Simulating the 2018-2019 NBA Season

Chris Park

Setting up the datasets

```
# Reading in initial Elos and scores as data frames
team_info = read.table("nba_initial_elos.csv", header=TRUE, sep=',')
scores = read.table("nba_scores.csv", header=TRUE, sep=',')

simulated_season = 2018

# Obtain list of unique conference names and unique division names
conferences = na.omit(unique(team_info$conference))
divisions = na.omit(unique(team_info$division))

# Create list of games that occurred prior to season being simulated
pre_season = subset(scores, scores$season < simulated_season & scores$season
>= 1946)

# Create list of regular season games for season being simulated
season_schedule = subset(scores, scores$season == simulated_season &
scores$game_type == "r")
```

Calculating Preseason Elo Ratings

```
weight = 8.7
hfa = 78.79

# Iterate through all games in the sport's history up to season being
simulated
for(i in 1:nrow(pre_season)) {

  # Find indices corresponding to home and away teams for current game
  home_index = which(team_info$team == pre_season$home_team[i])
  away_index = which(team_info$team == pre_season$away_team[i])

  # Find home and away team Elo ratings
  home_elo = team_info$rating[home_index]
  away_elo = team_info$rating[away_index]

  # Calculate home team win probability
  win_prob = 1 / (10^((away_elo - (home_elo +
hfa*pre_season$neutral[i]))/400) + 1)

  # Calculate actual margin of victory - must be positive
```

```
  score_diff = abs(pre_season$home_score[i] - pre_season$away_score[i])

  # Determine home team result
  if(pre_season$home_score[i] > pre_season$away_score[i]) {
    home_result = 1  # Home team wins
  } else if(pre_season$home_score[i] < pre_season$away_score[i]) {
    home_result = 0  # Home team loses
  } else {
    home_result = 0.5  # Tie
  }

  # Calculate amount each team's Elo rating is adjusted by
  home_elo_adjustment = weight * log(score_diff + 1) * (home_result -
win_prob)

  # Adjust Elo ratings - add point to winner and subtract points from loser
  team_info$rating[home_index] = team_info$rating[home_index] +
home_elo_adjustment
  team_info$rating[away_index] = team_info$rating[away_index] -
home_elo_adjustment

  # Adjust Elo ratings at end of season to regress 1/3 of the way towards
1500
  if(i < nrow(scores) && scores$season[i+1] > scores$season[i]) {
    for(j in 1:nrow(team_info)) {
      if(scores$season[i] >= team_info$inaugural_season[j]) {
        team_info$rating[j] = team_info$rating[j] - (team_info$rating[j] -
1500)/3
      }
    }

    # Identify all teams that existed at beginning of following season
    existing_teams = team_info[which(team_info$inaugural_season <=
(scores$season[i] + 1)),]

    # Calculate amount each team's Elo rating must be adjusted by to make
mean 1500
    expansion_adjustment = -1*(mean(existing_teams$rating) - 1500)

    # Perform expansion adjustment on teams that existed at beginning of
following season
    for(j in 1:nrow(team_info)) {
      if((scores$season[i] + 1) >= team_info$inaugural_season[j]) {
        team_info$rating[j] = team_info$rating[j] + expansion_adjustment
      }
    }
  }
}
# Getting rid of inaugural_season from the data frame
```

```r
team_info = subset(team_info, select = -c(inaugural_season))

# Team name and ratings of preseason Elo ratings
preseason_elos = team_info
preseason_elos
```

```
##                       team conference  division   rating
## 1             Brooklyn Nets    eastern  atlantic 1415.674
## 2          Charlotte Hornets    eastern southeast 1477.844
## 3              Chicago Bulls    eastern   central 1385.119
## 4        Cleveland Cavaliers    eastern   central 1556.645
## 5           Dallas Mavericks    western southwest 1402.775
## 6            Denver Nuggets    western northwest 1548.107
## 7            Houston Rockets    western southwest 1653.111
## 8             Indiana Pacers    eastern   central 1547.372
## 9       Los Angeles Clippers    western   