attributes Needed for Guards", legend.title = "Years")
```




```
ggradar(center_percentiles, plot.title = "Attributes Needed for Centers", legend.title = "Years")
```

Attributes Needed for Centers



Winning Teams

```
selected_columns <- selected_columns %>%
  mutate(team_winner = as.numeric(team_winner))

head(selected_columns, 5)
```

```
## # A tibble: 5 x 23
##   team_name season team_home_away team_winner assists blocks defensive_rebounds
##   <chr>      <int> <chr>                <dbl>   <int>  <int>              <int>
## 1 Celtics     2010 away                     0      18      7                 32
## 2 Lakers      2010 home                     1      11      3                 30
## 3 Celtics     2010 away                     0      17      4                 28
## 4 Lakers      2010 home                     1      17      8                 40
## 5 Lakers      2010 away                     0      12      1                 18
## # i 16 more variables: fast_break_points <chr>, field_goal_pct <dbl>,
## #   field_goals_made <int>, fouls <int>, free_throws_made <int>,
## #   largest_lead <chr>, offensive_rebounds <int>, points_in_paint <chr>,
## #   steals <int>, team_turnovers <int>, three_point_field_goals_made <int>,
## #   total_rebounds <int>, total_turnovers <int>, turnover_points <chr>,
## #   turnovers <int>, technical_fouls <int>
```

```
selected_columns <- selected_columns %>%
  mutate(
    team_winner = as.numeric(team_winner),
    fast_break_points = as.numeric(fast_break_points),
    largest_lead = as.numeric(largest_lead),
    points_in_paint = as.numeric(points_in_paint),
    turnover_points = as.numeric(turnover_points)
  )

head(selected_columns, 5)
```

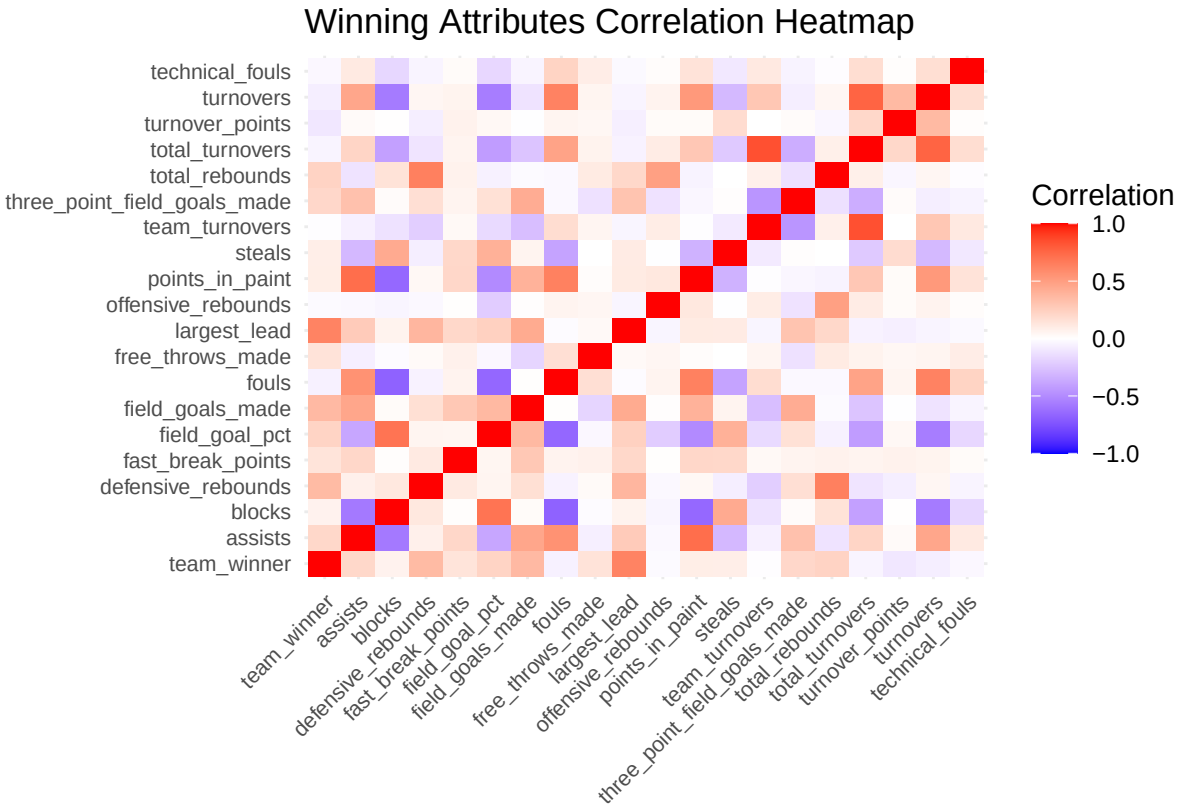
```
## # A tibble: 5 x 23
##   team_name season team_home_away team_winner assists blocks defensive_rebounds
##   <chr>      <int> <chr>                <dbl>   <int>  <int>              <int>
## 1 Celtics     2010 away                     0      18      7                 32
## 2 Lakers      2010 home                     1      11      3                 30
## 3 Celtics     2010 away                     0      17      4                 28
## 4 Lakers      2010 home                     1      17      8                 40
## 5 Lakers      2010 away                     0      12      1                 18
## # i 16 more variables: fast_break_points <dbl>, field_goal_pct <dbl>,
## #   field_goals_made <int>, fouls <int>, free_throws_made <int>,
## #   largest_lead <dbl>, offensive_rebounds <int>, points_in_paint <dbl>,
## #   steals <int>, team_turnovers <int>, three_point_field_goals_made <int>,
## #   total_rebounds <int>, total_turnovers <int>, turnover_points <dbl>,
## #   turnovers <int>, technical_fouls <int>
```

```
numeric_columns <- selected_columns %>%
  select(team_winner, assists, blocks, defensive_rebounds, fast_break_points, field_goal_pct,
         field_goals_made, fouls, free_throws_made, largest_lead, offensive_rebounds,
```

```
        points_in_paint, steals, team_turnovers, three_point_field_goals_made, total_rebounds,
        total_turnovers, turnover_points, turnovers, technical_fouls)

# Calculate the correlation matrix
correlation_matrix <- cor(numeric_columns, use = "complete.obs")
```

Here, we selected columns that we deemed important to our initial question of what makes a team win. We took out extremely specific statistics/variables to make it easier to understand. After we isolated the main categories that we wanted to interpret, we then made a correlation matrix with these variables. These variables were: team_winner, assits, blocks, defensive rebounds, fast break points, field goals made, fouls, free throws made, largest lead, offensive rebounds, points in point, steals, team turnovers, three point field goals made, total rebounds, total turnovers, total turnovers, turnover points, turnovers, and technical fouls. With tis wide plethora of variables, we look for the significance between each one.

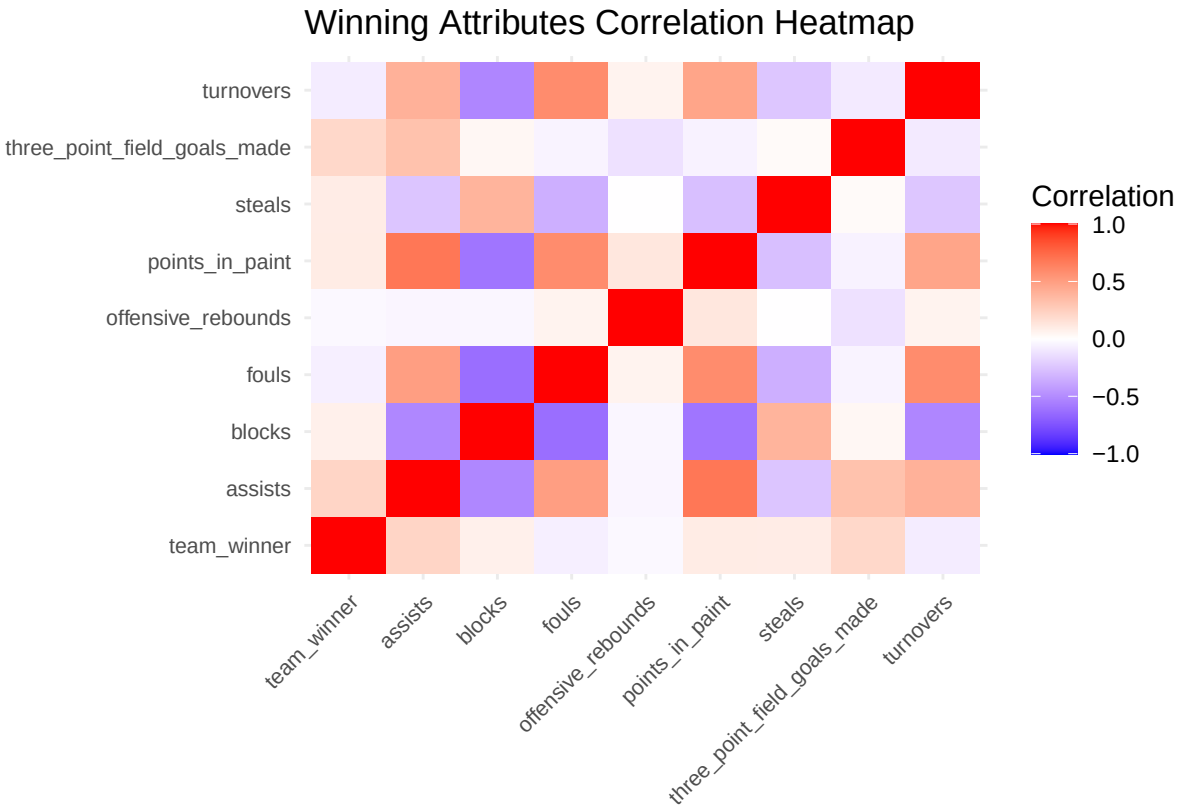
```
heatmap_data <- melt(correlation_matrix)

ggplot(heatmap_data, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text = element_text(size = 8)) +
  labs(title = "Winning Attributes Correlation Heatmap",
       x = "",
       y = "")
```



We decided to generate a heat map with the correlation matrix that we had generated in the previous code cell. This heatmap allows us to better interpret the relationship between each variable, and most importantly our output variable (team_winner). As we are looking at what variables have the most positive and negative correlations. The output variable of 1 means that the team has won and the output variable of 0 means the team has lost. The output variable shows a strong positive correlation with "assists," "defensive_rebounds," and "field_goal_pct," suggesting that these factors are closely associated with winning. Conversely, "technical_fouls" and "turnovers" exhibit weaker or negative correlations with winning, indicating that fewer of these contribute to better performance.

```
selected_columns <- selected_columns %>%
  mutate(
    team_winner = as.numeric(team_winner),
    fast_break_points = as.numeric(fast_break_points),
    largest_lead = as.numeric(largest_lead),
    points_in_paint = as.numeric(points_in_paint),
    turnover_points = as.numeric(turnover_points)
  )

numeric_columns_small <- selected_columns %>%
  select(team_winner, assists, blocks, fouls, offensive_rebounds,
         points_in_paint, steals, three_point_field_goals_made,
         turnovers)

# Calculate the correlation matrix
correlation_matrix_small <- cor(numeric_columns_small, use = "complete.obs")
```

```r
heatmap_data_small <- reshape2::melt(correlation_matrix_small)

ggplot(heatmap_data_small, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text = element_text(size = 8)) +
  labs(title = "Winning Attributes Correlation Heatmap",
       x = "",
       y = "")
```



The first heat map had a large number of variables which made it slightly harder to interpret. Therfore, for this heatmap we decided to lessen the amount of variables and focus on the most important ones. In this, it is important to note that we took out field goal percentage, even though it was a strong predictor. This is because, this is intuitive and it is an added variable that might throw off our interpretation and our goal here is to make it a more transparent dataset. "Team_winner" here shows strong positive correlations with "assists" and "points in paint," ensuring that effective teamwork and scoring in the paint are crucial for success. Conversely, a strong negative correlation with "turnovers" indicates that minimizing turnovers is important for winning. Additionally, "assists" correlate positively with "three_point_field_goals_made" which we can add to a significant variable.

It is also important to note that this a general heatmap(s) of NBA seasons from 2010 to 2024. So, this heatmap is a general trend from all of these seasons however, if we generate a heatmap by season the results

would be different. As the three point field goals would have a stronger correlation with team_winner after 2015 rather than points in the paint. Therefore, there are different trends in the heatmap itself.

Linear Regression Models

Winning Model

For our first prediction model, we are trying to predict the winner of a game. We will start by creating the dataset we will use for the multiple linear regression model. For this goal our target variable is a binary variable which represents a win or a loss for the game. We will start by giving the model all possible predictors then we will use regsubsets to find the best combination of predictors and see the evolution from the first to the second model. After doing so, we will check the assumptions and see if our model is viable.

Setting Up Data

```
nba_team_box_clean <- nba_team_box %>% select(team_winner, defensive_rebounds,field_goals_made, fouls, f
nba_team_box_clean
```

```
## # A tibble: 38,286 x 27
##    team_winner defensive_rebounds field_goals_made fouls free_throws_attempted
##    <lgl>                   <int>            <int> <int>                 <int>
##  1 FALSE                      32               29    25                    17
##  2 TRUE                       30               27    19                    37
##  3 FALSE                      28               28    21                    10
##  4 TRUE                       40               33    17                    19
##  5 FALSE                      18               31    22                    26
##  6 TRUE                       28               40    23                    13
##  7 FALSE                      26               32    23                    22
##  8 TRUE                       25               37    21                    23
##  9 TRUE                       32               34    20                    24
## 10 FALSE                      27               32    27                    24
## # i 38,276 more rows
## # i 22 more variables: points_in_paint <chr>, technical_fouls <int>,
## #   three_point_field_goals_attempted <int>, team_turnovers <int>,
## #   assists <int>, fast_break_points <chr>, field_goals_attempted <int>,
## #   free_throw_pct <dbl>, largest_lead <chr>, steals <int>,
## #   three_point_field_goal_pct <dbl>, total_rebounds <int>,
## #   turnover_points <chr>, team_score <int>, blocks <int>, ...
```

