EDA

Sean

2026-02-23

Context

Train.csv: contains the results of 940 regular season games. Games were played at the venue of the home team; thus, the home team may have had a home team advantage. A description of the game results data table columns, train.csv, is as follows:

GameID = Unique game ID Date = Date of match HomeConf = Home team conference HomeID = Home team ID HomeTeam = Home team name HomePts = Home team pts, e.g., points scored by the home team AwayConf = Away team conference AwayID = Away team ID AwayTeam = Away team name AwayPts = Away team pts, e.g., points scored by the away team HomeWinMargin = Home team points minus away team points, e.g., HomeWinMargin>0 indicates home team won the match, HomeWinMargin<0 indicates away team won the match, HomeWinMargin=0 indicates the game ended in a tie. Predictions.csv: contains seventy-five (75) end-of-season rivalry derby games that are to be played. Contestants must predict the results of these games in the field named "Team1_WinMargin." For example, Team1_WinMargin = 15 indicates that the user is predicting Team 1 to win by 15 points, Team1_WinMargin = -20 indicates that the user is predicting Team 1 to lose by 20 points (conversely, Team2 is predicted to win by 20 points), and Team1_WinMargin = 0 indicates the game is predicted to end in a tie. Team1_WinMargin must be entered from the perspective of Team 1.

All derby matches are played at a neutral venue with fan support split 50% for team 1 and 50% for team 2; thus, neither team has a home team advantage. A description of the prediction data table data, predictions.csv, is as follows:

GameID = Unique game ID Date = Date of match (to be played) Team1_Conf = Team 1 conference Team1_ID = Team 1 ID Team1 = Team 1 name Team2_Conf = Team 2 conference Team2_ID = Team 2 ID Team2 = Team 2 name Team1_WinMargin = User Predicted Number of Points Team 1 is expected to win. E.g., Team1_WinMargin > 0 indicates Team 1 is predicted to win the game by the specified number of points, Team1_WinMargin < 0 indicates that Team 1 is predicted to lose the game by the specified number of points, Team1_WinMargin = 0 indicates the game is expected to end in a tie.

About: There is a new sport in town - Rocketball Premier League (RPL). This is a combination of soccer, basketball, team handball, football, and cricket. The inaugural season was played in 2025 and included 165 teams across 13 conferences. There were 940 regular season games played from 1/1/25 through 6/30/25. All regular season games were played at the home team's venue.

At the end of the inaugural season, instead of a traditional US playoff system, RPL has seventy-five (75) European style rivalry "derby" games played at a neutral site. Winning teams earn bragging rights. All derby matches are played on 7/4/25. Only 150 out of the 165 teams chose to participate in the end-of-season derby.

You are hired by the RPL to be a member of their sports analytics team. You are asked to develop a prediction model that can be used to rank teams and predict the winning team and victory margin for the derby matches. Your tasks as an RPL analyst are as follows:

Predict the winning team and victory margin for seventy-five (75) end-of-season rivalry "derby" matches.

Rank all 165 teams using the results of the 940 regular season matches. For example, you must rank teams from No. 1 (Best) to No. 165 (Worst). The No. 1 (Best) ranked team will win the Rocketball Supporters Shield and be deemed RPL Champion.

Competition: Contestants are provided with a training data set (Train.csv) that includes the results of the 940 season games. Each team played between ten and fourteen regular season games. Games were played against teams in the same conference opponents and against teams in different conference.

Each contestant can develop their own ranking and prediction model. For example, these models can be based on win/loss, margin of victory, points scored, or any combination of these items. Your prediction model can incorporate the team's strength of schedule, results of common opponents, the team's conference strength, etc. All data is provided in the competition datasets.

Common types of sports analytics predictions models include counting, probability and statistical, linear and non-linear regression, weighted and moving averages, probit and logit, fuzzy logic, random forests, neural networks, machine learning, deep learning, and models of models that incorporate the results of different approaches as inputs.

Users can develop a prediction model using a tool of your choice. For example, Python, MATLAB, Excel, VBA, Java, C++.

Mission: Using the game results of the 940 matches (Train.csv file), user will perform the following calculations as part of the competition:

Rank all 165 teams from No. 1 (Best) to No. 165 (Worst). Predict the Win Margin for the 75 rivalry derby matches.

Load Libraries & Data

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.2

## Warning: package 'readr' was built under R version 4.4.3

## Warning: package 'purrr' was built under R version 4.4.2

## Warning: package 'dplyr' was built under R version 4.4.2

## Warning: package 'forcats' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(readr)
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.4.2
```

```r
predictions <- read_csv("Predictions.csv")
```

```
## Rows: 75 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Date, Team1_Conf, Team1, Team2_Conf, Team2
## dbl (3): GameID, Team1_ID, Team2_ID
## lgl (1): Team1_WinMargin
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
rankings    <- read_excel("Rankings.xlsx")
train       <- read_csv("Train.csv")
```

```
## Rows: 940 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Date, HomeConf, HomeTeam, AwayConf, AwayTeam
## dbl (6): GameID, HomeID, HomePts, AwayID, AwayPts, HomeWinMargin
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Note: For ranking system, we can use a combination of ELO rating algorithim, win margins, win/loss record, games played, and etc. - To Start, we can bucket all team into their respective conferences and start every team with a fixed base ELO rating, and then calculate ending ELO as the season has played on. We can further stratify based on conference strength, and then adjust the ELO ratings based on the strength of the opponent and the conference overall

Note: For the forecasting/modeling, the goal is predict margin of victory for 75 upcoming derby matches. There are a limited number of parameters and data points to work with; therefore, complexity of the model will be severly limited. Machine learning models will not likely be applicable here. I am thinking Bayesian regression (Conditioning on ELO ratings POSSIBILY); however, any numerical estimator will work and most should be tested along with ensemble models.

Note: STEP 1: To start, we should begin by trying to find value in the data and understanding the data. We can start by looking at the given datasets and using inference techniques to try to find any trends or patterns in the data. Or we can make more advanced columns based on aggregates of other columns.

Note: STEP 2: Next, we will build out the ELO rating algorithim for this fictional league.

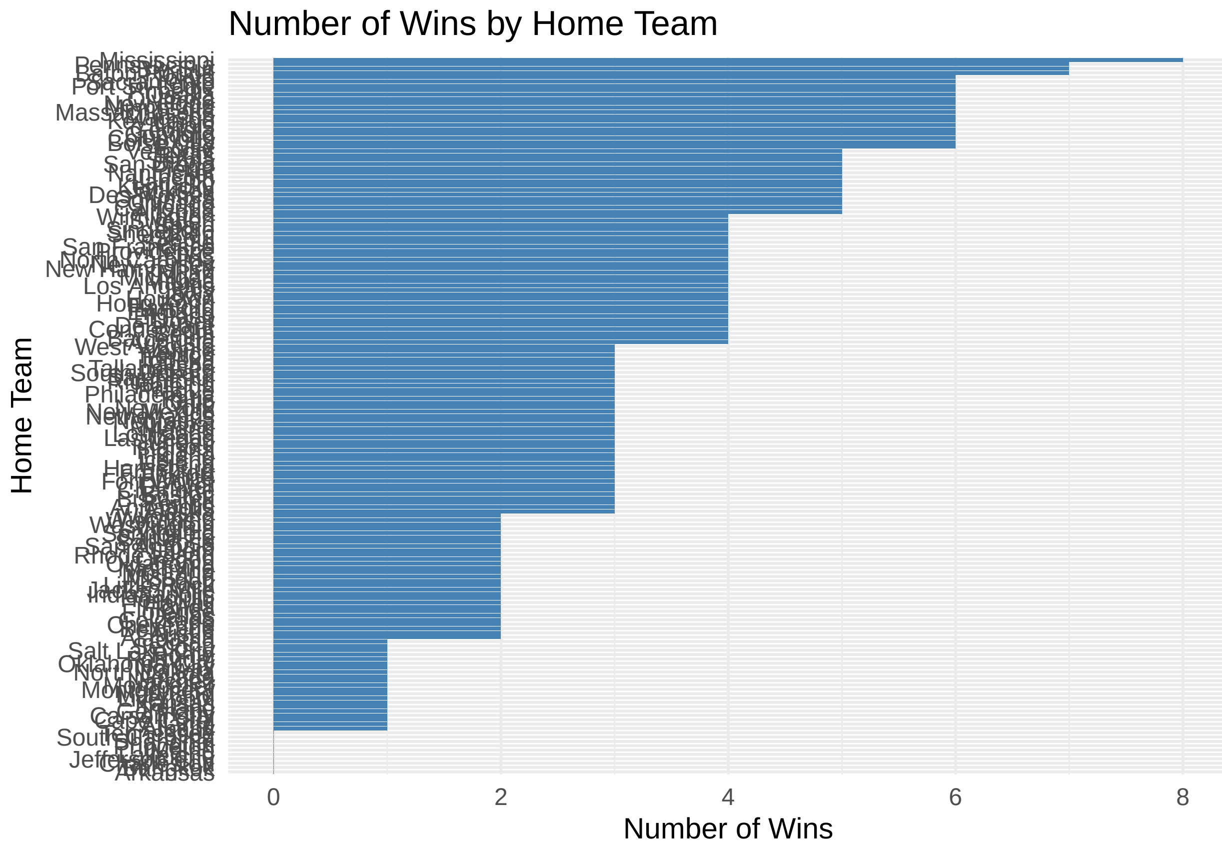
Note: STEP 3: Finally, using the ELO ratings, we can build/test multiple regression models to find the best fit for the data.