pacific 1505.576
## 10         Memphis Grizzlies    western southwest 1378.706
## 11                Miami Heat    eastern southeast 1501.926
## 12           Milwaukee Bucks    eastern   central 1511.652
## 13  Minnesota Timberwolves    western northwest 1529.850
## 14    New Orleans Pelicans    western southwest 1561.625
## 15   Oklahoma City Thunder    western northwest 1565.474
## 16             Orlando Magic    eastern southeast 1383.716
## 17              Phoenix Suns    western   pacific 1340.832
## 18   Portland Trailblazers    western northwest 1550.620
## 19        San Antonio Spurs    western southwest 1536.468
## 20          Toronto Raptors    eastern  atlantic 1583.975
## 21                Utah Jazz    western northwest 1598.934
## 22      Washington Wizards    eastern southeast 1500.161
## 23            Boston Celtics    eastern  atlantic 1564.219
## 24   Golden State Warriors    western   pacific 1662.123
## 25           New York Knicks    eastern  atlantic 1405.056
## 26       Philadelphia 76ers    eastern  atlantic 1589.076
## 27             Atlanta Hawks    eastern southeast 1392.873
## 28           Detroit Pistons    eastern   central 1481.072
## 29       Los Angeles Lakers    western   pacific 1477.896
## 30          Sacramento Kings    western   pacific 1391.535
## 31          Anderson Packers       <NA>      <NA> 1499.999
## 32             Chicago Stags       <NA>      <NA> 1499.999
## 33          Cleveland Rebels       <NA>      <NA> 1499.999
## 34    Denver Nuggets (1st)       <NA>      <NA> 1499.999
## 35           Detroit Falcons       <NA>      <NA> 1499.999
## 36         Indianapolis Jets       <NA>      <NA> 1499.999
## 37  Indianapolis Olympians       <NA>      <NA> 1499.999
## 38       Pittsburgh Ironmen       <NA>      <NA> 1499.999
## 39 Providence Steamrollers       <NA>      <NA> 1499.999
## 40       Sheboygan Redskins       <NA>      <NA> 1499.999
## 41         St. Louis Bombers       <NA>      <NA> 1499.999
## 42           Toronto Huskies       <NA>      <NA> 1499.999
## 43      Washington Capitols       <NA>      <NA> 1499.999
```

```
## 44          Waterloo Hawks       <NA>      <NA> 1499.999
## 45       Baltimore Bullets       <NA>      <NA> 1499.999
```

Simulation of the 2018-2019 NBA Season

```
set.seed(37)

# Determine number of times to simulate the season
iterations = 10000

# Create data frame to hold simulation results
summary = data.frame(matrix(0, ncol = 6, nrow = nrow(team_info)))
colnames(summary) = c("team", "average_wins", "playoffs", "division_titles",
"conf_champ", "championships")
summary$team = team_info$team

# Create data frame to hold number of wins by each team in each iteration
histories = data.frame(matrix(0, ncol = nrow(team_info), nrow = iterations))
colnames(histories) = team_info$team

# Simulate the season the given number of times
for(i in 1:iterations) {
  #if(i %% 1000 == 0) {print(i)}
  season_stats = team_info[,which(colnames(team_info) != "inaugural_season")]
  season_stats$wins = 0
  season_stats$rand = runif(nrow(team_info))

  # Simulate each game in current season
  for(j in 1:nrow(season_schedule)) {
    # Find indices corresponding to home and away teams for current game
    home_index = which(season_stats$team == season_schedule$home_team[j])
    away_index = which(season_stats$team == season_schedule$away_team[j])

    # Find home and away team Elo ratings
    home_elo = season_stats$rating[home_index]
    away_elo = season_stats$rating[away_index]

    # Calculate home team win probability
    win_prob = 1 / (10^((away_elo - (home_elo +
hfa*season_schedule$neutral[j]))/400) + 1)
    u = runif(1)  # Generate a random number used to determine the winner of
the game

    # Determine which team wins the simulated game and increment their win
total by 1
    if(u < win_prob) {
      season_stats$wins[home_index] = season_stats$wins[home_index] + 1
    } else {
      season_stats$wins[away_index] = season_stats$wins[away_index] + 1