```
nba_team_box_model <- na.omit(nba_team_box_clean)
nba_team_box_model <- nba_team_box_model %>%
  mutate(across(-all_of("team_winner"), as.double))
head(nba_team_box_model, 5)
```

```
## # A tibble: 5 x 27
##   team_winner defensive_rebounds field_goals_made fouls free_throws_attempted
##   <lgl>                   <dbl>            <dbl> <dbl>                 <dbl>
## 1 FALSE                      32               29    25                    17
## 2 TRUE                       30               27    19                    37
## 3 FALSE                      28               28    21                    10
## 4 TRUE                       40               33    17                    19
## 5 FALSE                      18               31    22                    26
## # i 22 more variables: points_in_paint <dbl>, technical_fouls <dbl>,
## #   three_point_field_goals_attempted <dbl>, team_turnovers <dbl>,
```

```
## #   assists <dbl>, fast_break_points <dbl>, field_goals_attempted <dbl>,
## #   free_throw_pct <dbl>, largest_lead <dbl>, steals <dbl>,
## #   three_point_field_goal_pct <dbl>, total_rebounds <dbl>,
## #   turnover_points <dbl>, team_score <dbl>, blocks <dbl>,
## #   field_goal_pct <dbl>, flagrant_fouls <dbl>, free_throws_made <dbl>, ...
```

We have set up the tibble for the prediction model. We have added our target variable in and will now pass it to the first model.

Winning Model with All Variables

```
winning_all_var_model <- lm(team_winner ~ . - three_point_field_goals_made - total_turnovers, data = nba
summary(winning_all_var_model)
```

```
##
## Call:
## lm(formula = team_winner ~ . - three_point_field_goals_made -
##     total_turnovers, data = nba_team_box_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56548 -0.26462 -0.03228  0.27281  1.06469
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -0.0157755  0.0728780  -0.216  0.82863
## defensive_rebounds                 0.0167662  0.0006789  24.695  < 2e-16 ***
## field_goals_made                   0.0192123  0.0047663   4.031 5.57e-05 ***
## fouls                             -0.0072372  0.0005012 -14.440  < 2e-16 ***
## free_throws_attempted             -0.0036084  0.0017729  -2.035  0.04183 *
## points_in_paint                    0.0017577  0.0003053   5.757 8.65e-09 ***
## technical_fouls                   -0.0112093  0.0023374  -4.796 1.63e-06 ***
## three_point_field_goals_attempted -0.0007497  0.0008445  -0.888  0.37470
## team_turnovers                     0.0040769  0.0004522   9.015  < 2e-16 ***
## assists                            0.0057254  0.0005158  11.100  < 2e-16 ***
## fast_break_points                 -0.0027473  0.0003391  -8.102 5.61e-16 ***
## field_goals_attempted             -0.0266517  0.0006304 -42.280  < 2e-16 ***
## free_throw_pct                     0.0031523  0.0004774   6.603 4.10e-11 ***
## largest_lead                       0.0173088  0.0003029  57.145  < 2e-16 ***
## steals                             0.0198254  0.0007335  27.030  < 2e-16 ***
## three_point_field_goal_pct         0.0034138  0.0005600   6.096 1.10e-09 ***
## total_rebounds                     0.0091804  0.0005314  17.277  < 2e-16 ***
## turnover_points                   -0.0052778  0.0003949 -13.363  < 2e-16 ***
## team_score                         0.0061439  0.0022355   2.748  0.00599 **
## blocks                            -0.0008271  0.0007122  -1.161  0.24554
## field_goal_pct                    -0.0047070  0.0006611  -7.120 1.10e-12 ***
## flagrant_fouls                    -0.0008583  0.0020150  -0.426  0.67016
## free_throws_made                   0.0060569  0.0030767   1.969  0.04900 *
## offensive_rebounds                 0.0159117  0.0008490  18.741  < 2e-16 ***
## turnovers                         -0.0181778  0.0007218 -25.186  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.3445 on 30391 degrees of freedom
## Multiple R-squared:  0.5256, Adjusted R-squared:  0.5253
## F-statistic:  1403 on 24 and 30391 DF,  p-value: < 2.2e-16
```

This is our first iteration of the model. We have an adjusted R^2 value of 0.5252 with 24 predictors.

Finding Best Predictors

```
winning_best_subset <- regsubsets(team_winner~., data = nba_team_box_model,
                                  nbest = 1, nvmax = NULL, force.in = NULL,
                                  force.out = NULL, method = "exhaustive")
winning_summary_best_subset <- summary(winning_best_subset)
as.data.frame(winning_summary_best_subset$outmat)
```

```
which.max(winning_summary_best_subset$adjr2)
```

```
## [1] 22
```

We are now using regsubsets to find the best combination of predictors for our second variation of the model.

Best Predictors for Winning Model

```
winning_summary_best_subset$which[24,]
```

```
##                      (Intercept)                defensive_rebounds
##                             TRUE                              TRUE
##                 field_goals_made                             fouls
##                             TRUE                              TRUE
##             free_throws_attempted                    points_in_paint
##                             TRUE                              TRUE
##                  technical_fouls three_point_field_goals_attempted
##                             TRUE                              TRUE
##                    team_turnovers                           assists
##                             TRUE                              TRUE
##                 fast_break_points             field_goals_attempted
##                             TRUE                              TRUE
##                    free_throw_pct                       largest_lead
##                             TRUE                              TRUE
##                             steals        three_point_field_goal_pct
##                             TRUE                              TRUE
##                    total_rebounds                    turnover_points
##                             TRUE                              TRUE
##                        team_score                            blocks
##                             TRUE                              TRUE
##                    field_goal_pct                     flagrant_fouls
##                             TRUE                              TRUE
##                  free_throws_made                offensive_rebounds
```

```
##                              TRUE                              TRUE
##      three_point_field_goals_made                         turnovers
##                             FALSE                              TRUE
##                   total_turnovers
##                             FALSE
```

This is a binary list of if a column is considered a significant predictor or not. True means put it in the next model and false means don't.

Winning Model with Best Predictors

```
# Create the linear regression model with best predictors
winning_model <- lm(team_winner ~ . -total_turnovers -three_point_field_goals_made, data = nba_team_box_
```

```
# Summary of the model
summary(winning_model)
```

```
##
## Call:
## lm(formula = team_winner ~ . - total_turnovers - three_point_field_goals_made,
##     data = nba_team_box_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56548 -0.26462 -0.03228  0.27281  1.06469
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -0.0157755  0.0728780  -0.216  0.82863
## defensive_rebounds                 0.0167662  0.0006789  24.695  < 2e-16 ***
## field_goals_made                   0.0192123  0.0047663   4.031 5.57e-05 ***
## fouls                             -0.0072372  0.0005012 -14.440  < 2e-16 ***
## free_throws_attempted             -0.0036084  0.0017729  -2.035  0.04183 *
## points_in_paint                    0.0017577  0.0003053   5.757 8.65e-09 ***
## technical_fouls                   -0.0112093  0.0023374  -4.796 1.63e-06 ***
## three_point_field_goals_attempted -0.0007497  0.0008445  -0.888  0.37470
## team_turnovers                     0.0040769  0.0004522   9.015  < 2e-16 ***
## assists                            0.0057254  0.0005158  11.100  < 2e-16 ***
## fast_break_points                 -0.0027473  0.0003391  -8.102 5.61e-16 ***
## field_goals_attempted             -0.0266517  0.0006304 -42.280  < 2e-16 ***
## free_throw_pct                     0.0031523  0.0004774   6.603 4.10e-11 ***
## largest_lead                       0.0173088  0.0003029  57.145  < 2e-16 ***
## steals                             0.0198254  0.0007335  27.030  < 2e-16 ***
## three_point_field_goal_pct         0.0034138  0.0005600   6.096 1.10e-09 ***
## total_rebounds                     0.0091804  0.0005314  17.277  < 2e-16 ***
## turnover_points                   -0.0052778  0.0003949 -13.363  < 2e-16 ***
## team_score                         0.0061439  0.0022355   2.748  0.00599 **
## blocks                            -0.0008271  0.0007122  -1.161  0.24554
## field_goal_pct                    -0.0047070  0.0006611  -7.120 1.10e-12 ***
## flagrant_fouls                    -0.0008583  0.0020150  -0.426  0.67016
## free_throws_made                   0.0060569  0.0030767   1.969  0.04900 *
```

```
## offensive_rebounds                 0.0159117  0.0008490  18.741  < 2e-16 ***
## turnovers                         -0.0181778  0.0007218 -25.186  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3445 on 30391 degrees of freedom
## Multiple R-squared:  0.5256, Adjusted R-squared:  0.5253
## F-statistic:  1403 on 24 and 30391 DF,  p-value: < 2.2e-16
```

```
# Tidy the model output
filtered_tidy_model <- tidy(winning_model)
print(winning_model)
```

```
##
## Call:
## lm(formula = team_winner ~ . - total_turnovers - three_point_field_goals_made,
##     data = nba_team_box_model)
##
## Coefficients:
##                       (Intercept)                  defensive_rebounds
##                         -0.0157755                           0.0167662
##                  field_goals_made                              fouls
##                          0.0192123                         -0.0072372
##             free_throws_attempted                     points_in_paint
##                         -0.0036084                           0.0017577
##                   technical_fouls  three_point_field_goals_attempted
##                         -0.0112093                         -0.0007497
##                    team_turnovers                            assists
##                          0.0040769                           0.0057254
##                 fast_break_points              field_goals_attempted
##                         -0.0027473                         -0.0266517
##                    free_throw_pct                        largest_lead
##                          0.0031523                           0.0173088
##                           steals         three_point_field_goal_pct
##                          0.0198254                           0.0034138
##                    total_rebounds                     turnover_points
##                          0.0091804                         -0.0052778
##                      team_score                                blocks
##                          0.0061439                         -0.0008271
##                    field_goal_pct                      flagrant_fouls
##                         -0.0047070                         -0.0008583
##                  free_throws_made                  offensive_rebounds
##                          0.0060569                           0.0159117
##                       turnovers
##                         -0.0181778
```

```
# Check the model's residuals
filtered_glance_model <- glance(winning_model)
print(winning_model)
```

```
##
## Call:
## lm(formula = team_winner ~ . - total_turnovers - three_point_field_goals_made,
```

```
##     data = nba_team_box_model)
##
## Coefficients:
##                         (Intercept)                  defensive_rebounds
##                           -0.0157755                           0.0167662
##                   field_goals_made                                fouls
##                            0.0192123                          -0.0072372
##              free_throws_attempted                       points_in_paint
##                           -0.0036084                           0.0017577
##                     technical_fouls  three_point_field_goals_attempted
##                           -0.0112093                          -0.0007497
##                      team_turnovers                             assists
##                            0.0040769                           0.0057254
##                  fast_break_points                field_goals_attempted
##                           -0.0027473                          -0.0266517
##                     free_throw_pct                         largest_lead
##                            0.0031523                           0.0173088
##                           steals           three_point_field_goal_pct
##                            0.0198254                           0.0034138
##                     total_rebounds                       turnover_points
##                            0.0091804                          -0.0052778
##                       team_score                                blocks
##                            0.0061439                          -0.0008271
##                     field_goal_pct                      flagrant_fouls
##                           -0.0047070                          -0.0008583
##                   free_throws_made                  offensive_rebounds
##                            0.0060569                           0.0159117
##                        turnovers
##                           -0.0181778
```

As you can tell this model is slightly better because we are now using 22 predictors to capture the same adjusted R^2 value of .5252.

Model Evaluation

```
# Mean Squared Error (MSE)
mse <- mean(resid(winning_model)^2)
cat("Mean Squared Error: ", mse, "\n")
```

```
## Mean Squared Error:  0.1185893
```

```
# Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
cat("Root Mean Squared Error: ", rmse, "\n")
```

```
## Root Mean Squared Error:  0.344368
```

Our model's performance looks pretty decent! With a Mean Squared Error (MSE) of 0.1186 and a Root Mean Squared Error (RMSE) of 0.3444, it seems to be making predictions that are fairly close to the actual values. While it might not be perfect, these numbers suggest that our model is doing a decent job overall.

Hypothesis

H0: There is no difference in effect to winning by each selected variables

H1: One of the selected variables differ from the rest.

Anova

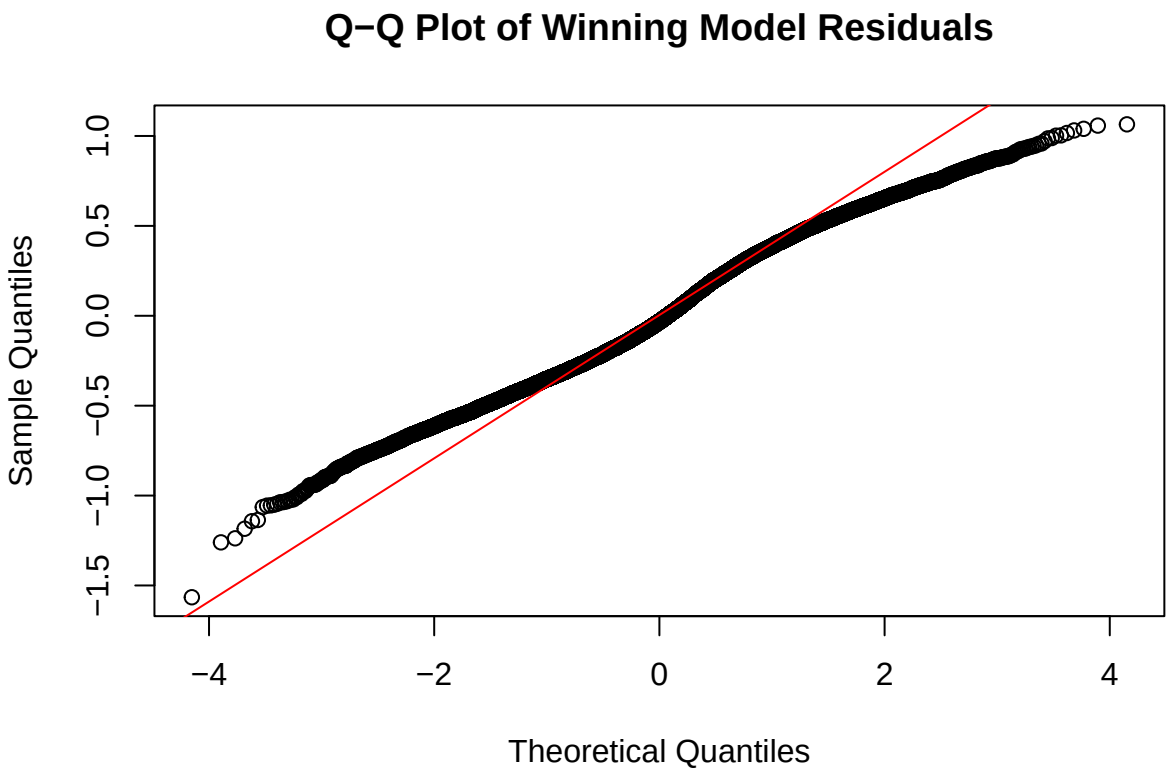
```
anova_result <- anova(winning_model)
anova_result
```