Note: STEP 4: Then, we optimize our forecasts for the 75 derby matches and submit our predictions.

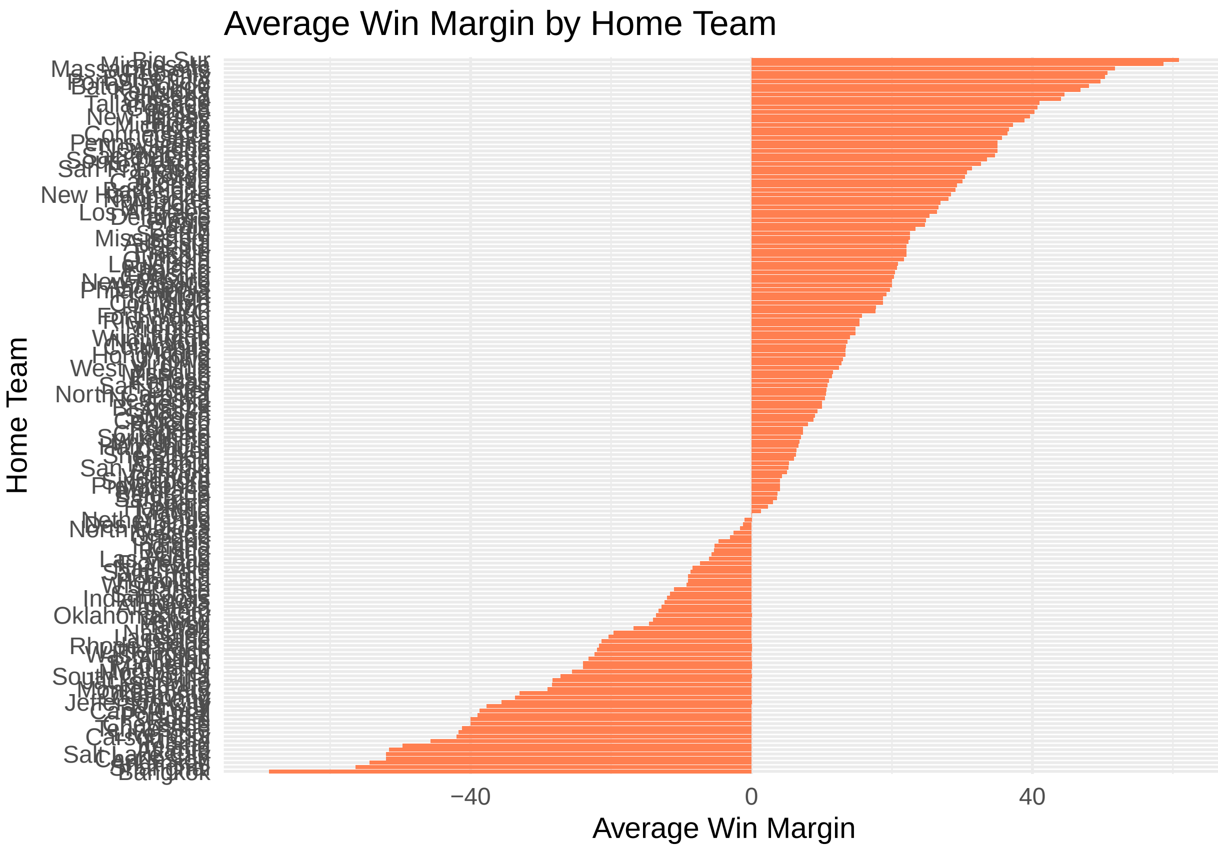
STEP 1

```r
# Explore the data and find any trends or patterns. Begin with inspecting the "train" dataset

train_summary = train %>%
  group_by(HomeTeam) %>%
  summarise(
    GamesPlayed = n(),
    Wins = sum(HomeWinMargin > 0),
```

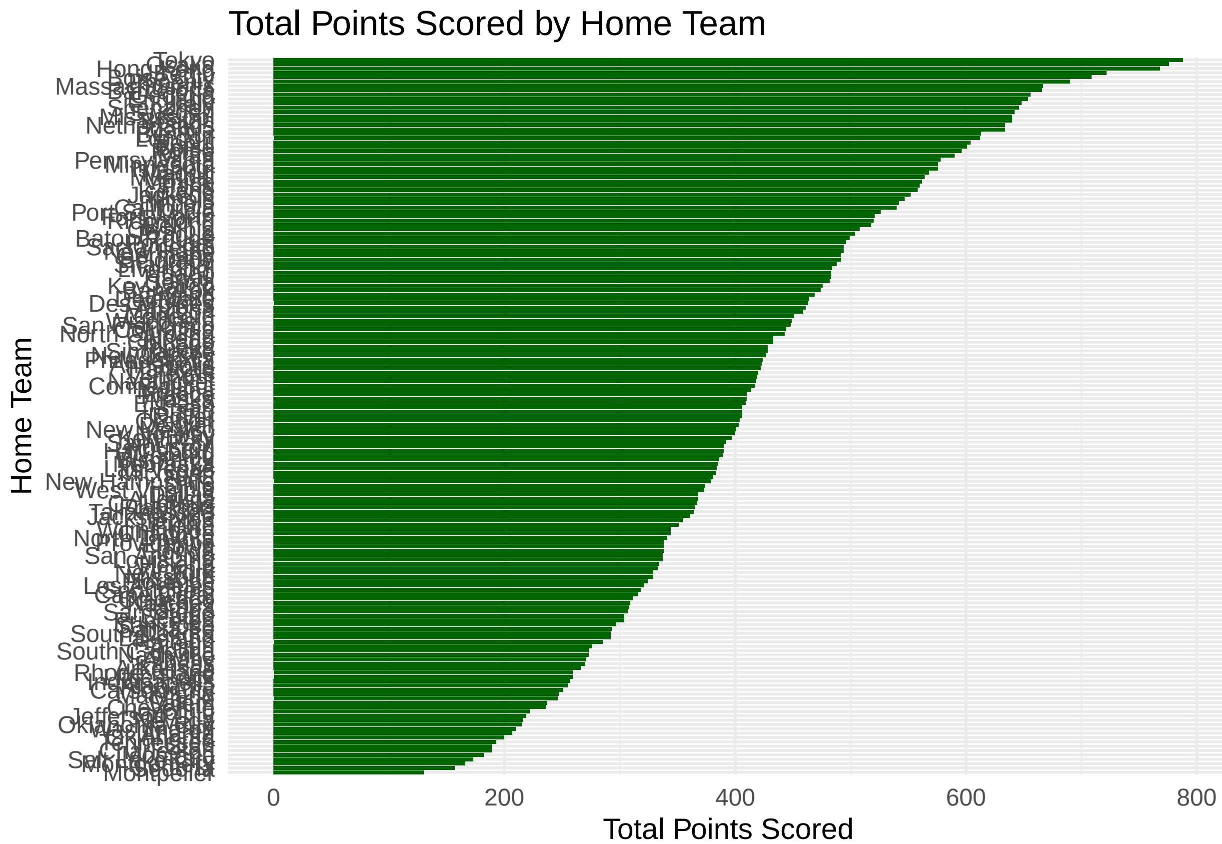
```r
    Losses = sum(HomeWinMargin < 0),
    Ties = sum(HomeWinMargin == 0),
    TotalPointsScored = sum(HomePts),
    TotalPointsAllowed = sum(AwayPts),
    AverageWinMargin = mean(HomeWinMargin)
  ) %>%
  arrange(desc(Wins))

plot_of_wins = ggplot(train_summary, aes(x = reorder(HomeTeam, Wins), y = Wins)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Number of Wins by Home Team", x = "Home Team", y = "Number of Wins") +
  theme_minimal()

plot_of_win_margin = ggplot(train_summary, aes(x = reorder(HomeTeam, AverageWinMargin), y = AverageWinMa
  geom_bar(stat = "identity", fill = "coral") +
  coord_flip() +
  labs(title = "Average Win Margin by Home Team", x = "Home Team", y = "Average Win Margin") +
  theme_minimal()

plot_of_points_scored = ggplot(train_summary, aes(x = reorder(HomeTeam, TotalPointsScored), y = TotalPoi
  geom_bar(stat = "identity", fill = "darkgreen") +
  coord_flip() +
  labs(title = "Total Points Scored by Home Team", x = "Home Team", y = "Total Points Scored") +
  theme_minimal()

plot_of_points_allowed = ggplot(train_summary, aes(x = reorder(HomeTeam, TotalPointsAllowed), y = TotalP
  geom_bar(stat = "identity", fill = "darkred") +
  coord_flip() +
  labs(title = "Total Points Allowed by Home Team", x = "Home Team", y = "Total Points Allowed") +
  theme_minimal()

plot_of_games_played = ggplot(train_summary, aes(x = reorder(HomeTeam, GamesPlayed), y = GamesPlayed)) +
  geom_bar(stat = "identity", fill = "purple") +
  coord_flip() +
  labs(title = "Number of Games Played by Home Team", x = "Home Team", y = "Number of Games Played") +
  theme_minimal()

plot_of_conference = ggplot(train, aes(x = HomeConf)) +
  geom_bar(fill = "orange") +
  labs(title = "Number of Games by Conference", x = "Conference", y = "Number of Games") +
  theme_minimal()

# Display all plots

plot_of_wins
```