```

```
    }

    # Calculate actual margin of victory - must be positive
    score_diff = abs(season_schedule$home_score[j] -
season_schedule$away_score[j])

    # Determine home team result
    if(season_schedule$home_score[j] > season_schedule$away_score[j]) {
      home_result = 1  # Home team wins
    } else if(season_schedule$home_score[j] < season_schedule$away_score[j])
{
      home_result = 0  # Home team loses
    } else {
      home_result = 0.5  # Tie
    }

    # Calculate amount each team's Elo rating is adjusted by
    home_elo_adjustment = weight * log(score_diff + 1) * (home_result -
win_prob)

    # Adjust Elo ratings after game has been simulated to get team's new
strength
    season_stats$rating[home_index] = season_stats$rating[home_index] +
home_elo_adjustment
    season_stats$rating[away_index] = season_stats$rating[away_index] -
home_elo_adjustment
  }

  # Add number of wins for each team during this iteration to sum
  summary$average_wins = summary$average_wins + season_stats$wins

  # Define data frame that contains division winners
  division_winners = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(division_winners) = c("team", "conference", "division", "rating",
"wins", "rand")

  # Define data frame that contains non-division winners
  non_division_winners = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(non_division_winners) = c("team", "conference", "division",
"rating", "wins", "rand")

  # Define number of wild card teams per league and data frame that contains
wild card teams
  num_wild_cards = 5
  wild_card_teams = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(wild_card_teams) = c("team", "conference", "division", "rating",
"wins", "rand")

  # For each division
```

```
  for(div in divisions) {
    div_standings = season_stats[which(season_stats$division == div),]  # 
Identify all teams in current division
    div_standings = div_standings[order(-div_standings$wins, -
div_standings$rand),]  # Sort division by wins and random number
    division_winners = rbind(division_winners, div_standings[1,])  # Add 
division winner to 'division_winners' data frame
    non_division_winners = rbind(non_division_winners, 
div_standings[2:nrow(div_standings),])
        # Add non-division winners to 'non_division_winners' data frame
  }

  # For each conference/league
  for(conference in conferences) {
    wc_standings = non_division_winners[which(non_division_winners$conference 
== conference),]
        # Identify all non-division winners from the current conference
    wc_standings = wc_standings[order(-wc_standings$wins, -
wc_standings$rand),]  # Sort by wins and random number
    wild_card_teams = rbind(wild_card_teams, wc_standings[1:num_wild_cards,]) 
# Identify wild card teams from conference
  }

  # Sort division winners and wild card teams by conference, wins, and random 
number for seeding purposes
  division_winners = division_winners[order(division_winners$conference, -
division_winners$wins, -division_winners$rand),]
  wild_card_teams = wild_card_teams[order(wild_card_teams$conference, -
wild_card_teams$wins, -wild_card_teams$rand),]

  # Increment the number of division titles and playoff appearances for each 
division winner by 1
  for(team in division_winners$team) {
    index = which(season_stats$team == team)  # Index of division winner
    summary$playoffs[index] = summary$playoffs[index] + 1  # Increment 
playoff appearances
    summary$division_titles[index] = summary$division_titles[index] + 1  # 
Increment division titles
  }

  # Increment the number of playoff appearances for each wild card team by 1
  for(team in wild_card_teams$team) {
    index = which(season_stats$team == team)  # Index of wild card team
    summary$playoffs[index] = summary$playoffs[index] + 1  # Increment 
playoff appearances
  }

  # Create playoff bracket with every rating initialized to -Inf.  Must have 
a length that is a power of 2.