```
## Analysis of Variance Table
##
## Response: team_winner
##                                    Df Sum Sq Mean Sq   F value    Pr(>F)
## defensive_rebounds                  1  964.1  964.12 8123.2022 < 2.2e-16 ***
## field_goals_made                    1  723.8  723.76 6098.0722 < 2.2e-16 ***
## fouls                               1   14.3   14.34  120.8475 < 2.2e-16 ***
## free_throws_attempted               1  255.8  255.79 2155.1338 < 2.2e-16 ***
## points_in_paint                     1    3.8    3.82   32.1754 1.421e-08 ***
## technical_fouls                     1    0.1    0.13    1.1168  0.290610
## three_point_field_goals_attempted   1  127.6  127.57 1074.8836 < 2.2e-16 ***
## team_turnovers                      1   85.0   85.05  716.5849 < 2.2e-16 ***
## assists                             1  199.7  199.69 1682.5340 < 2.2e-16 ***
## fast_break_points                   1    4.4    4.39   37.0072 1.191e-09 ***
## field_goals_attempted               1  324.5  324.47 2733.8407 < 2.2e-16 ***
## free_throw_pct                      1  124.4  124.43 1048.3838 < 2.2e-16 ***
## largest_lead                        1  845.5  845.52 7123.9207 < 2.2e-16 ***
## steals                              1   30.6   30.62  258.0047 < 2.2e-16 ***
## three_point_field_goal_pct          1   28.8   28.77  242.3611 < 2.2e-16 ***
## total_rebounds                      1   70.9   70.87  597.1386 < 2.2e-16 ***
## turnover_points                     1   84.5   84.52  712.1663 < 2.2e-16 ***
## team_score                          1    2.2    2.16   18.2399 1.953e-05 ***
## blocks                              1    0.1    0.11    0.9584  0.327592
## field_goal_pct                      1    2.3    2.26   19.0795 1.258e-05 ***
## flagrant_fouls                      1    0.1    0.08    0.6697  0.413158
## free_throws_made                    1    0.8    0.84    7.0423  0.007965 **
## offensive_rebounds                  1   28.4   28.37  239.0735 < 2.2e-16 ***
## turnovers                           1   75.3   75.28  634.3108 < 2.2e-16 ***
## Residuals                       30391 3607.0    0.12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wanted to run a quick anova test to conduct a test for our hypothesis. We clearly see that avg_FGA, avg_FTA and avg_ASTS were rejected meaning they are different from the rest.

Q-Q Plot - Normality Assumption

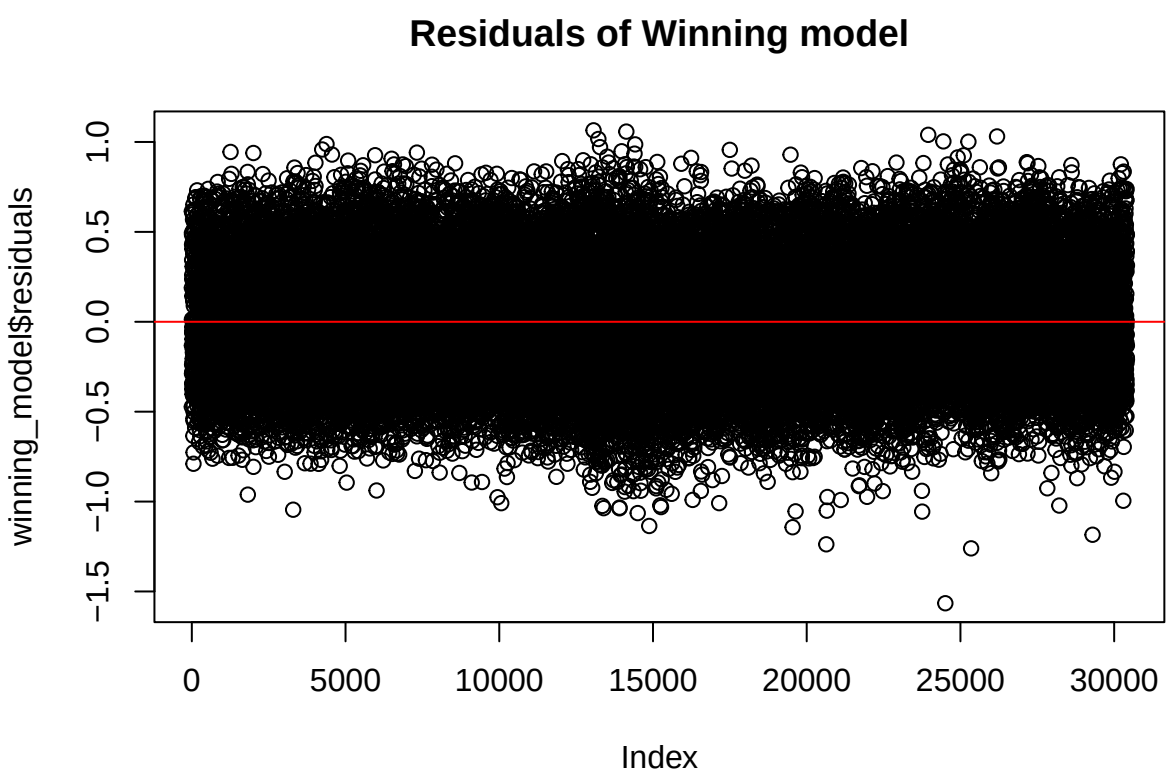
```
qqnorm(winning_model$residuals, main = "Q-Q Plot of Winning Model Residuals")
qqline(winning_model$residuals, col = "red")
```



After running a normality assumption test, we can tell there are very slight deviations from normal.

Residuals Plot for Winning Model

```
# Plot residuals
plot(winning_model$residuals)
abline(h = 0, col = "red")
title('Residuals of Winning model')
```



After plotting the residuals, we can tell our data passes the constant variance assumption.

Durbin Watson Test - Independence

```
dw_test <- durbinWatsonTest(winning_model)
dw_test
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.3868188      2.773548       0
##  Alternative hypothesis: rho != 0
```

Running a quick Durbin Watson test tells us if there is any autocorrelation and according to this test, there is very slight autocorrelation.

Conclusion

After running an anova test, a multiple linear regression model and testing the predictors, the attributes to playing like a winning team are shooting as many shots as possible. This makes sense this two of our variables that differed from the anova test had to do with shooting shots average field goals attempted and average free throws attempted.

Future Plans

After running these linear regression models and understanding the limitations of our data especially our response variable, the route we would take in the future is using classification/clustering rather than regression. This would help us better understand the overall picture and even create a ranking system of predicted wins at the end of a season and the starting playoff brackets.

MVPs - Binary

For our second prediction model, we are trying to predict the MVP of a season. We will start by creating the dataset we will use for the multiple linear regression model. For this goal our target variable is a binary variable which represents if the player is an mvp or not. We will start by giving the model all possible predictors then we will use regsubsets to find the best combination of predictors and see the evolution from the first to the second model. After doing so, we will check the assumptions and see if our model is viable.

Setting Up Data

```
# Define MVPs for each year
mvp_winners <- c(
    "2010" = "LeBron James",
    "2011" = "Derrick Rose",
    "2012" = "LeBron James",
    "2013" = "LeBron James",
    "2014" = "Kevin Durant",
    "2015" = "Stephen Curry",
    "2016" = "Stephen Curry",
    "2017" = "Russell Westbrook",
    "2018" = "James Harden",
    "2019" = "Giannis Antetokounmpo",
    "2020" = "Giannis Antetokounmpo",
    "2021" = "Nikola Jokić",
    "2022" = "Nikola Jokić",
    "2023" = "Giannis Antetokounmpo",
    "2024" = "To be determined"
)

# Create isMVP column based on MVP winners
player_stats_2010_2024$isMVP <- ifelse(player_stats_2010_2024$athlete_display_name == mvp_winners[as.cha

# Print the modified DataFrame
# Print rows where isMVP is 1 using subset()
mvp_players <- subset(player_stats_2010_2024, isMVP == 1)
print(mvp_players)
```

```
## # A tibble: 12 x 21
##    athlete_display_name  season name         short_name avg_PTS avg_FLS avg_STLS
##    <chr>                  <int> <chr>        <chr>        <dbl>   <dbl>    <dbl>
##  1 Derrick Rose            2011 Derrick Rose D. Rose       25.4    1.74    1.10
##  2 Giannis Antetokounmpo   2019 Giannis Ant~ G. Anteto~    27.4    3.20    1.26
##  3 Giannis Antetokounmpo   2020 Giannis Ant~ G. Anteto~    29.1    3.15    0.932
##  4 Giannis Antetokounmpo   2023 Giannis Ant~ G. Anteto~    30.3    3.09    0.791
##  5 James Harden            2018 James Harden J. Harden     29.9    2.4     1.83
##  6 Kevin Durant            2014 Kevin Durant K. Durant     31.6    2.13    1.22
##  7 LeBron James            2010 LeBron James L. James      29.6    1.63    1.66
##  8 LeBron James            2012 LeBron James L. James      28      1.68    1.86
##  9 LeBron James            2013 LeBron James L. James      26.5    1.52    1.71
## 10 Russell Westbrook       2017 Russell Wes~ R. Westbr~    32.0    2.38    1.67
## 11 Stephen Curry           2015 Stephen Cur~ S. Curry      24.6    2.02    1.99
## 12 Stephen Curry           2016 Stephen Cur~ S. Curry      29.1    2.05    2.03
```

```
## # i 14 more variables: avg_REBS <dbl>, avg_BLKS <dbl>, avg_FGM <dbl>,
## #   avg_FGA <dbl>, avg_3PM <dbl>, avg_3PA <dbl>, avg_MINS <dbl>, avg_FTM <dbl>,
## #   avg_FTA <dbl>, avg_OREBS <dbl>, avg_DREBS <dbl>, avg_ASTS <dbl>,
## #   avg_TURN <dbl>, isMVP <dbl>
```

```
player_stats_2010_2024_clean <- player_stats_2010_2024 %>% select(-athlete_display_name, -season, -name
nba_team_box_clean
```

```
## # A tibble: 38,286 x 27
##    team_winner defensive_rebounds field_goals_made fouls free_throws_attempted
##    <lgl>                   <int>            <int> <int>                 <int>
##  1 FALSE                      32               29    25                    17
##  2 TRUE                       30               27    19                    37
##  3 FALSE                      28               28    21                    10
##  4 TRUE                       40               33    17                    19
##  5 FALSE                      18               31    22                    26
##  6 TRUE                       28               40    23                    13
##  7 FALSE                      26               32    23                    22
##  8 TRUE                       25               37    21                    23
##  9 TRUE                       32               34    20                    24
## 10 FALSE                      27               32    27                    24
## # i 38,276 more rows
## # i 22 more variables: points_in_paint <chr>, technical_fouls <int>,
## #   three_point_field_goals_attempted <int>, team_turnovers <int>,
## #   assists <int>, fast_break_points <chr>, field_goals_attempted <int>,
## #   free_throw_pct <dbl>, largest_lead <chr>, steals <int>,
## #   three_point_field_goal_pct <dbl>, total_rebounds <int>,
## #   turnover_points <chr>, team_score <int>, blocks <int>, ...
```

```
names(player_stats_2010_2024_clean)
```

```
##  [1] "avg_PTS"   "avg_FLS"   "avg_STLS"  "avg_REBS"  "avg_BLKS"  "avg_FGM"
##  [7] "avg_FGA"   "avg_3PM"   "avg_3PA"   "avg_MINS"  "avg_FTM"   "avg_FTA"
## [13] "avg_OREBS" "avg_DREBS" "avg_ASTS"  "avg_TURN"  "isMVP"
```

```
player_stats_2010_2024_model <- na.omit(player_stats_2010_2024_clean)
player_stats_2010_2024_model <- player_stats_2010_2024_model %>%
  mutate(across(-all_of("isMVP"), as.double))
head(player_stats_2010_2024_model, 5)
```

```
## # A tibble: 5 x 17
##   avg_PTS avg_FLS avg_STLS avg_REBS avg_BLKS avg_FGM avg_FGA avg_3PM avg_3PA
##     <dbl>   <dbl>    <dbl>    <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1    3.73   0.733    0.133      1.4    0         1.47    2.93   0.667    1.67
## 2    2.9    0.46     0.2        1.06   0.08      1.14    2.56   0.3      1.12
## 3    7.32   0.946    0.625      1.57   0.0536    2.59    6.32   1.07     3.11
## 4    6.64   1.2      0.582      1.44   0.0182    2.31    6.45   0.873    3
## 5    3.78   0.652    0.435      1.37   0.0435    1.30    3.83   0.565    1.91
## # i 8 more variables: avg_MINS <dbl>, avg_FTM <dbl>, avg_FTA <dbl>,
## #   avg_OREBS <dbl>, avg_DREBS <dbl>, avg_ASTS <dbl>, avg_TURN <dbl>,
## #   isMVP <dbl>
```

We have set up the tibble for the prediction model. We have added our target variable in and will now pass it to the first model.