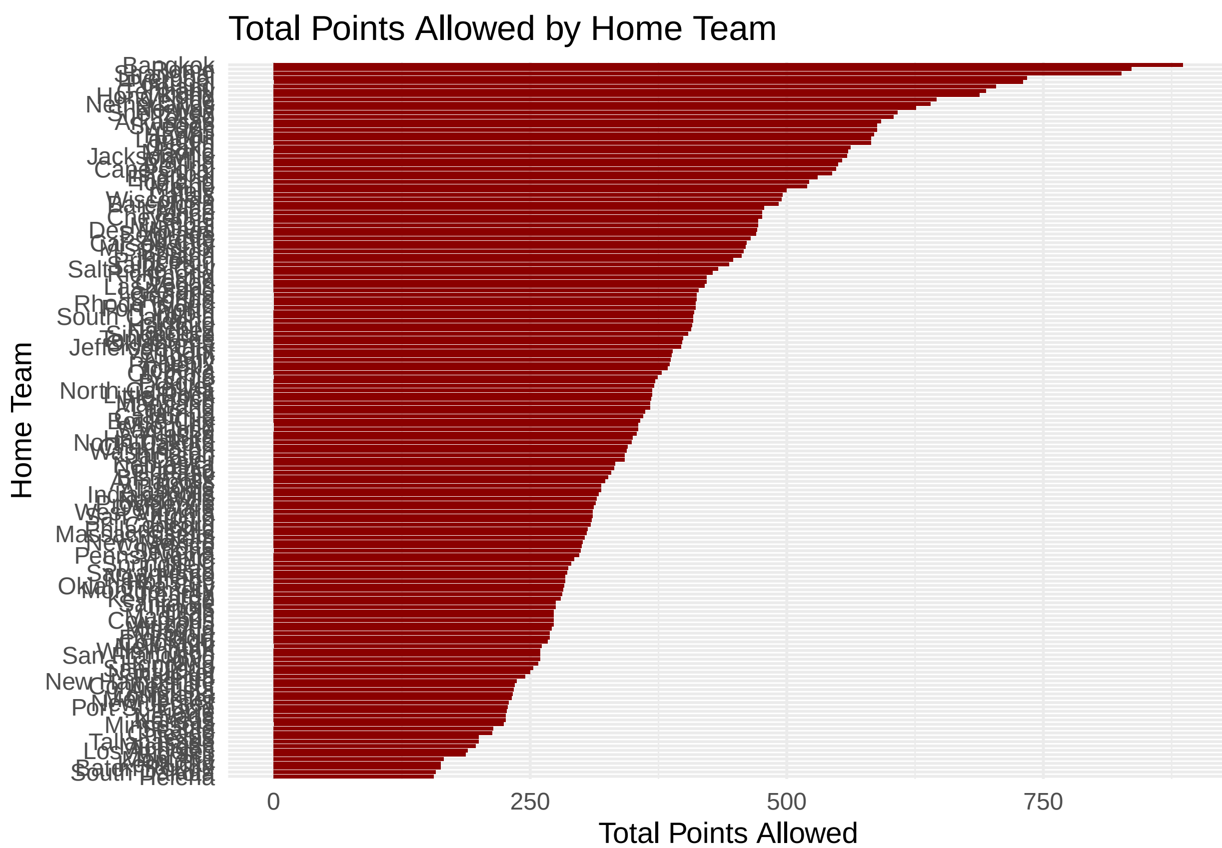

```
plot_of_win_margin
```



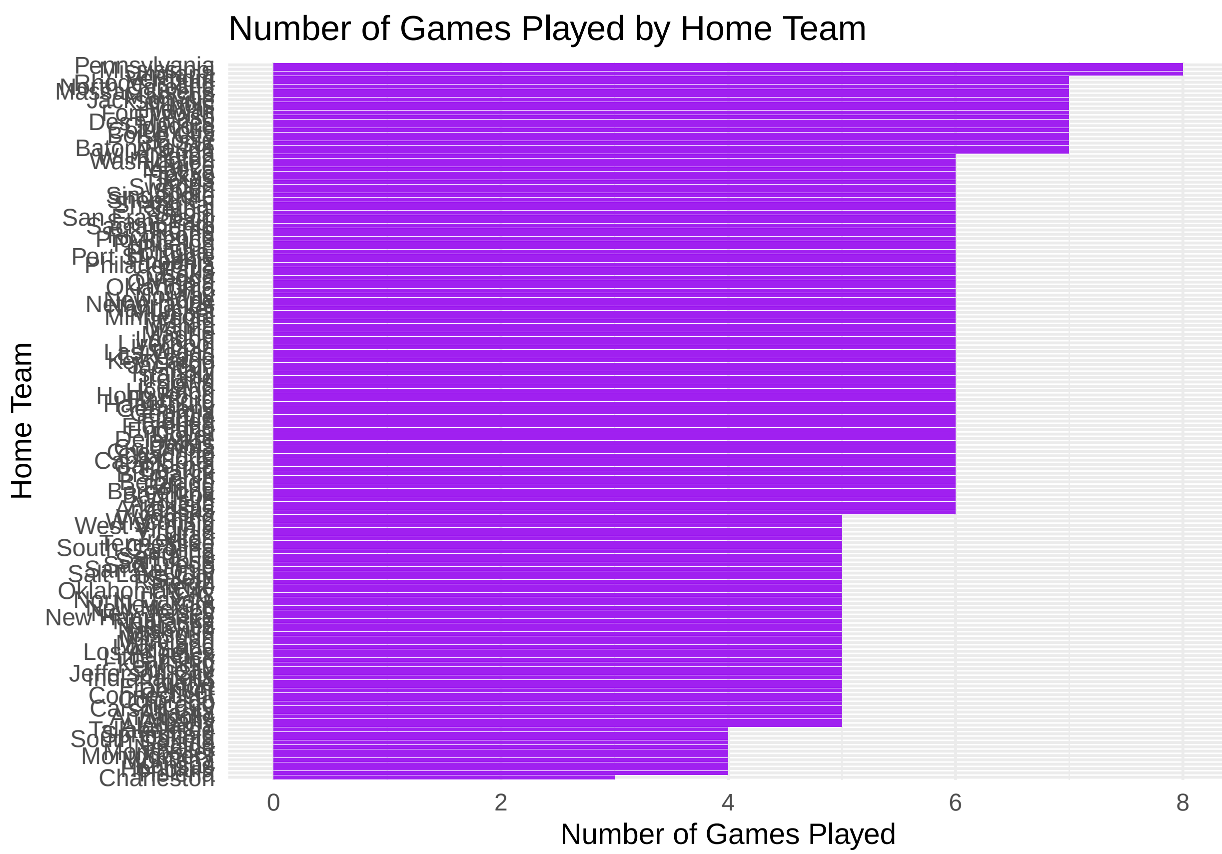

```
plot_of_points_scored
```



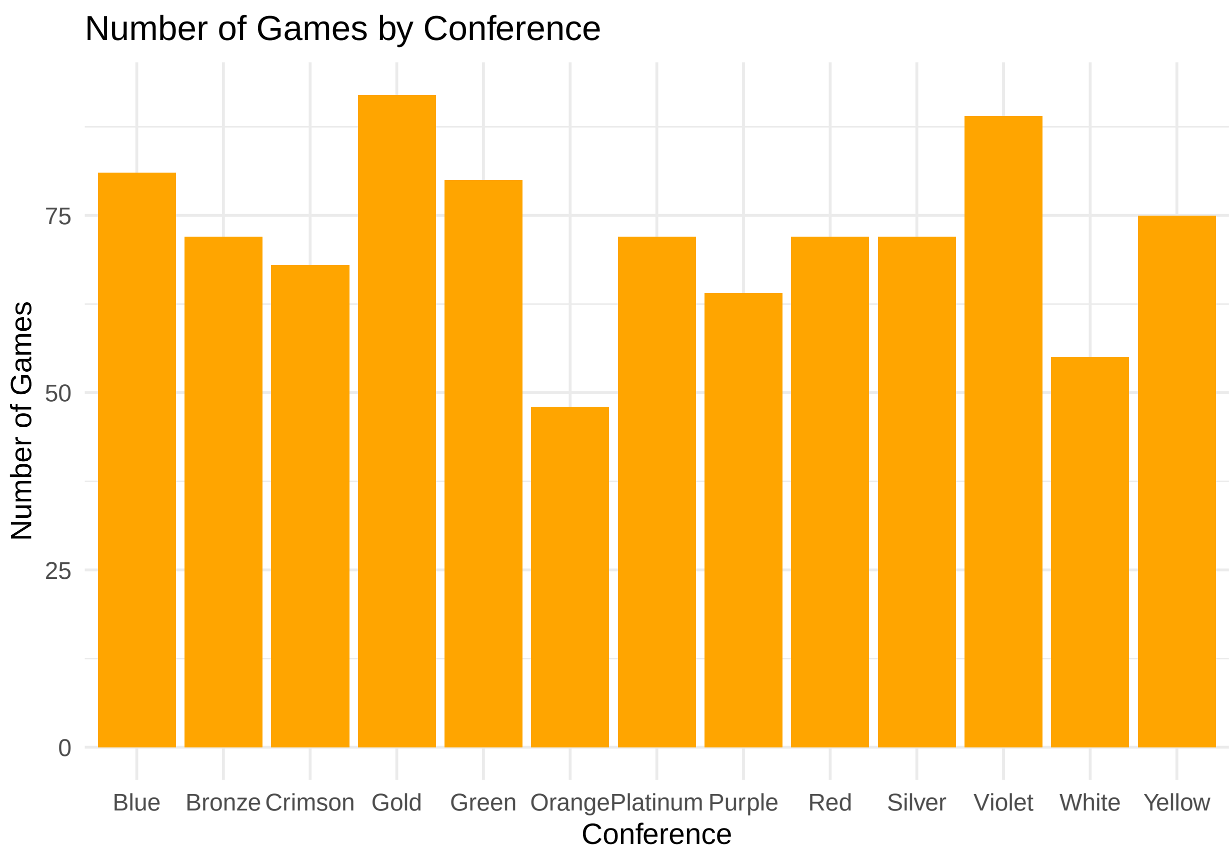

```
plot_of_points_allowed
```




```
plot_of_games_played
```




```
plot_of_conference
```




```
# Question: What kinds of inference can be gained from looking at these plots? List the plot and the po

# Plot of Wins by Home Team:
# - This plot shows the distribution of wins among the home teams. We can identify which teams have the
# Plot of Average Win Margin by Home Team:
# - This plot shows the average margin of victory for each home team. A higher average win margin may i
# Plot of Total Points Scored by Home Team:
# - This plot shows the total points scored by each home team. It can indicate which teams have strong
# Plot of Total Points Allowed by Home Team:
# - This plot shows the total points allowed by each home team. It can indicate which teams have strong
# Plot of Number of Games Played by Home Team:
# - This plot shows how many games each home team has played. It can indicate if there are any teams th
# Plot of Number of Games by Conference:
# - This plot shows the distribution of games across different conferences. It can indicate if there ar


# Question 1.1: For the "Number of Wins by Home Team" plot, what numerical distinction would you use to

# Question 1.2: For the "Average Win Margin by Home Team" plot, what numerical distinction would you us

# Question 1.3: For the "Total Points Scored by Home Team" plot, what numerical distinction would you u

# Question 1.4: For the "Total Points Allowed by Home Team" plot, what numerical distinction would you

# Question 1.5: For the "Number of Games Played by Home Team" plot, what kinds of advantages/disadvanta
```

```r
# Question 1.6: For the "Number of Games Played by Conference" plot, what kinds of advantages/disadvant

# Answers for Questions 1.1 - 1.6 (make sure to include reasons "why/how" for the "numerical distinctio

# 1.1: To identify "dominant teams" in the "Number of Wins by Home Team" plot, we could set a numerical


# 1.1 in R:

win_threshold = quantile(train_summary$Wins, 0.75)
dominant_teams = train_summary %>%
  filter(Wins > win_threshold) %>%
  select(HomeTeam, Wins)
dominant_teams
```

```
## # A tibble: 36 x 2
##    HomeTeam      Wins
##    <chr>        <int>
##  1 Mississippi      8
##  2 Baton Rouge      7
##  3 Big Sur          7
##  4 Pennsylvania     7
##  5 Boise            6
##  6 Boise City       6
##  7 Columbus         6
##  8 Concord          6
##  9 Georgia          6
## 10 Illinois         6
## # i 26 more rows
```

```r
# 1.2: To identify "strong teams" vs "weak teams" in the "Average Win Margin by Home Team" plot, we cou


# 1.2 in R:

win_margin_threshold = 10
strong_teams = train_summary %>%
  filter(AverageWinMargin > win_margin_threshold) %>%
  select(HomeTeam, AverageWinMargin)
weak_teams = train_summary %>%
  filter(AverageWinMargin <= win_margin_threshold) %>%
  select(HomeTeam, AverageWinMargin)
strong_teams
```

```
## # A tibble: 79 x 2
##    HomeTeam     AverageWinMargin
##    <chr>                   <dbl>
##  1 Mississippi              22.5
##  2 Baton Rouge              48
##  3 Big Sur                  60.9
##  4 Pennsylvania             35
##  5 Boise                    40.3
##  6 Boise City               50.3
##  7 Columbus                 13.4
```

```
##   8 Concord                 20.3
##   9 Georgia                 40.7
##  10 Illinois                38.9
## # i 69 more rows
```

```
weak_teams
```

```
## # A tibble: 86 x 2
##    HomeTeam   AverageWinMargin
##    <chr>                 <dbl>
##  1 Des Moines            -1.14
##  2 Serbia                10
##  3 Vermont                4.29
##  4 Hartford               2.33
##  5 Providence             4
##  6 Shenzhen               6.33
##  7 Singapore              4
##  8 Spain                  3
##  9 Sweden                 9
## 10 Alaska                 9.33
## # i 76 more rows
```

```
# 1.3: To identify "strong offenses" in the "Total Points Scored by Home Team" plot, we could set a num

# 1.3 in R:

points_scored_threshold = quantile(train_summary$TotalPointsScored, 0.75)
strong_offenses = train_summary %>%
  filter(TotalPointsScored > points_scored_threshold) %>%
  select(HomeTeam, TotalPointsScored)
strong_offenses
```

```
## # A tibble: 41 x 2
##    HomeTeam      TotalPointsScored
##    <chr>                     <dbl>
##  1 Mississippi                 640
##  2 Big Sur                     613
##  3 Pennsylvania                578
##  4 Boise                       601
##  5 Boise City                  709
##  6 Georgia                     656
##  7 Illinois                    547
##  8 Massachusetts               667
##  9 Minnesota                   576
## 10 Olympia                     508
## # i 31 more rows
```

```
# 1.4: To identify "strong defenses" in the "Total Points Allowed by Home Team" plot, we could set a nu

# 1.4 in R:

points_allowed_threshold = quantile(train_summary$TotalPointsAllowed, 0.25)
strong_defenses = train_summary %>%
  filter(TotalPointsAllowed < points_allowed_threshold) %>%
  select(HomeTeam, TotalPointsAllowed)
strong_defenses
```

```
## # A tibble: 41 x 2
##    HomeTeam       TotalPointsAllowed
##    <chr>                       <dbl>
##  1 Baton Rouge                   163
##  2 Big Sur                       187
##  3 Columbus                      273
##  4 Illinois                      275
##  5 Madison                       273
##  6 Minnesota                     224
##  7 Port St. Lucie                228
##  8 Arizona                       273
##  9 Chicago                       213
## 10 Kentucky                      163
## # i 31 more rows
```