```

```
  playoff_bracket = data.frame(matrix(-Inf, ncol = 6, nrow = 16))
  colnames(playoff_bracket) = c("team", "league", "division", "rating",
"wins", "rand")
  next_round = NULL

  # NBA: playoffs do not consider division winners because it is seeded based
on wins only
  # These next several lines of code places the top 8 east teams and top 8
west teams into the bracket
  playoff_standings = season_stats[order(season_stats$conference, -
season_stats$wins, -season_stats$rand),]
  playoff_bracket[1,] = playoff_standings[1,]
  playoff_bracket[2,] = playoff_standings[2,]
  playoff_bracket[3,] = playoff_standings[3,]
  playoff_bracket[4,] = playoff_standings[4,]
  playoff_bracket[5,] = playoff_standings[5,]
  playoff_bracket[6,] = playoff_standings[6,]
  playoff_bracket[7,] = playoff_standings[7,]
  playoff_bracket[8,] = playoff_standings[8,]
  playoff_bracket[9,] = playoff_standings[16,]
  playoff_bracket[10,] = playoff_standings[17,]
  playoff_bracket[11,] = playoff_standings[18,]
  playoff_bracket[12,] = playoff_standings[19,]
  playoff_bracket[13,] = playoff_standings[20,]
  playoff_bracket[14,] = playoff_standings[21,]
  playoff_bracket[15,] = playoff_standings[22,]
  playoff_bracket[16,] = playoff_standings[23,]

  playoff_bracket$seed = rep(1:8,2)  # Append seed for each team in playoff
bracket
  games_per_round = c(7, 7, 7, 7)  # Specify number of games played in each
round of playoffs
  reseed = FALSE  # TRUE if reseed after first round; FALSE if not

  # Simulate every round in the playoffs until the championship game/round
  for(round in 1:(length(games_per_round)-1)) {
    for(j in 1:2) {  # Divide 'playoff_bracket' into two halves, separated by
conference
      for(k in 1:(nrow(playoff_bracket)/4)) {  # Match 1 seed with 8 seed, 2
seed with 7 seed, etc.
        high_seed_index = 0.5*nrow(playoff_bracket)*j-
(0.5*nrow(playoff_bracket)-k)
        low_seed_index = 0.5*nrow(playoff_bracket)*j-(k-1)

        # Obtain Elo ratings for high and low seeds
        high_seed_elo = playoff_bracket$rating[high_seed_index]
        low_seed_elo = playoff_bracket$rating[low_seed_index]

        # Calculate win probability for each team when they play at home