MVP Model with all Variables

```
mvp_all_var_model <- lm(isMVP ~ ., data = player_stats_2010_2024_model)
```

```
summary(mvp_all_var_model)
```

```
##
## Call:
## lm(formula = isMVP ~ ., data = player_stats_2010_2024_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06971 -0.00382 -0.00015  0.00321  0.98267
##
## Coefficients: (2 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0036337  0.0011937   3.044 0.002342 **
## avg_PTS     -0.0062797  0.0021817  -2.878 0.004009 **
## avg_FLS     -0.0021330  0.0010127  -2.106 0.035211 *
## avg_STLS     0.0058356  0.0017939   3.253 0.001146 **
## avg_REBS     0.0021551  0.0005633   3.826 0.000131 ***
## avg_BLKS    -0.0009249  0.0016155  -0.573 0.566994
## avg_FGM      0.0198083  0.0045320   4.371 1.25e-05 ***
## avg_FGA     -0.0036880  0.0008004  -4.608 4.14e-06 ***
## avg_3PM      0.0066907  0.0044437   1.506 0.132199
## avg_3PA      0.0023559  0.0014932   1.578 0.114670
## avg_MINS    -0.0010832  0.0001529  -7.083 1.54e-12 ***
## avg_FTM             NA         NA      NA       NA
## avg_FTA      0.0097301  0.0018071   5.384 7.49e-08 ***
## avg_OREBS   -0.0045559  0.0015117  -3.014 0.002589 **
## avg_DREBS           NA         NA      NA       NA
## avg_ASTS     0.0010278  0.0005404   1.902 0.057227 .
## avg_TURN     0.0022474  0.0015583   1.442 0.149276
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03863 on 7634 degrees of freedom
## Multiple R-squared:  0.04898,    Adjusted R-squared:  0.04723
## F-statistic: 28.08 on 14 and 7634 DF,  p-value: < 2.2e-16
```

This is our first iteration of the model. We have an adjusted R^2 value of 0.04723 with 14 predictors.

Finding Best Predictors

```
mvp_best_subset <- regsubsets(isMVP~., data = player_stats_2010_2024_model, nbest = 1, nvmax = NULL, fo
mvp_summary_best_subset <- summary(mvp_best_subset)
as.data.frame(mvp_summary_best_subset$outmat)
```

```r
which.max(mvp_summary_best_subset$adjr2)
```

```
## [1] 12
```

We are now using regsubsets to find the best combination of predictors for our second variation of the model.

Best Predictors for MVP Model

```r
mvp_summary_best_subset$which[12,]
```

```
## (Intercept)     avg_PTS     avg_FLS    avg_STLS    avg_REBS    avg_BLKS
##        TRUE       FALSE        TRUE        TRUE        TRUE       FALSE
##     avg_FGM     avg_FGA     avg_3PM     avg_3PA    avg_MINS     avg_FTM
##        TRUE        TRUE       FALSE        TRUE        TRUE        TRUE
##     avg_FTA   avg_OREBS   avg_DREBS    avg_ASTS    avg_TURN
##        TRUE       FALSE        TRUE        TRUE        TRUE
```

This is a binary list of if a column is considered a significant predictor or not. True means put it in the next model and false means don't.

MVP Model with Best Predictors

```r
# Create the linear regression model with best predictors
mvp_model <- lm(isMVP ~ . - avg_PTS - avg_BLKS - avg_3PM - avg_OREBS, data = player_stats_2010_2024_mod
```

```r
# Summary of the model
summary(mvp_model)
```

```
##
## Call:
## lm(formula = isMVP ~ . - avg_PTS - avg_BLKS - avg_3PM - avg_OREBS,
##     data = player_stats_2010_2024_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06937 -0.00383 -0.00013  0.00323  0.98219
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0036406  0.0011797   3.086  0.00204 **
## avg_FLS     -0.0022219  0.0010011  -2.219  0.02649 *
## avg_STLS     0.0058109  0.0017920   3.243  0.00119 **
## avg_REBS    -0.0025011  0.0011644  -2.148  0.03175 *
## avg_FGM      0.0072151  0.0013081   5.516 3.59e-08 ***
## avg_FGA     -0.0036707  0.0007035  -5.218 1.86e-07 ***
## avg_3PA      0.0025270  0.0004188   6.035 1.67e-09 ***
## avg_MINS    -0.0010833  0.0001509  -7.178 7.72e-13 ***
## avg_FTM     -0.0061842  0.0021648  -2.857  0.00429 **
```

```
## avg_FTA      0.0096354  0.0017871   5.392 7.19e-08 ***
## avg_DREBS    0.0045791  0.0015110   3.030  0.00245 **
## avg_ASTS     0.0010657  0.0005342   1.995  0.04607 *
## avg_TURN     0.0022324  0.0015573   1.433  0.15176
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03863 on 7636 degrees of freedom
## Multiple R-squared:  0.04893,    Adjusted R-squared:  0.04744
## F-statistic: 32.74 on 12 and 7636 DF,  p-value: < 2.2e-16
```

```
# Tidy the model output
filtered_tidy_model <- tidy(mvp_model)
print(mvp_model)
```

```
##
## Call:
## lm(formula = isMVP ~ . - avg_PTS - avg_BLKS - avg_3PM - avg_OREBS,
##     data = player_stats_2010_2024_model)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS      avg_FGM      avg_FGA
##    0.003641    -0.002222     0.005811    -0.002501     0.007215    -0.003671
##     avg_3PA     avg_MINS      avg_FTM      avg_FTA    avg_DREBS     avg_ASTS
##    0.002527    -0.001083    -0.006184     0.009635     0.004579     0.001066
##    avg_TURN
##    0.002232
```

```
# Check the model's residuals
filtered_glance_model <- glance(mvp_model)
print(mvp_model)
```

```
##
## Call:
## lm(formula = isMVP ~ . - avg_PTS - avg_BLKS - avg_3PM - avg_OREBS,
##     data = player_stats_2010_2024_model)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS      avg_FGM      avg_FGA
##    0.003641    -0.002222     0.005811    -0.002501     0.007215    -0.003671
##     avg_3PA     avg_MINS      avg_FTM      avg_FTA    avg_DREBS     avg_ASTS
##    0.002527    -0.001083    -0.006184     0.009635     0.004579     0.001066
##    avg_TURN
##    0.002232
```

As you can tell this model is slightly better because we are now using 12 predictors to capture the same adjusted R^2 value of 0.04744.

Model Evaluation

```r
# Mean Squared Error (MSE)
mse <- mean(resid(mvp_model)^2)
cat("Mean Squared Error: ", mse, "\n")
```

```
## Mean Squared Error:  0.001489723
```

```r
# Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
cat("Root Mean Squared Error: ", rmse, "\n")
```

```
## Root Mean Squared Error:  0.03859693
```

The linear regression model we built seems to perform quite well. The Mean Squared Error (MSE) is really low at 0.0015, which means the average squared difference between the actual and predicted values is tiny. Similarly, the Root Mean Squared Error (RMSE) is 0.0386, indicating that, on average, our predictions are off by just a little bit. Overall, these stats suggest that our model is pretty accurate in predicting the outcomes based on the given data.

Hypothesis

H0: There is no difference in effect to selecting MVPs by each selected variables

H1: One of the selected variables differ from the rest.

Anova

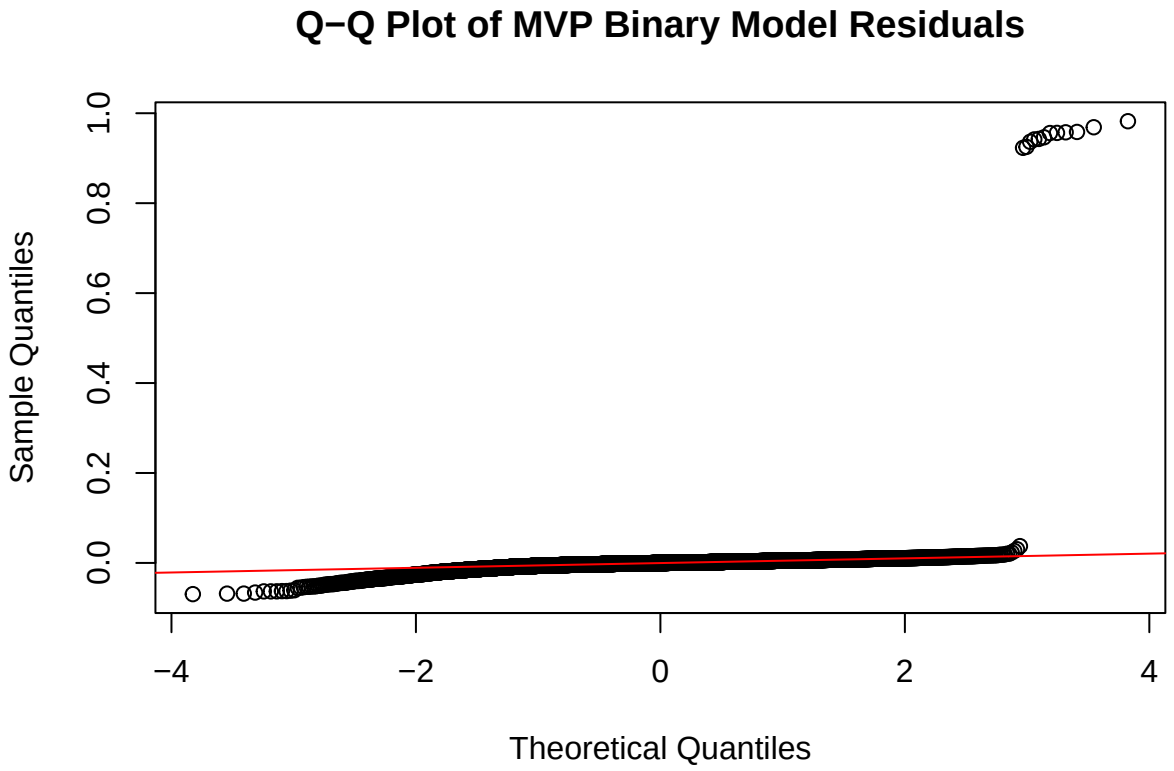
```r
anova_result <- anova(mvp_model)
anova_result
```

```
## Analysis of Variance Table
##
## Response: isMVP
##             Df  Sum Sq  Mean Sq F value    Pr(>F)
## avg_FLS      1  0.0094 0.009429  6.3184 0.0119693 *
## avg_STLS     1  0.0831 0.083051 55.6543 9.592e-14 ***
## avg_REBS     1  0.0666 0.066580 44.6169 2.564e-11 ***
## avg_FGM      1  0.0832 0.083163 55.7298 9.233e-14 ***
## avg_FGA      1  0.0245 0.024509 16.4244 5.113e-05 ***
## avg_3PA      1  0.0092 0.009232  6.1869 0.0128914 *
## avg_MINS     1  0.1002 0.100228 67.1653 2.904e-16 ***
## avg_FTM      1  0.1176 0.117585 78.7965 < 2.2e-16 ***
## avg_FTA      1  0.0446 0.044606 29.8918 4.713e-08 ***
## avg_DREBS    1  0.0236 0.023648 15.8468 6.932e-05 ***
## avg_ASTS     1  0.0212 0.021185 14.1967 0.0001659 ***
## avg_TURN     1  0.0031 0.003066  2.0549 0.1517587
## Residuals 7636 11.3949 0.001492
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wanted to run a quick anova test to conduct a test for our hypothesis. We clearly see that avg_STLS, avg_REBS, avg_FGM, avg_FGA, avg_MINS, avg_FTM, avg_FTA, avg_DREBS, and avg_ASTS were rejected meaning they are different from the rest.

Q-Q Plot - Normality Assumption

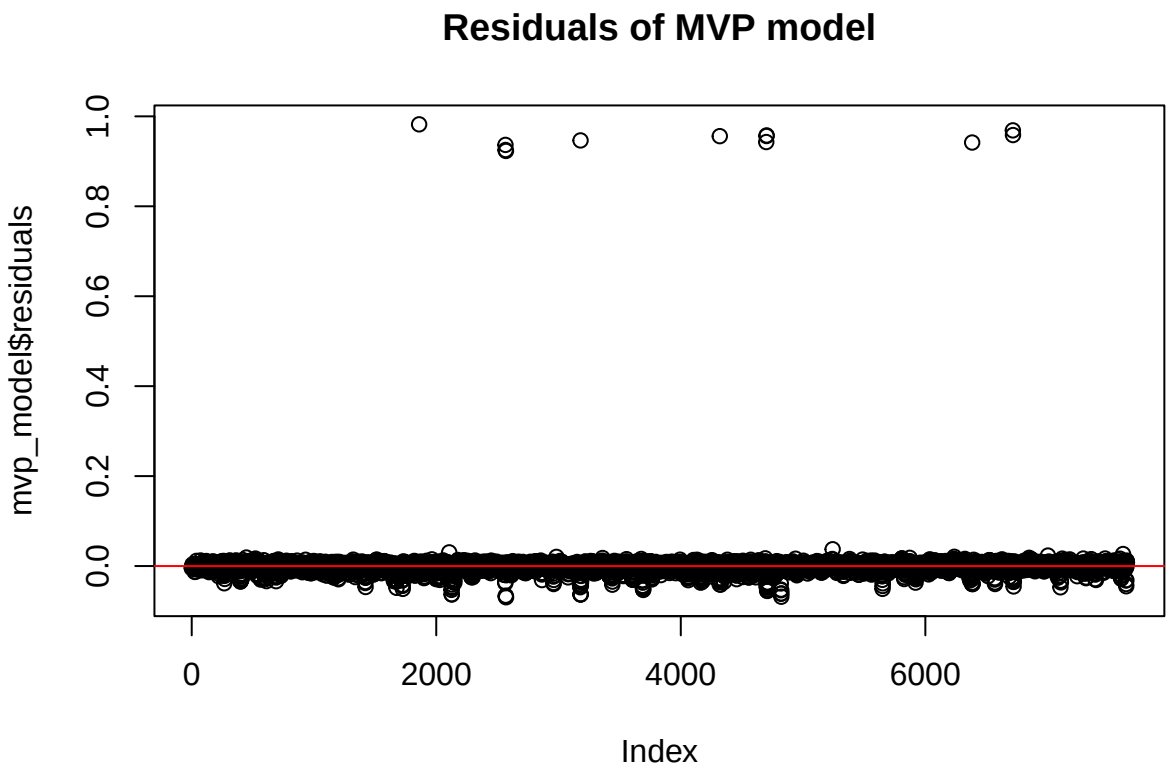
```
qqnorm(mvp_model$residuals, main = "Q-Q Plot of MVP Binary Model Residuals")
qqline(mvp_model$residuals, col = "red")
```



After running a normality assumption test, we can tell there are very large deviations from normal. This is due to the fact that we only have 14 truth values for being an MVP and this is causing the large deviation on the right side of the plot. We have decided to disregard these outliers as they are essential to our model.

Residuals Plot for MVP Model - Constant Variance Assumption