```r
# 1.5: Playing more games can have both advantages and disadvantages for a team's performance in future


# 1.5 in R:

train_summary = train_summary %>%
  mutate(FatigueIndex = GamesPlayed / max(GamesPlayed)) # Example of a fatigue index based on the numbe

# 1.6: Playing more games within a conference can have advantages such as increased familiarity with op

# 1.6 in R:

conference_strength = train %>%
  group_by(HomeConf) %>%
  summarise(
    AverageWinMargin = mean(HomeWinMargin),
    TotalPointsScored = sum(HomePts),
    TotalPointsAllowed = sum(AwayPts)
  ) %>%
  arrange(desc(AverageWinMargin))
conference_strength = conference_strength %>%
  mutate(StrengthOfSchedule = AverageWinMargin / max(AverageWinMargin)) # Example of a strength of sche


# Question 1.7: What other trends, patterns, inference, or insights can you think of to gain from explo

# 1.7: Additional questions to explore the data:

#  - Is there a correlation between the number of games played and the average win margin for home team

# Answer: If there is a positive correlation, it may suggest that teams that play more games have bette

# Explore answer in R:

correlation_games_win_margin = cor(train_summary$GamesPlayed, train_summary$AverageWinMargin)
correlation_games_win_margin
```

```
## [1] 0.1564232
```

```r
#  - How does the performance of teams vary across different conferences? This could help identify if c

# Answer: If certain conferences consistently have higher average win margins or total points scored, i

# Explore answer in R:

conference_performance = train %>%
  group_by(HomeConf) %>%
  summarise(
    AverageWinMargin = mean(HomeWinMargin),
    TotalPointsScored = sum(HomePts),
    TotalPointsAllowed = sum(AwayPts)
  ) %>%
  arrange(desc(AverageWinMargin))
conference_performance
```

```
## # A tibble: 13 x 4
##    HomeConf AverageWinMargin TotalPointsScored TotalPointsAllowed
##    <chr>               <dbl>             <dbl>              <dbl>
##  1 Red                19.5                5498               4094
##  2 Blue               11.7                5267               4317
##  3 Violet             10.7                5927               4971
##  4 Gold               10.5                5692               4722
##  5 Green               9.48               5772               5014
##  6 Crimson             8.29               5118               4554
##  7 Orange              4.21               3258               3056
##  8 White               2.33               3477               3349
##  9 Platinum            1.31               7264               7170
## 10 Purple              1                  3944               3880
## 11 Yellow              0.427              4609               4577
## 12 Bronze             -0.389              6680               6708
## 13 Silver             -8.22               6228               6820
```

```r
#  - Are there any teams that consistently perform well against specific opponents or in certain condit

# Answer: If certain teams consistently perform well against specific opponents or in certain condition


#  - What is the distribution of win margins across all games, and does it differ significantly between

# Answer: If the distribution of win margins is significantly different between home and away teams, it

# Explore the answer in R:

win_margin_distribution = train %>%
  group_by(HomeTeam) %>%
  summarise(
    AverageWinMargin = mean(HomeWinMargin),
    WinMarginVariance = var(HomeWinMargin)
  ) %>%
  arrange(desc(AverageWinMargin))
win_margin_distribution
```

```
## # A tibble: 165 x 3
```

```
##    HomeTeam       AverageWinMargin WinMarginVariance
##    <chr>                     <dbl>             <dbl>
##  1 Big Sur                    60.9             1150.
##  2 Minnesota                  58.7              897.
##  3 Massachusetts              51.7             2246.
##  4 Phoenix                    50.7             1041.
##  5 Boise City                 50.3             1305.
##  6 Port St. Lucie             49.7              551.
##  7 Baton Rouge                48                984
##  8 Kentucky                   46.8              149.
##  9 Helena                     44.5             1980.
## 10 Chicago                    44                696
## # i 155 more rows
```

```
#  - How do the points scored and points allowed by teams correlate with their overall win/loss record 

# Answer: If there is a strong correlation between points scored/allowed and win/loss record or ranking

points_correlation = train_summary %>%
  summarise(
    CorrelationPointsScoredWins = cor(TotalPointsScored, Wins),
    CorrelationPointsAllowedWins = cor(TotalPointsAllowed, Wins),
    CorrelationPointsScoredRanking = cor(TotalPointsScored, GamesPlayed), # Assuming ranking is based o
    CorrelationPointsAllowedRanking = cor(TotalPointsAllowed, GamesPlayed) # Assuming ranking is based 
  )
points_correlation
```

```
## # A tibble: 1 x 4
##   CorrelationPointsScoredWins CorrelationPointsAllowedW~1 CorrelationPointsSco~2
##                         <dbl>                       <dbl>                  <dbl>
## 1                       0.609                      -0.317                  0.549
## # i abbreviated names: 1: CorrelationPointsAllowedWins,
## #   2: CorrelationPointsScoredRanking
## # i 1 more variable: CorrelationPointsAllowedRanking <dbl>
```