```

```
against their playoff opponent
        high_seed_home_win_prob = 1 / (10^((low_seed_elo - (high_seed_elo +
hfa))/400) + 1)
        low_seed_home_win_prob = 1 / (10^((high_seed_elo - (low_seed_elo +
hfa))/400) + 1)

        # Create array of win probabilities where high seed gets 1 more home
game than low seed
        win_probs = c(rep(high_seed_home_win_prob,
ceiling(games_per_round[round]/2)), 1-rep(low_seed_home_win_prob,
floor(games_per_round[round]/2)))

        u = runif(games_per_round[round])  # Generate random numbers for each
game in the round
        high_seed_wins = sum(u < win_probs)/games_per_round[round]  #
Calculate proportion of games won by higher seed

        if(high_seed_wins > 0.50) {  # If high seed won more than 50% of
games in series
          next_round = rbind(next_round, playoff_bracket[high_seed_index,])
# Advance high seed to next round
        } else{  # If low seed won more than 50% of games in series
          next_round = rbind(next_round, playoff_bracket[low_seed_index,])  #
Advance low seed to next round
        }
      }
    }

    playoff_bracket = next_round  # Reset playoff bracket to consist of all
remaining teams

    if(reseed) {  # Reseeds after each round
      playoff_bracket = playoff_bracket[order(playoff_bracket$league,
playoff_bracket$seed),]  # Reseed for next round
    } else {  # Do not reseed, but ensure higher seed in matchup gets home
court advantage
      if(nrow(playoff_bracket) >= 4) {  # If
        for(j in 1:2) {
          for(k in 1:(nrow(playoff_bracket)/4)) {
            index_1 = 0.5*nrow(playoff_bracket)*j-(0.5*nrow(playoff_bracket)-
k)
            index_2 = 0.5*nrow(playoff_bracket)*j-(k-1)
            if(playoff_bracket$seed[index_1] > playoff_bracket$seed[index_2])
{
              temp = playoff_bracket[index_1,]
              playoff_bracket[index_1,] = playoff_bracket[index_2,]
              playoff_bracket[index_2,] = temp
            }
          }
```

```
        }
      }
    }
    next_round = NULL  # Reset list of teams in subsequent round to an empty
data frame
  }

  # Sorting playoff bracket
  playoff_bracket = playoff_bracket[order(-playoff_bracket$wins, -