```
# Plot residuals
plot(mvp_model$residuals)
abline(h = 0, col = "red")
title('Residuals of MVP model')
```



After plotting the residuals, we can tell our data has outliers for the constant variance assumption. Once again, there are 14 outliers which directly correlates to the 14 MVP's in the dataset. These MVP's will continue to skew assumptions no matter what so we can disregard them.

Durbin Watson Test - Independence

```
dw_test <- durbinWatsonTest(mvp_model)
dw_test
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2132159      1.573561       0
##  Alternative hypothesis: rho != 0
```

Running a quick Durbin Watson test tells us if there is any autocorrelation and according to this test, there is no autocorrelation according to this test.

Conclusion

After running an anova test, a multiple linear regression model and testing the predictors, the attributes to winning the MVP are playing lots of minutes, shooting as many field goals as possible while maintaining a high field goal percentage and scoring the most points on your team. A limitation we encountered was that creating prediction models with very small number of truth values is extremely hard hence the small R^2 values.

Future Plans

After running these linear regression models and understanding the limitations of our data especially our response variable, the route we would take in the future is by definitely using classification/clustering. We came to a point where we knew linear regression wasn't going to be enough to have the highest quality set of predictors.

MVPs - Votes

For our third prediction model, we are trying to predict the MVP of a season. We will start by creating the dataset we will use for the multiple linear regression model. For this goal our target variable is a integer variable which represents how many votes the player got for MVP of that season. We will start by giving the model all possible predictors then we will use regsubsets to find the best combination of predictors and see the evolution from the first to the second model. After doing so, we will check the assumptions and see if our model is viable.

Setting Up Data

```
#Instead of just using MVP/not MVP, we are using MVP voting from basketball reference's website to get
mvp_votes <- read.csv('~/Downloads/mvp_votes.csv')
head(mvp_votes, 5)
```

```
##            name season votes
## 1  LeBron James   2010  1166
## 2  Kevin Durant   2010   609
## 3   Kobe Bryant   2010   599
## 4 Dwight Howard   2010   478
## 5   Dwyane Wade   2010   119
```

```
#Merging MVP votes data with the player data from before and replacing NANs with 0s
player_stats_2010_2024_merged <- merge(player_stats_2010_2024, mvp_votes, by = c('name', 'season'), all
player_stats_2010_2024_merged$votes[is.na(player_stats_2010_2024_merged$votes)] <- 0
head(player_stats_2010_2024_merged, 5)
```

```
##          name season athlete_display_name  short_name  avg_PTS   avg_FLS
## 1 A.J. Lawson   2023          A.J. Lawson A.J. Lawson 3.733333 0.7333333
## 2 A.J. Lawson   2024          A.J. Lawson A.J. Lawson 2.900000 0.4600000
## 3  A.J. Price   2010           A.J. Price  A.J. Price 7.321429 0.9464286
## 4  A.J. Price   2011           A.J. Price  A.J. Price 6.636364 1.2000000
## 5  A.J. Price   2012           A.J. Price  A.J. Price 3.782609 0.6521739
##    avg_STLS avg_REBS   avg_BLKS  avg_FGM  avg_FGA   avg_3PM  avg_3PA  avg_MINS
## 1 0.1333333 1.400000 0.00000000 1.466667 2.933333 0.6666667 1.666667  7.266667
## 2 0.2000000 1.060000 0.08000000 1.140000 2.560000 0.3000000 1.120000  6.640000
## 3 0.6250000 1.571429 0.05357143 2.589286 6.321429 1.0714286 3.107143 15.410714
## 4 0.5818182 1.436364 0.01818182 2.309091 6.454545 0.8727273 3.000000 15.963636
## 5 0.4347826 1.369565 0.04347826 1.304348 3.826087 0.5652174 1.913043 12.478261
##     avg_FTM   avg_FTA avg_OREBS avg_DREBS  avg_ASTS  avg_TURN isMVP votes
## 1 0.1333333 0.5333333 0.4000000  1.000000 0.1333333 0.2000000     0     0
## 2 0.3200000 0.5000000 0.3000000  0.760000 0.4000000 0.2800000     0     0
## 3 1.0714286 1.3392857 0.2142857  1.357143 1.8928571 1.0535714     0     0
## 4 1.1454545 1.6545455 0.3272727  1.109091 2.1272727 1.0909091     0     0
## 5 0.6086957 0.7608696 0.3043478  1.065217 1.8913043 0.6956522     0     0
```

```
#Getting second dataframe with only MVP vote getters
mvp_votes_only <- merge(player_stats_2010_2024, mvp_votes, by = c('name', 'season'), all.x = F)
head(mvp_votes_only, 5)
```

```
##                name season athlete_display_name    short_name  avg_PTS  avg_FLS
## 1        Al Horford   2016           Al Horford    A. Horford 14.94624 2.021505
## 2      Al Jefferson   2014         Al Jefferson  A. Jefferson 21.78667 2.413333
## 3 Amar'e Stoudemire   2010    Amar'e Stoudemire A. Stoudemire 22.96939 3.418367
## 4    Andre Drummond   2016       Andre Drummond   A. Drummond 16.24419 2.953488
## 5     Anthony Davis   2014        Anthony Davis      A. Davis 20.64706 2.941176
##    avg_STLS  avg_REBS avg_BLKS  avg_FGM  avg_FGA    avg_3PM    avg_3PA avg_MINS
## 1 0.8602151  7.139785 1.559140 6.301075 12.54839 1.07526882 3.06451613 31.94624
## 2 0.9066667 10.813333 1.080000 9.586667 18.85333 0.04000000 0.20000000 35.06667
## 3 0.6428571  8.551020 1.091837 8.448980 15.33673 0.01020408 0.10204082 34.90816
## 4 1.4186047 14.500000 1.372093 6.837209 13.08140 0.02325581 0.09302326 32.70930
## 5 1.3382353  9.911765 2.779412 7.750000 14.86765 0.02941176 0.13235294 34.85294
##    avg_FTM  avg_FTA avg_OREBS avg_DREBS  avg_ASTS avg_TURN isMVP votes
## 1 1.268817 1.559140  1.731183  5.408602 3.1827957 1.333333     0     1
## 2 2.573333 3.706667  2.120000  8.693333 2.0666667 1.626667     0    34
## 3 6.061224 7.897959  2.816327  5.734694 1.0204082 2.581633     0     5
## 4 2.546512 7.209302  4.790698  9.709302 0.7790698 1.860465     0     1
## 5 5.117647 6.470588  3.044118  6.867647 1.5441176 1.617647     0     5
```

We have set up the tibble for the prediction model. We have added our target variable in and will now pass it to the first model.

MVP Model with All Parameters

```
MVP_votes_all_var_model <- lm(votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS + avg_PTS + avg_FGM + avg

# Tidy the model output
filtered_tidy_model <- tidy(MVP_votes_all_var_model)
summary(MVP_votes_all_var_model)
```

```
##
## Call:
## lm(formula = votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGM + avg_FGA + avg_3PM + avg_3PA + avg_MINS +
##     avg_FTA + avg_OREBS + avg_ASTS + avg_TURN, data = player_stats_2010_2024_merged)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -177.00  -10.47   -0.15    9.15 1191.13
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.06363    1.71601   5.282 1.31e-07 ***
## avg_FLS      -6.83218    1.45576  -4.693 2.74e-06 ***
## avg_STLS     20.51520    2.57873   7.956 2.04e-15 ***
## avg_REBS      5.48479    0.80970   6.774 1.35e-11 ***
## avg_BLKS      4.58935    2.32234   1.976 0.048171 *
## avg_PTS      11.86273    3.13634   3.782 0.000157 ***
## avg_FGM      -3.58687    6.51497  -0.551 0.581953
## avg_FGA     -11.36564    1.15062  -9.878  < 2e-16 ***
## avg_3PM     -14.98454    6.38798  -2.346 0.019015 *
## avg_3PA       6.78673    2.14655   3.162 0.001575 **
```

```
## avg_MINS     -2.49699    0.21985 -11.358  < 2e-16 ***
## avg_FTA       6.85823    2.59778   2.640 0.008307 **
## avg_OREBS   -13.76453    2.17313  -6.334 2.52e-10 ***
## avg_ASTS     -0.01153    0.77690  -0.015 0.988158
## avg_TURN      8.60994    2.24006   3.844 0.000122 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.54 on 7634 degrees of freedom
## Multiple R-squared:  0.1759, Adjusted R-squared:  0.1743
## F-statistic: 116.3 on 14 and 7634 DF,  p-value: < 2.2e-16
```

```
# Check the model's residuals
filtered_glance_model <- glance(MVP_votes_all_var_model)
print(MVP_votes_all_var_model)
```

```
##
## Call:
## lm(formula = votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGM + avg_FGA + avg_3PM + avg_3PA + avg_MINS +
##     avg_FTA + avg_OREBS + avg_ASTS + avg_TURN, data = player_stats_2010_2024_merged)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS     avg_BLKS      avg_PTS
##     9.06363     -6.83218     20.51520      5.48479      4.58935     11.86273
##     avg_FGM      avg_FGA      avg_3PM      avg_3PA     avg_MINS      avg_FTA
##    -3.58687    -11.36564    -14.98454      6.78673     -2.49699      6.85823
##   avg_OREBS     avg_ASTS     avg_TURN
##   -13.76453     -0.01153      8.60994
```

This is our first iteration of the model. We have an adjusted R^2 value of 0.1744 with 14 predictors.

Finding Best Predictors

```
mvp_votes_best_subset <- regsubsets(votes~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS + avg_PTS + avg_FGM
mvp_votes_summary_best_subset <- summary(mvp_votes_best_subset)
as.data.frame(mvp_votes_summary_best_subset$outmat)
```

```
which.max(mvp_votes_summary_best_subset$adjr2)
```

```
## [1] 12
```

We are now using regsubsets to find the best combination of predictors for our second variation of the model.

Best Predictors for MVP Model

```
mvp_votes_summary_best_subset$which[12,]
```

```
## (Intercept)     avg_FLS    avg_STLS    avg_REBS    avg_BLKS     avg_PTS
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
##     avg_FGM     avg_FGA     avg_3PM     avg_3PA    avg_MINS     avg_FTA
##       FALSE        TRUE        TRUE        TRUE        TRUE        TRUE
##   avg_OREBS    avg_ASTS    avg_TURN
##        TRUE       FALSE        TRUE
```

This is a binary list of if a column is considered a significant predictor or not. True means put it in the next model and false means don't.

MVP Model with Best Predictors

```
# Create the linear regression model with best predictors
mvp_votes_model <- lm(votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS + avg_PTS + avg_FGA + avg_3PM + a
```

```
# Summary of the model
summary(mvp_votes_model)
```

```
##
## Call:
## lm(formula = votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGA + avg_3PM + avg_3PA + avg_MINS + avg_FTA +
##     avg_OREBS + avg_TURN, data = player_stats_2010_2024_merged)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -178.35  -10.51   -0.16    9.14 1191.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.0339     1.7131   5.273 1.38e-07 ***
## avg_FLS      -6.8255     1.4238  -4.794 1.67e-06 ***
## avg_STLS     20.4409     2.4906   8.207 2.63e-16 ***
## avg_REBS      5.4327     0.8038   6.759 1.50e-11 ***
## avg_BLKS      4.4586     2.2867   1.950   0.0512 .
## avg_PTS      10.2366     1.0296   9.942  < 2e-16 ***
## avg_FGA     -11.4788     1.1155 -10.291  < 2e-16 ***
## avg_3PM     -13.5178     5.7737  -2.341   0.0192 *
## avg_3PA       6.9043     2.1304   3.241   0.0012 **
## avg_MINS     -2.4963     0.2148 -11.624  < 2e-16 ***
## avg_FTA       8.1135     1.2186   6.658 2.97e-11 ***
## avg_OREBS   -13.8962     2.1433  -6.483 9.52e-11 ***
## avg_TURN      8.5558     1.7329   4.937 8.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.53 on 7636 degrees of freedom
## Multiple R-squared:  0.1758, Adjusted R-squared:  0.1745
## F-statistic: 135.7 on 12 and 7636 DF,  p-value: < 2.2e-16
```