STEP 2: Build ELO Rating System Algorithim - USE THE "elo" package

```
library(elo)
```

```
## Warning: package 'elo' was built under R version 4.4.3
```

```
# Step 1: Initialize ratings for all 165 teams


ELO_RATINGS_v1 = data.frame(
  Team = unique(c(train$HomeTeam, train$AwayTeam)),
  Rating = 1500 # Starting ELO rating for all teams
)


# Step 2: Update ELO ratings based on game results

for (i in 1:nrow(train)) {
```

```
  home_team = train$HomeTeam[i]
  away_team = train$AwayTeam[i]
  home_points = train$HomePts[i]
  away_points = train$AwayPts[i]

  # Get current ratings
  home_rating = ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == home_team]
  away_rating = ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == away_team]

  # Calculate expected scores
  expected_home = 1 / (1 + 10^((away_rating - home_rating) / 400))
  expected_away = 1 / (1 + 10^((home_rating - away_rating) / 400))

  # Determine actual scores
  if (home_points > away_points) {
    actual_home = 1
    actual_away = 0
  } else if (home_points < away_points) {
    actual_home = 0
    actual_away = 1
  } else {
    actual_home = 0.5
    actual_away = 0.5
  }

  # Update ratings
  K = 40 # K-factor for rating updates
  new_home_rating = home_rating + K * (actual_home - expected_home)
  new_away_rating = away_rating + K * (actual_away - expected_away)

  ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == home_team] <- new_home_rating
  ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == away_team] <- new_away_rating
}


# Step 3: Consider adendums to the ELO ratings based on factors found from intial data analysis
# Goal: Create a data frame of all questions asked and the numerical result(s) from the analysis.
#       Iterate through the data frame and consider whether or not the results from the questions/analy
#       Decide if complexity in the rating system is rewarded or penalized


# Apply results from analysis (Question 1.1-1.6 & 5 questions for 1.7)

# 1.1 - Win Threshold for Dominant Teams:

# Question: Is the current rating system sufficient in identifying dominant teams based on the results


# 1.1 Answer: The current ELO rating system may not be sufficient in identifying dominant teams based s


# 1.2 - Average Win Margin Threshold for Strong vs Weak Teams:
```

```
# Question: Is the current rating system sufficient in identifying strong vs weak teams based on the av

# 1.2 Answer: The current ELO rating system may not be sufficient in identifying strong vs weak teams b


# 1.3 - Total Points Scored Threshold for Strong Offenses:

# Question: Is the current rating system sufficient in identifying strong offenses based on total point

# 1.3 Answer: The current ELO rating system may not be sufficient in identifying strong offenses based

# 1.4 - Total Points Allowed by Home Team

# Question: Is the current rating system sufficient in identifying strong defenses? If not, how can we

# 1.4 Answer: The current ELO rating system may not be sufficient in identifying strong defenses based

# 1.5 - Fatigue Index - Number of Games Played by Home Team

# Question: Is the current rating system sufficient in accounting for the advantages/disadvantages of p

# Answer: The current ELO rating system may not be sufficient in accounting for the advantages/disadvan

# Outline for ELO algo (including adendums from 1.1-1.5)

# 1. Initialize ratings for all 165 teams with a base rating (e.g., 1500).
# 2. For each game in the training dataset:
#    a. Retrieve the current ratings of the home and away teams.
#    b. Calculate the expected scores for both teams based on their current ratings.
#    c. Determine the actual scores based on the game outcome (win/loss/tie).
#    d. Update the ratings of both teams using the ELO formula, incorporating any adjustments based on
# 3. After processing all games, the final ELO ratings will reflect the performance of each team throug


# Question: How do we statistically validated the any of the adjustments made to the ELO algorithim bas

# 1.1 - To statistically validate the adjustments made to the ELO algorithm based on the insights about

# 1.2 - To statistically validate the adjustments made to the ELO algorithm based on the insights about

# 1.3 - To statistically validate the adjustments made to the ELO algorithm based on the insights about

# 1.4 - To statistically validate the adjustments made to the ELO algorithm based on the insights about

# 1.5 - To statistically validate the adjustments made to the ELO algorithm based on the insights about



# UPDATED ELO ALGORITHIM:
```

```r
# Validation Pipeline:

# Test 1.1 - 1.5 adjustments to the ELO algorithm by comparing predicted outcomes using adjusted ELO ra

# Test individually and collectively for all 1.1-1.5 adjustments to see which adjustments have the most

# Create the adjustments inside of the main algorithm loop that - when called - will test the algorithm


ELO_ALGO_TEST = function(train_data, adjustments = c("dominant_teams", "strong_offenses", "strong_defens
  # Initialize ELO ratings
  ELO_RATINGS = data.frame(
    Team = unique(c(train_data$HomeTeam, train_data$AwayTeam)),
    Rating = 1500 # Starting ELO rating for all teams
  )

  # Store predictions and actual outcomes for validation
  predictions = data.frame(
    HomeTeam = character(),
    AwayTeam = character(),
    PredictedOutcome = numeric(),
    ActualOutcome = numeric()
  )

  for (i in 1:nrow(train_data)) {
    home_team = train_data$HomeTeam[i]
    away_team = train_data$AwayTeam[i]
    home_points = train_data$HomePts[i]
    away_points = train_data$AwayPts[i]

    # Get current ratings
    home_rating = ELO_RATINGS$Rating[ELO_RATINGS$Team == home_team]
    away_rating = ELO_RATINGS$Rating[ELO_RATINGS$Team == away_team]

    # Calculate expected scores
    expected_home = 1 / (1 + 10^((away_rating - home_rating) / 400))
    expected_away = 1 / (1 + 10^((home_rating - away_rating) / 400))

    # Determine actual scores
    if (home_points > away_points) {
      actual_home = 1
      actual_away = 0
    } else if (home_points < away_points) {
      actual_home = 0
      actual_away = 1
    } else {
      actual_home = 0.5
      actual_away = 0.5
    }
```

```r
  # Store predictions for validation
  predictions <- rbind(predictions, data.frame(
    HomeTeam = home_team,
    AwayTeam = away_team,
    PredictedOutcome = expected_home,
    ActualOutcome = actual_home
  ))

  # Update ratings with adjustments based on insights from data analysis
  K = 40 # K-factor for rating updates

  # Adjustments for dominant teams, strong offenses/defenses, and fatigue index can be incorporated h


  dominant_teams_adjustment = if ("dominant_teams" %in% adjustments) {
    # Example adjustment for dominant teams
    if (home_team %in% dominant_teams$HomeTeam) {
      home_rating <- home_rating + 20 # Bonus for dominant teams
    }
    if (away_team %in% dominant_teams$HomeTeam) {
      away_rating <- away_rating + 20 # Bonus for dominant teams
    }
  }

  strong_offenses_adjustment = if ("strong_offenses" %in% adjustments) {
    # Example adjustment for strong offenses
    if (home_team %in% strong_offenses$HomeTeam) {
      home_rating <- home_rating + 10 # Bonus for strong offenses
    }
    if (away_team %in% strong_offenses$HomeTeam) {
      away_rating <- away_rating + 10 # Bonus for strong offenses
    }
  }

  strong_defenses_adjustment = if ("strong_defenses" %in% adjustments) {
    # Example adjustment for strong defenses
    if (home_team %in% strong_defenses$HomeTeam) {
      home_rating <- home_rating + 10 # Bonus for strong defenses
    }
    if (away_team %in% strong_defenses$HomeTeam) {
      away_rating <- away_rating + 10 # Bonus for strong defenses
    }
  }

  fatigue_index_adjustment = if ("fatigue_index" %in% adjustments) {
    # Example adjustment for fatigue index
    home_fatigue_index = train_summary$FatigueIndex[train_summary$HomeTeam == home_team]
    away_fatigue_index = train_summary$FatigueIndex[train_summary$HomeTeam == away_team]

    home_rating <- home_rating - (home_fatigue_index * 20) # Penalty for fatigue
    away_rating <- away_rating - (away_fatigue_index * 20) # Penalty for fatigue
  }
```

```
    new_home_rating = home_rating + K * (actual_home - expected_home)
    new_away_rating = away_rating + K * (actual_away - expected_away)