playoff_bracket$rand),]

  # Repeat above process of finding Elo ratings of teams in championship,
calculating win probability, and simulating round
  high_seed_elo = playoff_bracket$rating[1]
  low_seed_elo = playoff_bracket$rating[2]
  high_seed_home_win_prob = 1 / (10^((low_seed_elo - (high_seed_elo +
hfa))/400) + 1)
  low_seed_home_win_prob = 1 / (10^((high_seed_elo - (low_seed_elo +
hfa))/400) + 1)
  win_probs = c(rep(high_seed_home_win_prob,
ceiling(games_per_round[length(games_per_round)]/2)), 1-
rep(low_seed_home_win_prob,
floor(games_per_round[length(games_per_round)]/2)))
  u = runif(games_per_round[length(games_per_round)])
  high_seed_wins = sum(u <
win_probs)/games_per_round[length(games_per_round)]

  if(high_seed_wins > 0.50) {  # High seed wins championship
    champion = playoff_bracket[1,]
  } else{  # Low seed wins championship
    champion = playoff_bracket[2,]
  }

  # Increment number of conference championships/pennants won by each team by
1
  for(team in playoff_bracket$team) {
    index = which(season_stats$team == team)
    summary$conf_champ[index] = summary$conf_champ[index] + 1
  }

  # Increment number of championships won by 1
  index = which(season_stats$team == champion$team)
  summary$championships[index] = summary$championships[index] + 1
  histories[i,] = season_stats$wins
}

# Calculate average number of wins across all iterations
summary$average_wins = summary$average_wins/iterations
summary
```

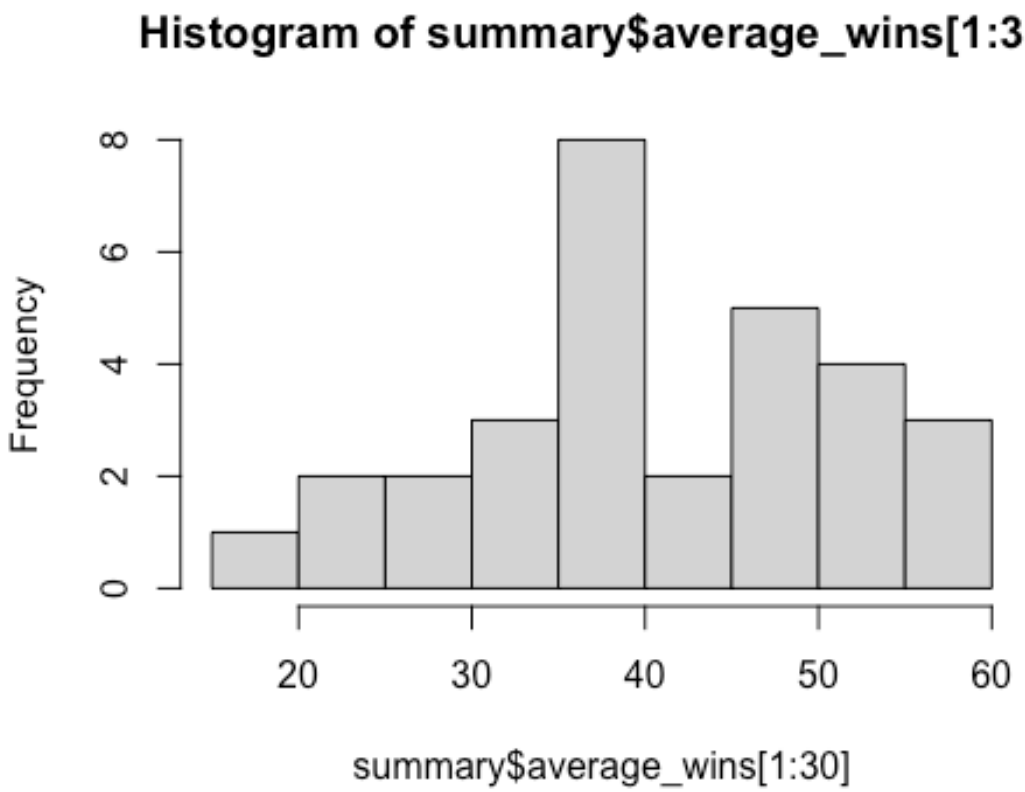
```
##                       team average_wins playoffs division_titles
conf_champ
## 1             Brooklyn Nets      35.0877     2727               0
9
## 2         Charlotte Hornets      39.2996     6691            3675
22
## 3             Chicago Bulls      24.0629       11               0
0
## 4       Cleveland Cavaliers      28.5243      172               0
0
## 5          Dallas Mavericks      32.7403       91               0
0
## 6            Denver Nuggets      49.8984     9477            2324
529
## 7           Houston Rockets      52.6039     9868            8517
3560