```r
# Tidy the model output
filtered_tidy_model <- tidy(mvp_votes_model)
print(mvp_votes_model)
```

```
##
## Call:
## lm(formula = votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGA + avg_3PM + avg_3PA + avg_MINS + avg_FTA +
##     avg_OREBS + avg_TURN, data = player_stats_2010_2024_merged)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS     avg_BLKS      avg_PTS
##       9.034       -6.826       20.441        5.433        4.459       10.237
##     avg_FGA      avg_3PM      avg_3PA     avg_MINS      avg_FTA    avg_OREBS
##     -11.479      -13.518        6.904       -2.496        8.113      -13.896
##    avg_TURN
##       8.556
```

```r
# Check the model's residuals
filtered_glance_model <- glance(mvp_votes_model)
print(mvp_votes_model)
```

```
##
## Call:
## lm(formula = votes ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGA + avg_3PM + avg_3PA + avg_MINS + avg_FTA +
##     avg_OREBS + avg_TURN, data = player_stats_2010_2024_merged)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS     avg_BLKS      avg_PTS
##       9.034       -6.826       20.441        5.433        4.459       10.237
##     avg_FGA      avg_3PM      avg_3PA     avg_MINS      avg_FTA    avg_OREBS
##     -11.479      -13.518        6.904       -2.496        8.113      -13.896
##    avg_TURN
##       8.556
```

As you can tell this model is slightly better because we are now using 12 predictors to capture the same adjusted R^2 value of 0.1745.

Model Evaluation

```r
# Mean Squared Error (MSE)
mse <- mean(resid(mvp_votes_model)^2)
cat("Mean Squared Error: ", mse, "\n")
```

```
## Mean Squared Error:  3078.515
```

```r
# Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
cat("Root Mean Squared Error: ", rmse, "\n")
```

```
## Root Mean Squared Error:  55.48437
```

The linear regression model was evaluated using two key metrics: Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The MSE for the model is 3078.515, which gives us an idea of the average squared difference between the actual and predicted values. The RMSE, which is easier to interpret, is 55.48437. This means that, on average, the model's predictions are off by about 55.5 units from the actual values. Overall, these metrics provide a snapshot of the model's prediction accuracy and help us understand how well it's performing.

Hypothesis

H0: There is no difference in effect to selecting MVPs by each selected variables

H1: One of the selected variables differ from the rest.

Anova

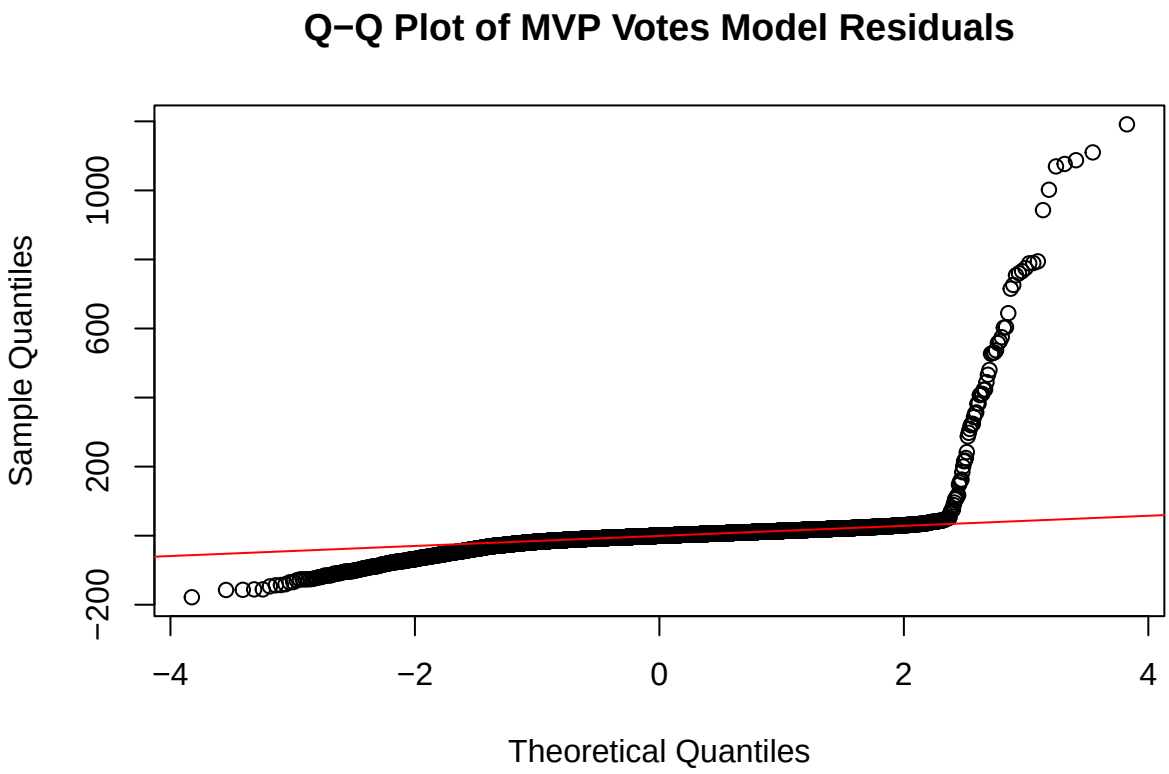
```
anova_result <- anova(mvp_votes_model)
anova_result
```

```
## Analysis of Variance Table
##
## Response: votes
##             Df   Sum Sq Mean Sq  F value    Pr(>F)
## avg_FLS      1   151964  151964  49.2788 2.411e-12 ***
## avg_STLS     1   746216  746216 241.9829 < 2.2e-16 ***
## avg_REBS     1   532063  532063 172.5375 < 2.2e-16 ***
## avg_BLKS     1     5331    5331   1.7289    0.1886
## avg_PTS      1  1173380 1173380 380.5034 < 2.2e-16 ***
## avg_FGA      1  1219577 1219577 395.4842 < 2.2e-16 ***
## avg_3PM      1    71097   71097  23.0554 1.604e-06 ***
## avg_3PA      1   244321  244321  79.2284 < 2.2e-16 ***
## avg_MINS     1   409871  409871 132.9128 < 2.2e-16 ***
## avg_FTA      1   224480  224480  72.7945 < 2.2e-16 ***
## avg_OREBS    1   169785  169785  55.0577 1.297e-13 ***
## avg_TURN     1    75169   75169  24.3758 8.094e-07 ***
## Residuals 7636 23547560    3084
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wanted to run a quick anova test to conduct a test for our hypothesis. We clearly see that avg_FLS, avg_STLS, avg_REBS, avg_FGM, avg_FGA, avg_MINS, avg_FTM, avg_FTA, avg_DREBS, and avg_ASTS were rejected meaning they are different from the rest.

Q-Q Plot - Normality Assumption

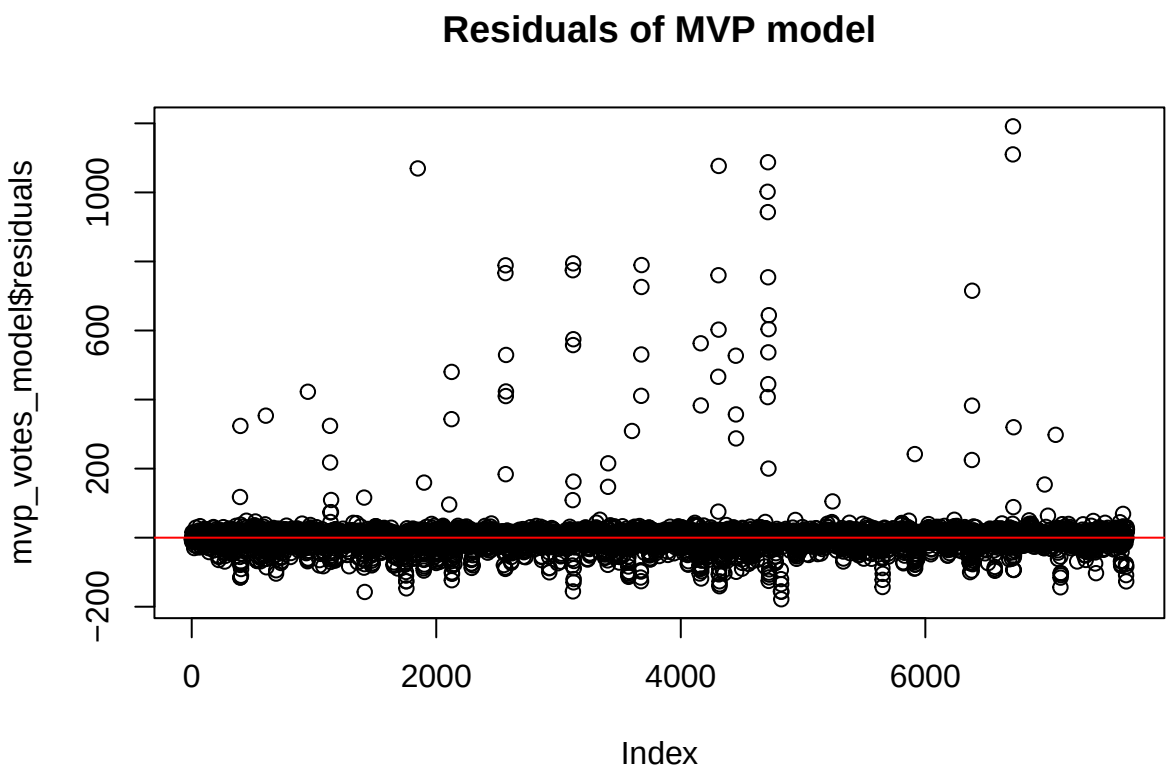
```
qqnorm(mvp_votes_model$residuals, main = "Q-Q Plot of MVP Votes Model Residuals")
qqline(mvp_votes_model$residuals, col = "red")
```



After running a normality assumption test, we can tell there are very large deviations from normal. This is due to the fact that we only have 70 truth values for MVP votes and this is causing the large deviation on the right side of the plot. We have decided to disregard these outliers as they are essential to our model.

Residuals Plot for MVP Model

```
# Plot residuals
plot(mvp_votes_model$residuals)
abline(h = 0, col = "red")
title('Residuals of MVP model')
```



After plotting the residuals, we can tell our data has outliers for the constant variance assumption. Once again, there are 70 outliers which directly correlates to the 70 MVP votes in the dataset. These MVP's will continue to skew assumptions no matter what so we can disregard them.

Durbin Watson Test - Independence

```
dw_test <- durbinWatsonTest(mvp_votes_model)
dw_test
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.4972392      1.005496       0
##  Alternative hypothesis: rho != 0
```

Running a quick Durbin Watson test tells us if there is any autocorrelation and according to this test, there is slight autocorrelation according to this test.

Analysis

When we decided on the idea of projecting NBA stats, we wanted to find some examples of underrated seasons. We thought that the best way to show a player was 'underrated' or under appreciated was to compare the amount of real life votes they received and their projected vote total.

```r
mvp_testing <- predict(mvp_votes_model, mvp_votes_only)
under_appreciated <- order(mvp_testing, decreasing = TRUE)[1:5]

print("Top 5 most underappreciated mvp-vote receiving seasons")
```

```
## [1] "Top 5 most underappreciated mvp-vote receiving seasons"
```

```r
num <- 1
for (i in under_appreciated) {
  print(paste0("(", num, ") ", mvp_votes_only$name[i], " ", mvp_votes_only$season[i], " - Projected ", i
  num <- num + 1
}
```

```
## [1] "(1) James Harden 2020 - Projected 205 more votes"
## [1] "(2) James Harden 2019 - Projected 201 more votes"
## [1] "(3) Joel Embiid 2024 - Projected 196 more votes"
## [1] "(4) James Harden 2017 - Projected 195 more votes"
## [1] "(5) Joel Embiid 2023 - Projected 189 more votes"
```

```r
all_testing <- predict(mvp_votes_model, player_stats_2010_2024_merged)
player_stats_2010_2024_merged$proj_votes <- player_stats_2010_2024_merged$votes
for (i in 1:nrow(player_stats_2010_2024_merged)) {
  player_stats_2010_2024_merged$proj_votes[i] <- player_stats_2010_2024_merged$proj_votes[i] + round(al
  if(isTRUE(player_stats_2010_2024_merged$proj_votes[i] < 0) == TRUE) {
    player_stats_2010_2024_merged$proj_votes[i] <- 0
  }
}
player_stats_2010_2024_merged <- na.omit(player_stats_2010_2024_merged)
head(player_stats_2010_2024_merged, 5)
```

```
##          name season athlete_display_name  short_name  avg_PTS   avg_FLS
## 1 A.J. Lawson   2023          A.J. Lawson A.J. Lawson 3.733333 0.7333333
## 2 A.J. Lawson   2024          A.J. Lawson A.J. Lawson 2.900000 0.4600000
## 3  A.J. Price   2010           A.J. Price  A.J. Price 7.321429 0.9464286
## 4  A.J. Price   2011           A.J. Price  A.J. Price 6.636364 1.2000000
## 5  A.J. Price   2012           A.J. Price  A.J. Price 3.782609 0.6521739
##    avg_STLS avg_REBS   avg_BLKS  avg_FGM  avg_FGA   avg_3PM  avg_3PA  avg_MINS
## 1 0.1333333 1.400000 0.00000000 1.466667 2.933333 0.6666667 1.666667  7.266667
## 2 0.2000000 1.060000 0.08000000 1.140000 2.560000 0.3000000 1.120000  6.640000
## 3 0.6250000 1.571429 0.05357143 2.589286 6.321429 1.0714286 3.107143 15.410714
## 4 0.5818182 1.436364 0.01818182 2.309091 6.454545 0.8727273 3.000000 15.963636
## 5 0.4347826 1.369565 0.04347826 1.304348 3.826087 0.5652174 1.913043 12.478261
##     avg_FTM   avg_FTA avg_OREBS avg_DREBS  avg_ASTS  avg_TURN isMVP votes
## 1 0.1333333 0.5333333 0.4000000  1.000000 0.1333333 0.2000000     0     0
## 2 0.3200000 0.5000000 0.3000000  0.760000 0.4000000 0.2800000     0     0
## 3 1.0714286 1.3392857 0.2142857  1.357143 1.8928571 1.0535714     0     0
## 4 1.1454545 1.6545455 0.3272727  1.109091 2.1272727 1.0909091     0     0
## 5 0.6086957 0.7608696 0.3043478  1.065217 1.8913043 0.6956522     0     0
##   proj_votes
## 1          4
## 2          6
## 3         12
```

```
## 4          2
## 5          0
```