    ELO_RATINGS$Rating[ELO_RATINGS$Team == home_team] <- new_home_rating
    ELO_RATINGS$Rating[ELO_RATINGS$Team == away_team] <- new_away_rating
  }

  # Validate predictions using MAE, RMSE, and hypothesis testing
  mae = mean(abs(predictions$PredictedOutcome - predictions$ActualOutcome))
  rmse = sqrt(mean((predictions$PredictedOutcome - predictions$ActualOutcome)^2))

  # Hypothesis testing can be conducted here to compare the adjusted ELO ratings against the original E

  return(list(
    ELO_RATINGS = ELO_RATINGS,
    Predictions = predictions,
    MAE = mae,
    RMSE = rmse
    # Include results of hypothesis testing here
  ))
}


# Execution:

results = ELO_ALGO_TEST(train_data = train, adjustments = c("dominant_teams", "strong_offenses", "stron


# Using "results", cross-validate the adjustments to determine which have the most statistical signific


# Question: After the ELO_ALGO_TEST was run and data was collected, what else is needed to test, valida

# 1. Analyze the results of the ELO_ALGO_TEST, specifically looking at the MAE and RMSE metrics to eval
# 2. Conduct hypothesis testing to determine if the adjustments made to the ELO algorithm (for dominant
# 3. Based on the results of the validation tests, identify which adjustments have the most statistical
# 4. Refine the ELO_ALGO_TEST function to create a finalized ELO_ALGO_FINAL function that incorporates




# Step 1 in R:
results = ELO_ALGO_TEST(train_data = train, adjustments = c("dominant_teams", "strong_offenses", "stron
mae = results$MAE
rmse = results$RMSE
print(paste("MAE:", mae))
```

```
## [1] "MAE: 0.426608830308534"
```

```
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 0.451006144571495"
```

```
# MAE: 0.426608830308534
# RMSE: 0.451006144571495

# Step 2 in R:

# Hypothesis testing can be conducted using a paired t-test to compare the predicted outcomes from the

# Original ELO Predictions:

original_predictions = data.frame(
  HomeTeam = character(),
  AwayTeam = character(),
  PredictedOutcome = numeric(),
  ActualOutcome = numeric()
)
for (i in 1:nrow(train)) {
  home_team = train$HomeTeam[i]
  away_team = train$AwayTeam[i]
  home_points = train$HomePts[i]
  away_points = train$AwayPts[i]

  # Get current ratings from original ELO ratings
  home_rating = ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == home_team]
  away_rating = ELO_RATINGS_v1$Rating[ELO_RATINGS_v1$Team == away_team]

  # Calculate expected scores using original ELO ratings
  expected_home = 1 / (1 + 10^((away_rating - home_rating) / 400))

  # Determine actual scores
  if (home_points > away_points) {
    actual_home = 1
  } else if (home_points < away_points) {
    actual_home = 0
  } else {
    actual_home = 0.5
  }

  # Store predictions for validation
  original_predictions <- rbind(original_predictions, data.frame(
    HomeTeam = home_team,
    AwayTeam = away_team,
    PredictedOutcome = expected_home,
    ActualOutcome = actual_home
  ))
}


# Now we have both the predictions from the adjusted ELO ratings and the results from the original elo

# Paired t-test to compare the predicted outcomes from the adjusted ELO ratings against the actual outc

t_test_result = t.test(results$Predictions$PredictedOutcome, original_predictions$PredictedOutcome, pai
print(t_test_result)
```

```
##
##  Paired t-test
##
## data:  results$Predictions$PredictedOutcome and original_predictions$PredictedOutcome
## t = -1.7668, df = 939, p-value = 0.07759
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.013941973  0.000731784
## sample estimates:
## mean difference
##    -0.006605095
```

```r
# Results:
# t  = -1.7668
# df = 939
# p-value = 0.07759
# Sample estimates: Mean Difference: -0.006605095
# 95% CI: [-0.013941973, 0.000731784]


# Step 3 in R - Which Adjustments had the most statistical significance in improving accuracy?

# To determine which adjustments had the most statistical significance in improving accuracy, we would

# Run 1: Only "dominant_teams" adjustment

results_dominant_teams = ELO_ALGO_TEST(train_data = train, adjustments = c("dominant_teams"))
mae_dominant_teams = results_dominant_teams$MAE
rmse_dominant_teams = results_dominant_teams$RMSE
print(paste("MAE with Dominant Teams Adjustment:", mae_dominant_teams))
```

```
## [1] "MAE with Dominant Teams Adjustment: 0.438924806542782"
```

```r
print(paste("RMSE with Dominant Teams Adjustment:", rmse_dominant_teams))
```

```
## [1] "RMSE with Dominant Teams Adjustment: 0.457196940180702"
```

```r
# MAE with Dominant Teams: 0.438
# RMSE with Dominant Teams: 0.457

# Run 2: Only "strong_offenses" adjustment

results_strong_offenses = ELO_ALGO_TEST(train_data = train, adjustments = c("strong_offenses"))
mae_strong_offenses = results_strong_offenses$MAE
rmse_strong_offenses = results_strong_offenses$RMSE
print(paste("MAE with Strong Offenses Adjustment:", mae_strong_offenses))
```

```
## [1] "MAE with Strong Offenses Adjustment: 0.454860438932721"
```

```r
print(paste("RMSE with Strong Offenses Adjustment:", rmse_strong_offenses))
```

```
## [1] "RMSE with Strong Offenses Adjustment: 0.466590167987282"
```

```r
# MAE with strong offenses: 0.455
# RMSE with strong offenses: 0.467

# Run 3: Only "strong_defenses" adjustment
```

```r
results_strong_defenses = ELO_ALGO_TEST(train_data = train, adjustments = c("strong_defenses"))
mae_strong_defenses = results_strong_defenses$MAE
rmse_strong_defenses = results_strong_defenses$RMSE
print(paste("MAE with Strong Defenses Adjustment:", mae_strong_defenses))
```

```
## [1] "MAE with Strong Defenses Adjustment: 0.456683162034806"
```

```r
print(paste("RMSE with Strong Defenses Adjustment:", rmse_strong_defenses))
```

```
## [1] "RMSE with Strong Defenses Adjustment: 0.4687096591706"
```

```r
# MAE with strong defenses: 0.457
# RMSE with strong defesnes: 0.469

# Run 4: Only "fatigue_index" adjustment

results_fatigue_index = ELO_ALGO_TEST(train_data = train, adjustments = c("fatigue_index"))
mae_fatigue_index = results_fatigue_index$MAE
rmse_fatigue_index = results_fatigue_index$RMSE
print(paste("MAE with Fatigue Index Adjustment:", mae_fatigue_index))
```

```
## [1] "MAE with Fatigue Index Adjustment: 0.465619602581491"
```

```r
print(paste("RMSE with Fatigue Index Adjustment:", rmse_fatigue_index))
```

```
## [1] "RMSE with Fatigue Index Adjustment: 0.474598493952068"
```

```r
# MAE with fatigue index: 0.466
# RMSE with fatigue index: 0.475

# Conclusion for Step 3:
# Based on the MAE and RMSE metrics for each individual adjustment, it appears that the "dominant_teams


# Step 4 in R - Refine the ELO_ALGO_TEST function to create a finalized ELO_ALGO_FINAL function that in

ELO_ALGO_FINAL = function(train_data) {
  # Initialize ELO ratings
  ELO_RATINGS = data.frame(
    Team = unique(c(train_data$HomeTeam, train_data$AwayTeam)),
    Rating = 1500 # Starting ELO rating for all teams
  )

  for (i in 1:nrow(train_data)) {
    home_team = train_data$HomeTeam[i]
    away_team = train_data$AwayTeam[i]
    home_points = train_data$HomePts[i]
    away_points = train_data$AwayPts[i]

    # Get current ratings
    home_rating = ELO_RATINGS$Rating[ELO_RATINGS$Team == home_team]
    away_rating = ELO_RATINGS$Rating[ELO_RATINGS$Team == away_team]

    # Calculate expected scores
    expected_home = 1 / (1 + 10^((away_rating - home_rating) / 400))
```

```
    # Determine actual scores
    if (home_points > away_points) {
      actual_home = 1
      actual_away = 0
    } else if (home_points < away_points) {
      actual_home = 0
      actual_away = 1
    } else {
      actual_home = 0.5
      actual_away = 0.5
    }

    # Update ratings with adjustments based on insights from data analysis (only dominant teams adjustm
    K = 40 # K-factor for rating updates

    dominant_teams_adjustment = if (home_team %in% dominant_teams$HomeTeam) {
      home_rating <- home_rating + 20 # Bonus for dominant teams
    }
    if (away_team %in% dominant_teams$HomeTeam) {
      away_rating <- away_rating + 20 # Bonus for dominant teams
    }

    new_home_rating = home_rating + K * (actual_home - expected_home)
    new_away_rating = away_rating + K * (actual_away - expected_away)

    ELO_RATINGS$Rating[ELO_RATINGS$Team == home_team] <- new_home_rating
    ELO_RATINGS$Rating[ELO_RATINGS$Team == away_team] <- new_away_rating
  }

  return(ELO_RATINGS)
}
```