## 8            Indiana Pacers      49.9727     9968            1576
223
## 9      Los Angeles Clippers      44.4396     6463             111
68
## 10        Memphis Grizzlies      30.8549       28               0
0
## 11               Miami Heat      39.7064     7028            3946
20
## 12          Milwaukee Bucks      55.8678    10000            8408
4838
## 13  Minnesota Timberwolves      39.9211     2254              21
2
## 14    New Orleans Pelicans      42.2500     4260             325
0
## 15   Oklahoma City Thunder      49.3921     9364            2037
316
## 16           Orlando Magic      33.9625     1910             535
39
## 17             Phoenix Suns      19.6572        0               0
0
## 18   Portland Trailblazers      49.6449     9396            2176
1420
## 19        San Antonio Spurs      45.5700     7378            1158
148
## 20          Toronto Raptors      56.4640    10000            7119
3524
## 21                Utah Jazz      51.1674     9723            3442
1090
## 22      Washington Wizards      37.1502     4597            1840
1
## 23           Boston Celtics      50.3372     9974            1116
617
## 24   Golden State Warriors      57.9511     9998            9888
2866
```

```
## 25         New York Knicks      23.1107        1                0
0
## 26      Philadelphia 76ers      51.5578     9991             1765
682
## 27           Atlanta Hawks      25.4490       21                4
0
## 28         Detroit Pistons      39.6274     6909               16
25
## 29      Los Angeles Lakers      38.3882     1366                1
0
## 30        Sacramento Kings      35.3407      334                0
1
## 31        Anderson Packers       0.0000        0                0
0
## 32           Chicago Stags       0.0000        0                0
0
## 33        Cleveland Rebels       0.0000        0                0
0
## 34    Denver Nuggets (1st)       0.0000        0                0
0
## 35         Detroit Falcons       0.0000        0                0
0
## 36       Indianapolis Jets       0.0000        0                0
0
## 37  Indianapolis Olympians       0.0000        0                0
0
## 38      Pittsburgh Ironmen       0.0000        0                0
0