Now that we are finished with our MVP projection model, the next step is to run it and see how it does compared to real life data.

```
print('Projected MVP winners by year')
```

```
## [1] "Projected MVP winners by year"
```

```
for(i in 2010:2024) {
  print(paste0(player_stats_2010_2024_merged %>% filter(season == i) %>% filter(proj_votes == max(proj_v
}
```

```
## [1] "2010: LeBron James - 1330 projected votes"
## [1] "2011: Derrick Rose - 1194 projected votes"
## [1] "2012: LeBron James - 1205 projected votes"
## [1] "2013: LeBron James - 1327 projected votes"
## [1] "2014: Kevin Durant - 1387 projected votes"
## [1] "2015: Stephen Curry - 1286 projected votes"
## [1] "2016: Stephen Curry - 1429 projected votes"
## [1] "2017: Russell Westbrook - 1061 projected votes"
## [1] "2018: James Harden - 1135 projected votes"
## [1] "2019: Giannis Antetokounmpo - 1116 projected votes"
## [1] "2020: Giannis Antetokounmpo - 1135 projected votes"
## [1] "2021: Joel Embiid - 761 projected votes"
## [1] "2022: Joel Embiid - 881 projected votes"
## [1] "2023: Joel Embiid - 1104 projected votes"
## [1] "2024: Joel Embiid - 1182 projected votes"
```

```
print('Actual MVP winners by year')
```

```
## [1] "Actual MVP winners by year"
```

```
for (i in 2010:2024) {
  print(paste0(player_stats_2010_2024_merged %>% filter(season == i) %>% filter(votes == max(votes)) %>%
}
```

```
## [1] "2010: LeBron James - 1166 votes"
## [1] "2011: Derrick Rose - 1132 votes"
## [1] "2012: LeBron James - 1074 votes"
## [1] "2013: LeBron James - 1207 votes"
## [1] "2014: Kevin Durant - 1232 votes"
## [1] "2015: Stephen Curry - 1198 votes"
## [1] "2016: Stephen Curry - 1310 votes"
## [1] "2017: Russell Westbrook - 888 votes"
## [1] "2018: James Harden - 965 votes"
## [1] "2019: Giannis Antetokounmpo - 941 votes"
## [1] "2020: Giannis Antetokounmpo - 962 votes"
## [1] "2021: Joel Embiid - 586 votes"
## [1] "2022: Joel Embiid - 706 votes"
## [1] "2023: Joel Embiid - 915 votes"
## [1] "2024: Joel Embiid - 986 votes"
```

There are a few things of note we can see here. First off, the only season where the projected MVP did not actually win the MVP award was this season, 2024. Our algorithm projected Joel Embiid to win the award, but Nikola Jokic won in real life. This is because Joel Embiid did not play enough games to qualify for the award, but he was on pace to win before getting injured.

Conclusion

After running an anova test, a multiple linear regression model and testing the predictors, the attributes to winning the MVP are playing lots of minutes, shooting as many field goals as possible while maintaining a high field goal percentage and scoring the most points on your team. A limitation we encountered was that creating prediction models with very small number of truth values is extremely hard hence the small Rˆ2 values.

Future Plans

After running these linear regression models and understanding the limitations of our data especially our response variable, the route we would take in the future is using classification/clustering.

Championships

For our fourth prediction model, we are trying to predict the champion of a season. We will start by creating the dataset we will use for the multiple linear regression model. For this goal our target variable is a binary variable which represents if the team is the champion of that season or not. We will start by giving the model all possible predictors then we will use regsubsets to find the best combination of predictors and see the evolution from the first to the second model. After doing so, we will check the assumptions and see if our model is viable.

Setting Up Data

```
cleaned_team_stats_2010_2024$is_champion <- 0

# Manually set `is_champion` to 1 for each championship-winning team and season
cleaned_team_stats_2010_2024 <- cleaned_team_stats_2010_2024 %>%
  mutate(is_champion = ifelse(grepl("Lakers", team_name) & season == 2010, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Mavericks", team_name) & season == 2011, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Heat", team_name) & season == 2012, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Heat", team_name) & season == 2013, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Spurs", team_name) & season == 2014, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Warriors", team_name) & season == 2015, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Cavaliers", team_name) & season == 2016, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Warriors", team_name) & season == 2017, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Warriors", team_name) & season == 2018, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Raptors", team_name) & season == 2019, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Lakers", team_name) & season == 2020, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Bucks", team_name) & season == 2021, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Warriors", team_name) & season == 2022, 1, is_champion)) %>%
  mutate(is_champion = ifelse(grepl("Nuggets", team_name) & season == 2023, 1, is_champion))


cleaned_team_stats_2010_2024 <- cleaned_team_stats_2010_2024 %>%
  arrange(season)

cleaned_team_stats_2010_2024 <- cleaned_team_stats_2010_2024 %>%
  select(-avg_FBPTS, -avg_LEAD, -avg_PIP)

head(cleaned_team_stats_2010_2024, 5)
```

```
## # A tibble: 5 x 22
##   team_name season name      abr   avg_PTS avg_ALLPTS avg_FLS avg_STLS avg_BLKS
##   <chr>      <int> <chr>     <chr>   <dbl>      <dbl>   <dbl>    <dbl>    <dbl>
## 1 76ers       2010 76ers     PHI      97.7      102.     20.5     8.13     5.35
## 2 Bobcats     2010 Bobcats   CHA      94.8       93.9    19.8     7.60     5.40
## 3 Bucks       2010 Bucks     MIL      97.1       95.8    22.1     7.01     4.58
## 4 Bulls       2010 Bulls     CHI      97.4       99.5    20.2     6.44     5.67
## 5 Cavaliers   2010 Cavaliers CLE     102.        96.0    19.7     6.78     5.35
## # i 13 more variables: avg_FGM <dbl>, avg_FGA <dbl>, avg_3PM <dbl>,
## #   avg_3PA <dbl>, avg_MINS <dbl>, avg_FTM <dbl>, avg_FTA <dbl>,
## #   avg_OREBS <dbl>, avg_DREBS <dbl>, avg_REBS <dbl>, avg_ASTS <dbl>,
## #   avg_TURN <dbl>, is_champion <dbl>
```

We have set up the tibble for the prediction model. We have added our target variable in and will now pass it to the first model.

Champion Model with All Variables

```
champion_all_var_model <- lm(is_champion ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS + avg_PTS + avg_FGM
```

```
# Summary of the model
summary(champion_all_var_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGM + avg_FGA + avg_3PM + avg_3PA + avg_FTA +
##     avg_ASTS + avg_TURN, data = cleaned_team_stats_2010_2024)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17611 -0.05859 -0.02571  0.00879  0.96382
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.436171   0.231905   1.881 0.060638 .
## avg_FLS      0.006605   0.004547   1.453 0.146993
## avg_STLS     0.023446   0.008071   2.905 0.003852 **
## avg_REBS     0.018145   0.004714   3.849 0.000135 ***
## avg_BLKS     0.001563   0.006138   0.255 0.799104
## avg_PTS      0.003329   0.011394   0.292 0.770273
## avg_FGM      0.010595   0.023658   0.448 0.654487
## avg_FGA     -0.025126   0.004833  -5.199 3.04e-07 ***
## avg_3PM     -0.007332   0.018387  -0.399 0.690266
## avg_3PA      0.003361   0.005453   0.616 0.537963
## avg_FTA     -0.009425   0.009415  -1.001 0.317328
## avg_ASTS     0.006900   0.003590   1.922 0.055189 .
## avg_TURN    -0.006872   0.007044  -0.976 0.329825
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1659 on 453 degrees of freedom
## Multiple R-squared:  0.08222,    Adjusted R-squared:  0.0579
## F-statistic: 3.382 on 12 and 453 DF,  p-value: 9.428e-05
```

```
# Tidy the model output
filtered_tidy_model <- tidy(champion_all_var_model)
print(champion_all_var_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGM + avg_FGA + avg_3PM + avg_3PA + avg_FTA +
##     avg_ASTS + avg_TURN, data = cleaned_team_stats_2010_2024)
```

```
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS     avg_BLKS      avg_PTS
##    0.436171     0.006605     0.023446     0.018145     0.001563     0.003329
##     avg_FGM      avg_FGA      avg_3PM      avg_3PA      avg_FTA     avg_ASTS
##    0.010595    -0.025126    -0.007332     0.003361    -0.009425     0.006900
##    avg_TURN
##   -0.006872
```

```
# Check the model's residuals
filtered_glance_model <- glance(champion_all_var_model)
print(champion_all_var_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS +
##     avg_PTS + avg_FGM + avg_FGA + avg_3PM + avg_3PA + avg_FTA +
##     avg_ASTS + avg_TURN, data = cleaned_team_stats_2010_2024)
##
## Coefficients:
## (Intercept)      avg_FLS     avg_STLS     avg_REBS     avg_BLKS      avg_PTS
##    0.436171     0.006605     0.023446     0.018145     0.001563     0.003329
##     avg_FGM      avg_FGA      avg_3PM      avg_3PA      avg_FTA     avg_ASTS
##    0.010595    -0.025126    -0.007332     0.003361    -0.009425     0.006900
##    avg_TURN
##   -0.006872
```

This is our first iteration of the model. We have an adjusted R^2 value of 0.05791 with 12 predictors.

Finding Best Predictors

```
champion_best_subset <- regsubsets(is_champion ~ avg_FLS + avg_STLS + avg_REBS + avg_BLKS + avg_PTS + a
champion_summary_best_subset <- summary(champion_best_subset)
as.data.frame(champion_summary_best_subset$outmat)
```

```
which.max(champion_summary_best_subset$adjr2)
```

```
## [1] 6
```

We are now using regsubsets to find the best combination of predictors for our second variation of the model.

Best Predictors for Champion Model

```
champion_summary_best_subset$which[6,]
```

```
## (Intercept)     avg_FLS    avg_STLS    avg_REBS    avg_BLKS     avg_PTS
##        TRUE       FALSE        TRUE        TRUE       FALSE        TRUE
##     avg_FGM     avg_FGA     avg_3PM     avg_3PA     avg_FTA    avg_ASTS
```

```
##       FALSE        TRUE       FALSE       FALSE        TRUE        TRUE
##    avg_TURN
##       FALSE
```

This is a binary list of if a column is considered a significant predictor or not. True means put it in the next model and false means don't.

Champion Model with Best Predictors

```
# Create the linear regression model with best predictors
champion_model <- lm(is_champion ~ avg_STLS + avg_REBS + avg_PTS + avg_FGA + avg_FTA + avg_ASTS, data =
```

```
# Summary of the model
summary(champion_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_STLS + avg_REBS + avg_PTS + avg_FGA +
##     avg_FTA + avg_ASTS, data = cleaned_team_stats_2010_2024)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17892 -0.05621 -0.02326  0.00628  0.97248
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.474106   0.183182   2.588 0.009955 **
## avg_STLS     0.021642   0.006579   3.289 0.001081 **
## avg_REBS     0.016248   0.004524   3.591 0.000364 ***
## avg_PTS      0.006242   0.001749   3.569 0.000395 ***
## avg_FGA     -0.022428   0.003782  -5.929 5.99e-09 ***
## avg_FTA     -0.009205   0.003475  -2.649 0.008349 **
## avg_ASTS     0.006736   0.002205   3.054 0.002388 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1653 on 459 degrees of freedom
## Multiple R-squared:  0.07671,    Adjusted R-squared:  0.06464
## F-statistic: 6.355 on 6 and 459 DF,  p-value: 1.935e-06
```

```
# Tidy the model output
filtered_tidy_model <- tidy(champion_model)
print(champion_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_STLS + avg_REBS + avg_PTS + avg_FGA +
##     avg_FTA + avg_ASTS, data = cleaned_team_stats_2010_2024)
##
## Coefficients:
## (Intercept)     avg_STLS     avg_REBS      avg_PTS      avg_FGA      avg_FTA
```

```
##    0.474106     0.021642     0.016248     0.006242    -0.022428    -0.009205
##    avg_ASTS
##    0.006736
```

```r
# Check the model's residuals
filtered_glance_model <- glance(champion_model)
print(champion_model)
```

```
##
## Call:
## lm(formula = is_champion ~ avg_STLS + avg_REBS + avg_PTS + avg_FGA +
##     avg_FTA + avg_ASTS, data = cleaned_team_stats_2010_2024)
##
## Coefficients:
## (Intercept)     avg_STLS     avg_REBS      avg_PTS      avg_FGA      avg_FTA
##    0.474106     0.021642     0.016248     0.006242    -0.022428    -0.009205
##    avg_ASTS
##    0.006736
```

As you can tell this model is a lot better because we are now using 6 predictors to capture the same adjusted R^2 value of 0.06464.