STEP 3: Modeling/Forecasting - Create a statistical model that can fit the results of HomeWinMargin and then use that model to predict the HomeWinMargin for the next 75 matchups

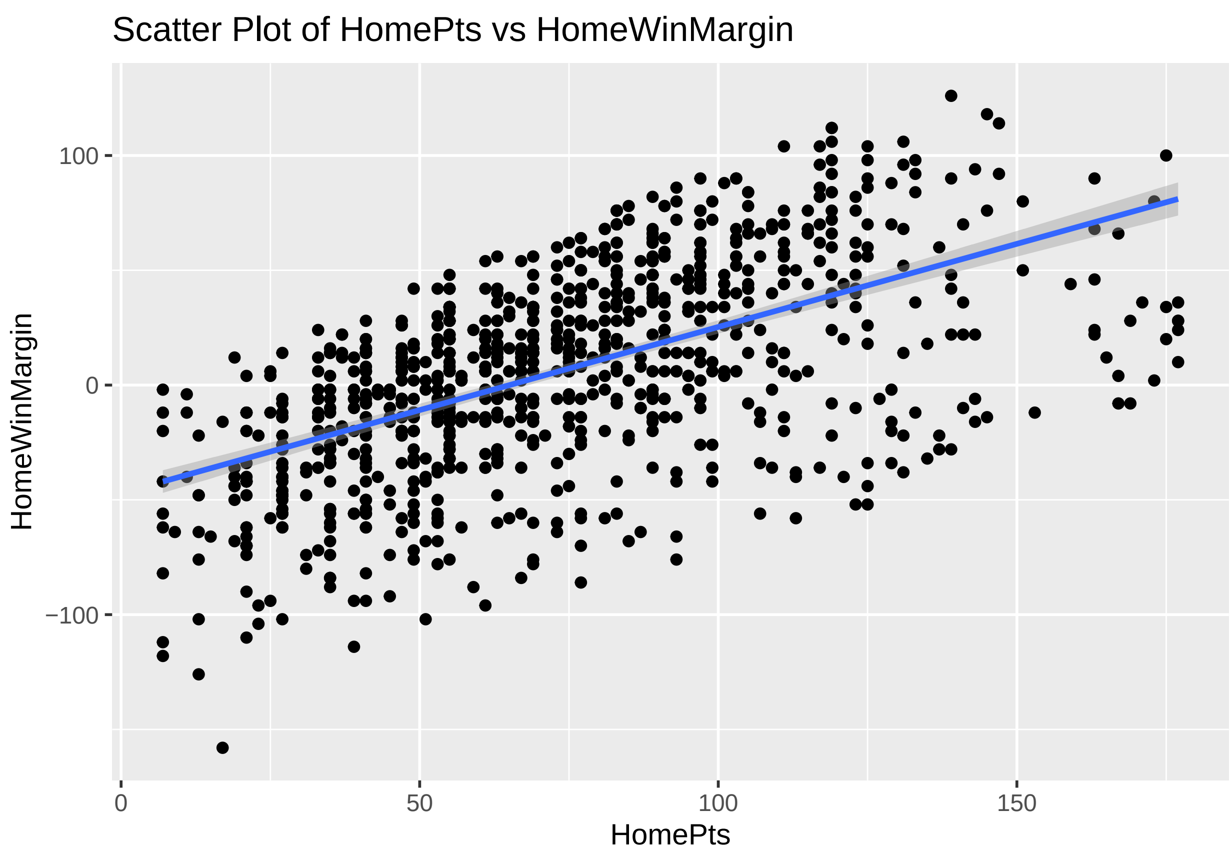
```
# Plan:
# 1. Explore the relationship between HomeWinMargin and other variables in the dataset (e.g., HomePts, 
# 2. Choose an appropriate statistical modeling technique (e.g., linear regression, random forest, etc.
# 3. Train the chosen model on the training dataset, using HomeWinMargin as the target variable and the
# 4. Evaluate the performance of the model using appropriate metrics (e.g., R-squared, mean absolute er
# 5. Use the trained model to predict the HomeWinMargin for the next 75 matchups, using the relevant pr


# 3.1 - What predictors should we use? What is the relationship between HomeWinMargin and other variabl

# 3.1 Answer: To identify potential predictors for the model, we can explore the relationships between 

# Correlation Coefficients:

correlation_home_win_margin = train %>%
```

```
  summarise(
    CorrelationHomePts = cor(HomeWinMargin, HomePts),
    CorrelationAwayPts = cor(HomeWinMargin, AwayPts),
    CorrelationHomeConf = cor(HomeWinMargin, as.numeric(as.factor(HomeConf))),
    CorrelationAwayConf = cor(HomeWinMargin, as.numeric(as.factor(AwayConf)))
  )
correlation_home_win_margin
```

```
## # A tibble: 1 x 4
##   CorrelationHomePts CorrelationAwayPts CorrelationHomeConf CorrelationAwayConf
##                <dbl>              <dbl>               <dbl>               <dbl>
## 1              0.576             -0.640             -0.0391              0.0939
```

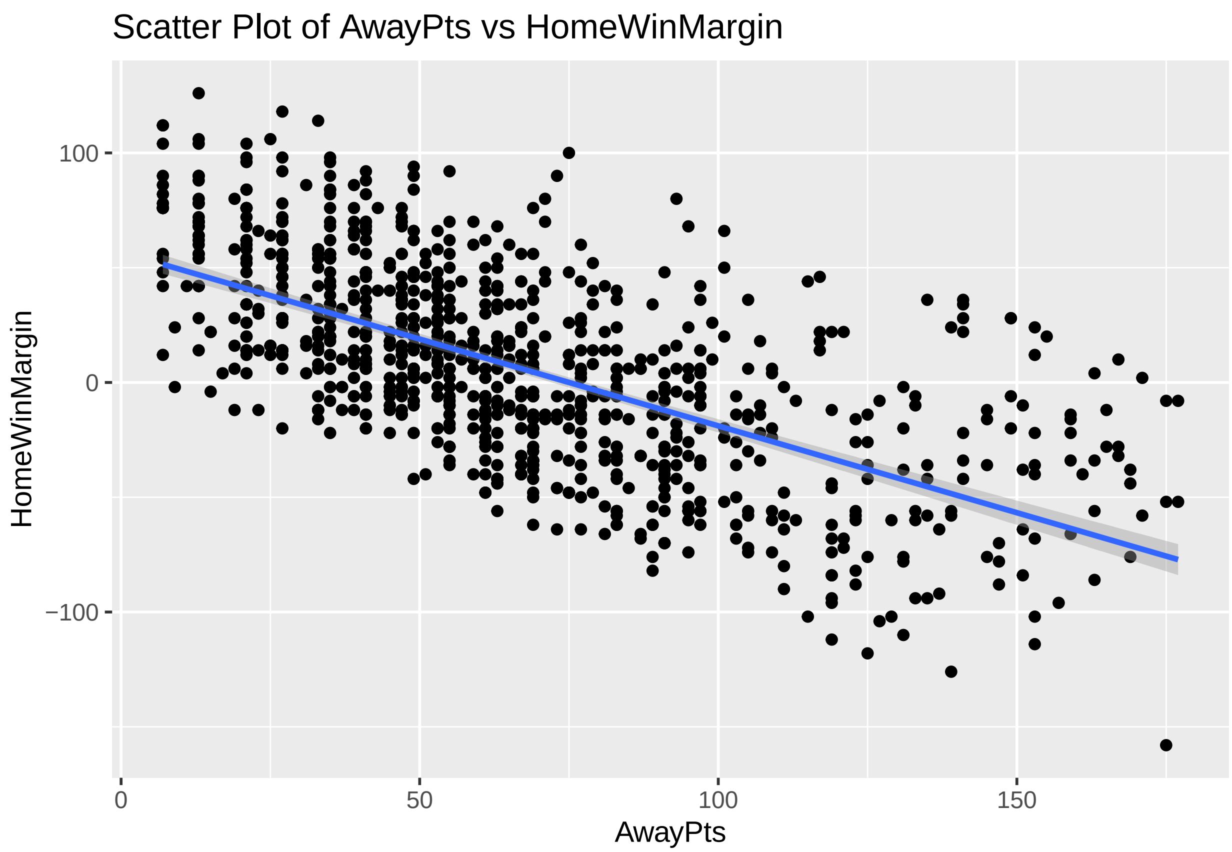
```
# CorrelationHomePts: 0.5758479
# CorrelationAwayPts: -0.6403852
# CorrelationHomeConf: -0.03906505
# CorrelationAwayConf: 0.09389065

# Scatter & Box Plots:

ggplot(train, aes(x = HomePts, y = HomeWinMargin)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Scatter Plot of HomePts vs HomeWinMargin")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```r
ggplot(train, aes(x = AwayPts, y = HomeWinMargin)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Scatter Plot of AwayPts vs HomeWinMargin")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# Conclusion: Based on the correlation coefficients and the scatter plots, it appears that HomePts and

# Follow-up question: Win margin is caluclated using home and away points directly, is it trivial or re

# Answer: Using HomePts and AwayPts as predictors for HomeWinMargin may be considered trivial or redund


# 3.1.1 - Avoiding Overfitting


# Keep in mind that: All derby matches are played at a neutral venue with fan support split 50% for tea

# Derived features
# - Ratio of HomePts to AwayPts
# - Difference in ELO ratings between the two teams (using the ELO ratings calculated in Step 2)
# - Fatigue index for both teams (number of games played leading up to the matchup)
# - Historical performance of the teams in derby matches (e.g., average win margin in previous derby ma
# - Team strength indicators (e.g., average points scored and allowed throughout the season, ranking ba
# - Possible ELO metrics, rolling averages of ELO ratings over the last 5-10 games, etc. To capture tea
```

```r
# - There needs to be guardrails put in place so the values for this are not incorrect. i.e. GameID 100
# - All/any metrics that are derived need to be calculated in a way that they are not using information
# - We can use the "lag" function to create features that are based on historical performance, ensuring
# - When we are creating derived features, we need to ensure that we are not introducing any data leaka
# - When using ELO derived features, each observation needs to reflect the correct index of the ELO rat


safe_derived_features = train %>%
  group_by(HomeTeam) %>%
  arrange(GameID) %>%
  mutate(
    RatioHomeAwayPts = HomePts / (AwayPts + 1), # Adding 1 to avoid division by zero
    ELO_Difference = ELO_RATINGS_v1$Rating[match(HomeTeam, ELO_RATINGS_v1$Team)] - ELO_RATINGS_v1$Ratin
    HomeFatigueIndex = cumsum(HomeTeam == lag(HomeTeam, default = first(HomeTeam))),
    AwayFatigueIndex = cumsum(AwayTeam == lag(AwayTeam, default = first(AwayTeam))),
    HistoricalWinMargin = ifelse(GameID > 1001, lag(cumsum(HomeWinMargin), default = 0) / (cumsum(HomeWi
  ) %>%
  ungroup()

safe_derived_features
```

```
## # A tibble: 940 x 16
##    GameID Date     HomeConf HomeID HomeTeam     HomePts AwayConf AwayID AwayTeam
##     <dbl> <chr>    <chr>     <dbl> <chr>          <dbl> <chr>     <dbl> <chr>
##  1   1001 1/1/2025 Yellow      123 Maine             21 Red          85 New Hope
##  2   1002 1/1/2025 Green        53 Wyoming           77 Purple       70 Maryland
##  3   1003 1/1/2025 Purple       68 Idaho            123 Red          83 Jeffers~
##  4   1004 1/1/2025 Purple       74 Salt Lake C~      51 Crimson      22 Michigan
##  5   1005 1/2/2025 Gold         33 El Paso           41 Orange       57 Denver
##  6   1006 1/2/2025 Crimson      26 Phoenix          129 Yellow      117 Columbia
##  7   1007 1/2/2025 Green        46 Boston           125 Crimson      25 Oregon
##  8   1008 1/2/2025 Red          79 Des Moines        81 White       110 Raleigh
##  9   1009 1/2/2025 Gold         28 San Antonio       49 Gold         36 Massach~
## 10   1010 1/2/2025 Crimson      19 Georgia           81 White       115 Wiscons~
## # i 930 more rows
## # i 7 more variables: AwayPts <dbl>, HomeWinMargin <dbl>,
## #   RatioHomeAwayPts <dbl>, ELO_Difference <dbl>, HomeFatigueIndex <int>,
## #   AwayFatigueIndex <int>, HistoricalWinMargin <dbl>
```