Model Evaluation

```r
# Mean Squared Error (MSE)
mse <- mean(resid(champion_model)^2)
cat("Mean Squared Error: ", mse, "\n")
```

```
## Mean Squared Error:  0.02690511
```

```r
# Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
cat("Root Mean Squared Error: ", rmse, "\n")
```

```
## Root Mean Squared Error:  0.1640278
```

The linear regression model we built seems to perform quite well. The Mean Squared Error (MSE) is really low at 0.0269, which means the average squared difference between the actual and predicted values is tiny. Similarly, the Root Mean Squared Error (RMSE) is 0.164, indicating that, on average, our predictions are off by just a little bit. Overall, these stats suggest that our model is pretty accurate in predicting the outcomes based on the given data.

Hypothesis

H0: There is no difference in effect to selecting MVPs by each selected variables

H1: One of the selected variables differ from the rest.

Anova

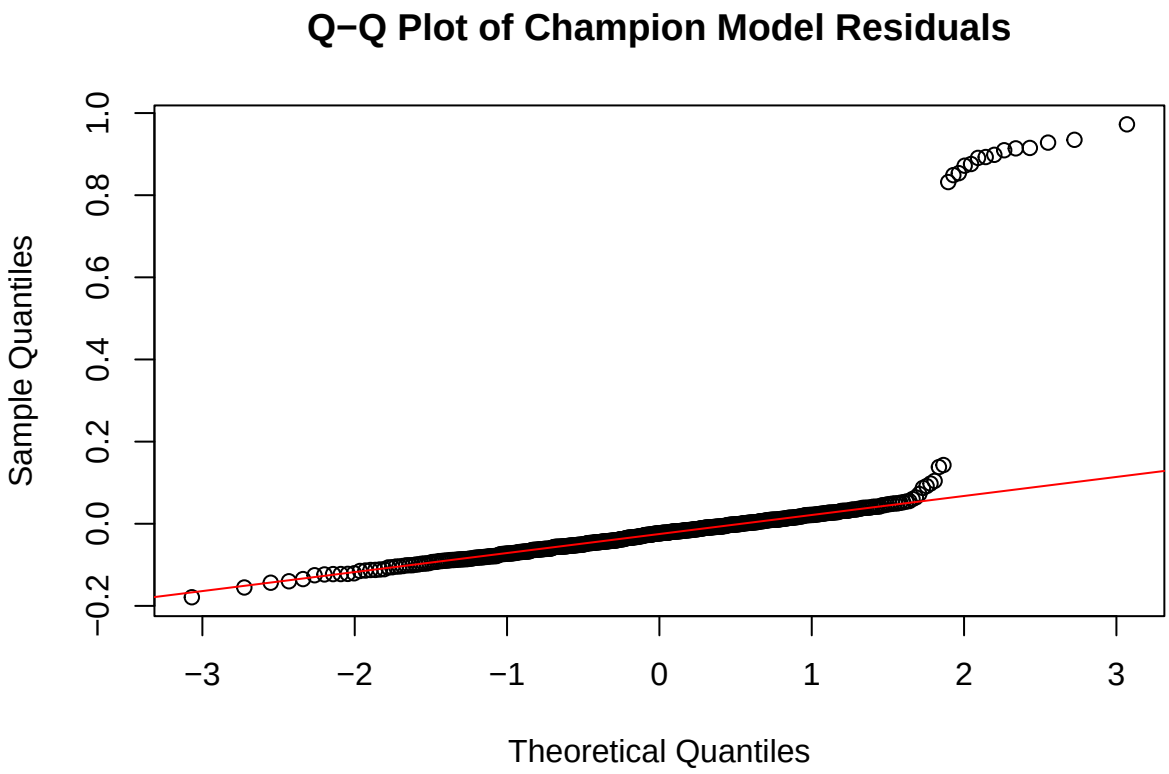
```r
anova_result <- anova(champion_model)
anova_result
```

```
## Analysis of Variance Table
##
## Response: is_champion
##            Df  Sum Sq Mean Sq F value   Pr(>F)
## avg_STLS    1  0.0145 0.01446  0.5293 0.467274
## avg_REBS    1  0.0002 0.00023  0.0085 0.926431
## avg_PTS     1  0.0008 0.00082  0.0300 0.862512
## avg_FGA     1  0.5453 0.54528 19.9622 9.96e-06 ***
## avg_FTA     1  0.2260 0.22603  8.2748 0.004207 **
## avg_ASTS    1  0.2548 0.25480  9.3280 0.002388 **
## Residuals 459 12.5378 0.02732
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wanted to run a quick anova test to conduct a test for our hypothesis. We clearly see that avg_FGA, avg_FTA, and avg_ASTS were rejected meaning they are different from the rest.

Q-Q Plot - Normality Assumption

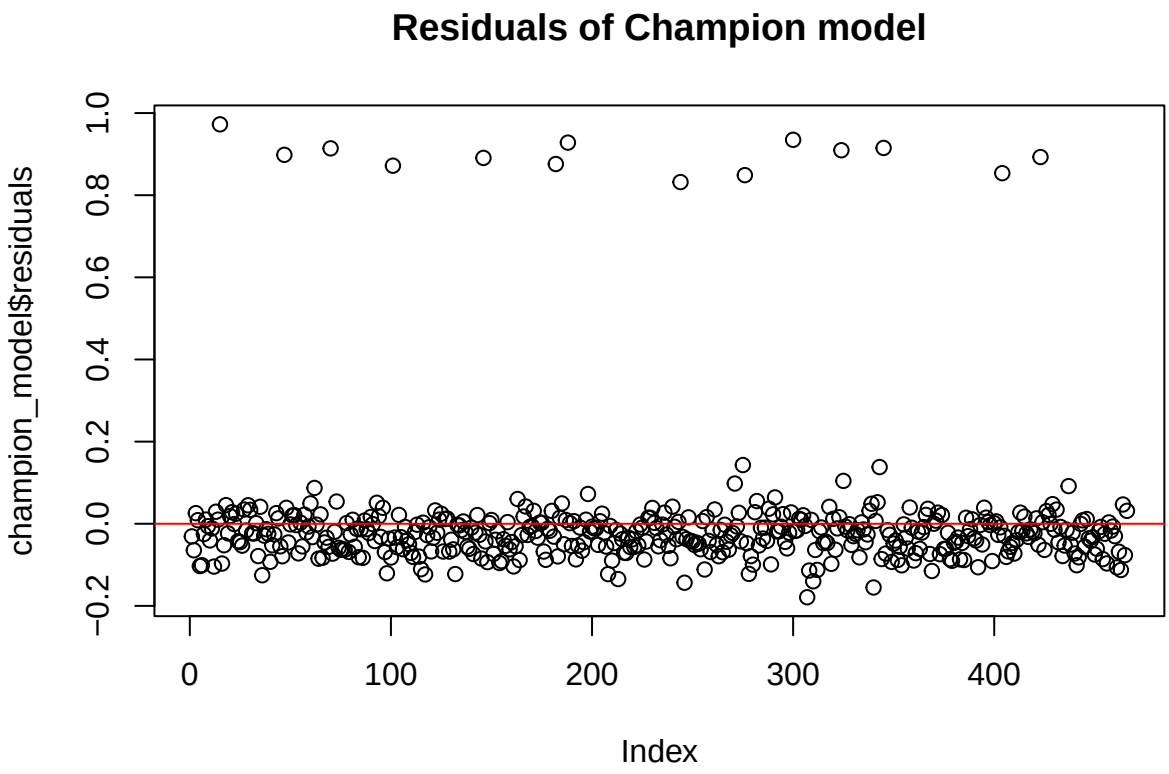
```r
qqnorm(champion_model$residuals, main = "Q-Q Plot of Champion Model Residuals")
qqline(champion_model$residuals, col = "red")
```



After running a normality assumption test, we can tell there are very large deviations from normal. This is due to the fact that we only have 14 truth values for being a champion and this is causing the large deviation on the right side of the plot. We have decided to disregard these outliers as they are essential to our model.

Residuals Plot for Champion Model

```
# Plot residuals
plot(champion_model$residuals)
abline(h = 0, col = "red")
title('Residuals of Champion model')
```



After plotting the residuals, we can tell our data has outliers for the constant variance assumption. Once again, there are 14 outliers which directly correlates to the 14 champions's in the dataset. These MVP's will continue to skew assumptions no matter what so we can disregard them.

Durbin Watson Test - Independence

```
dw_test <- durbinWatsonTest(champion_model)
dw_test
```

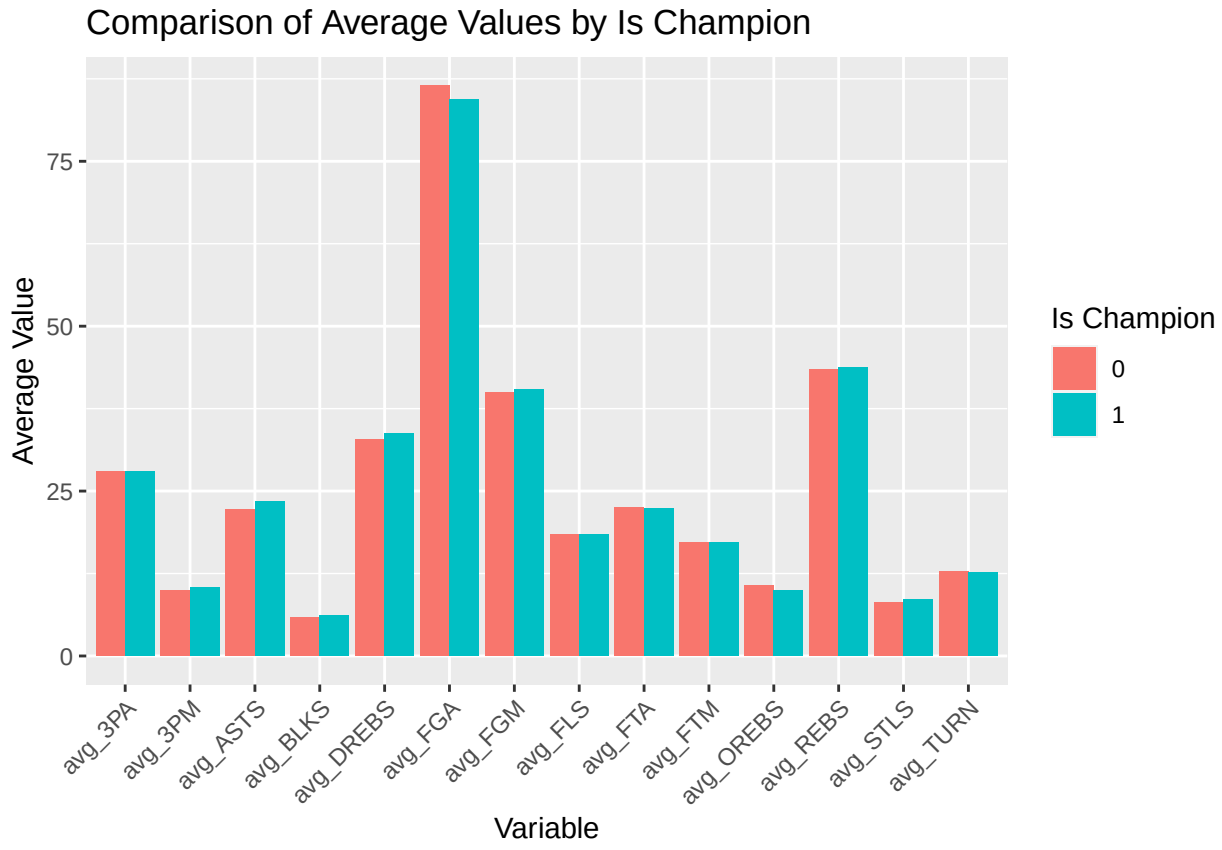
```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01736775      2.034581   0.994
##  Alternative hypothesis: rho != 0
```

Running a quick Durbin Watson test tells us if there is any autocorrelation and according to this test, there is no autocorrelation according to this test.

Analysis

```
summary_data <- cleaned_team_stats_2010_2024 %>%
  select(-avg_MINS) %>%
  group_by(is_champion) %>%
  summarise(across(avg_FLS:avg_TURN, mean))
```

```r
# Reshape the summary data frame from wide to long format
summary_data_long <- summary_data %>%
  pivot_longer(cols = avg_FLS:avg_TURN, names_to = "variable", values_to = "value")

# Create the bar plot
plot <- ggplot(summary_data_long, aes(x = variable, y = value, fill = factor(is_champion))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Variable", y = "Average Value", fill = "Is Champion") +
  ggtitle("Comparison of Average Values by Is Champion") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(plot)
```



Championship teams generally have higher averages in critical areas such as assists (avg_ASTS), field goals made (avg_FGM), free throws made (avg_FTM), and total rebounds (avg_REBS), strong rebounding are a large key factor in winning championships. Non-championship teams tend to have higher average values in field goals attempted (avg_FGA) and free throws attempted (avg_FTA), suggesting a less efficient scoring performance compared to championship teams. However, generally we notice that the differences that we mention above are not significantly larger. Therefore, we need to look at more nuisances that tell the whole story that were not avaliable in the dataset. For example, playoff experience, injuries, team chemistry and different measures like these.

Conclusion

After running an anova test, a multiple linear regression model and testing the predictors, the attributes to creating a championship team are shooting as many shots as possible, getting the most rebounds and shooting for a high score (>100 pts). A limitation we encountered was that creating prediction models with very small number of truth values is extremely hard hence the small R^2 values.

Future Plans

After running these linear regression models and understanding the limitations of our data especially our response variable, the route we would take in the future is using classification/clustering. Using clustering for this goal would have given us more of an idea of reasons behind certain predictions and where and why there was crossover for other predictors. Linear regression was not the right tool to use for a binary classification and if time allowed, we would have also added a clustering section to compare against each goal's models.