## 39 Providence Steamrollers       0.0000        0                0
0
## 40      Sheboygan Redskins       0.0000        0                0
0
## 41       St. Louis Bombers       0.0000        0                0
0
## 42         Toronto Huskies       0.0000        0                0
0
## 43     Washington Capitols       0.0000        0                0
0
## 44          Waterloo Hawks       0.0000        0                0
0
## 45       Baltimore Bullets       0.0000        0                0
0
##    championships
## 1              1
## 2              1
## 3              0
## 4              0
## 5              0
## 6            181
## 7           2171
```

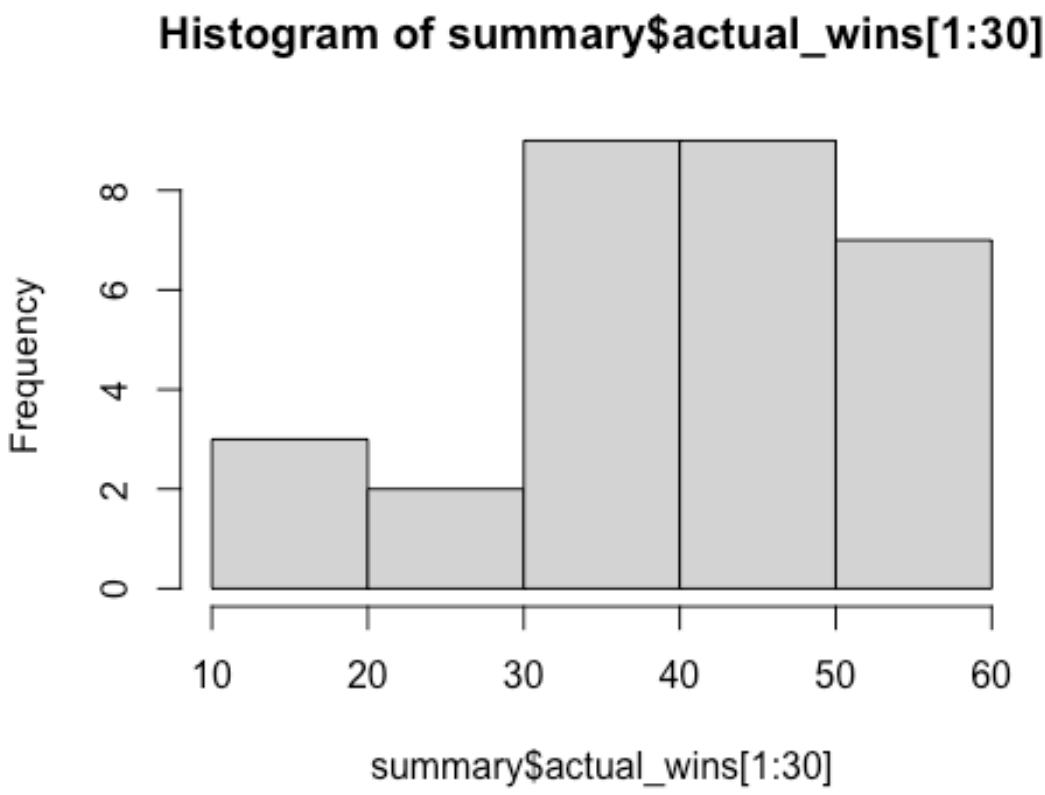
```
## 8             46
## 9             12
## 10             0
## 11             2
## 12          2669
## 13             0
## 14             0
## 15            99
## 16             4
## 17             0
## 18           660
## 19            37
## 20          1724
## 21           448
## 22             0
## 23           157
## 24          1620
## 25             0
## 26           166
## 27             0
## 28             2
## 29             0
## 30             0
## 31             0
## 32             0
## 33             0
## 34             0
## 35             0
## 36             0
## 37             0
## 38             0
## 39             0
## 40             0
## 41             0
## 42             0
## 43             0
## 44             0
## 45             0
```

```
# Inputting actual wins for teams
team_info$actual_wins =
c(42,39,22,19,33,54,53,48,48,33,39,60,36,33,49,42,19,53,48,58,50,32,49,57,17,
51,29,41,37,39,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

# Copying actual wins into team_info data frame
summary$actual_wins = team_info$actual_wins
summary$difference = summary$actual_wins - summary$average_wins

# Getting final season Elo ratings
final_elos = season_stats
```

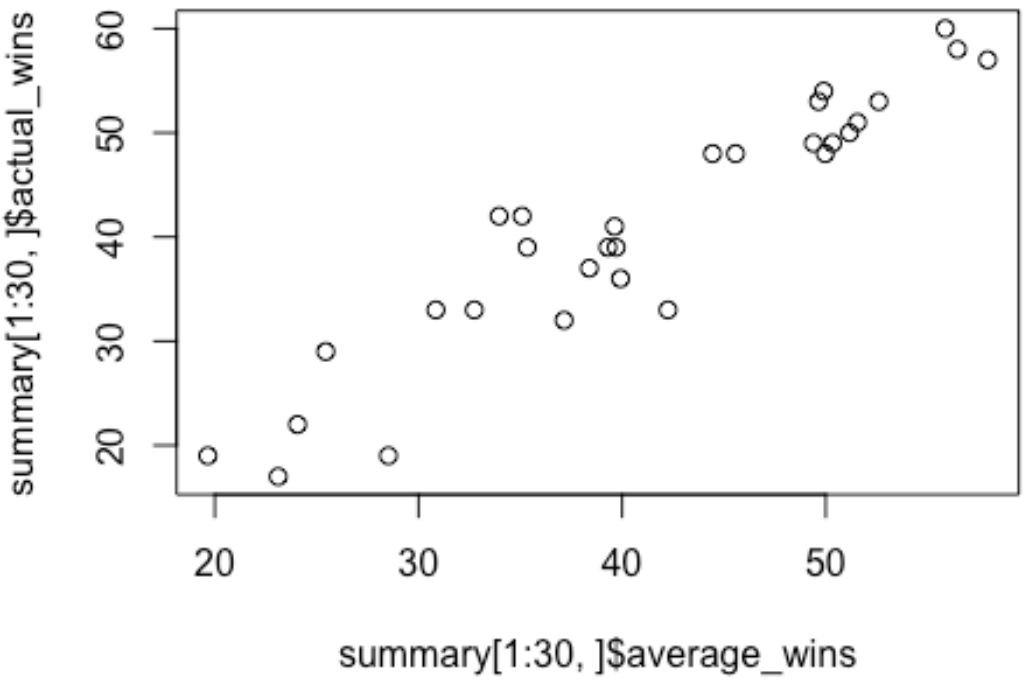
```r
# Histogram of simulated wins
hist(summary$average_wins[1:30])
```



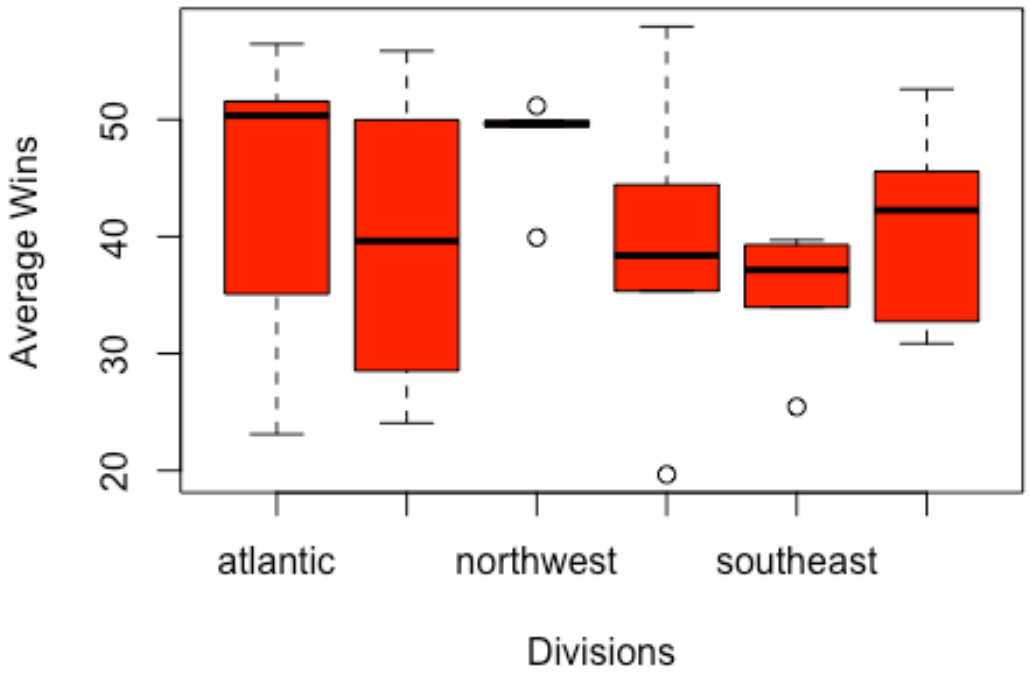

```r
# Histogram of actual wins
hist(summary$actual_wins[1:30])
```



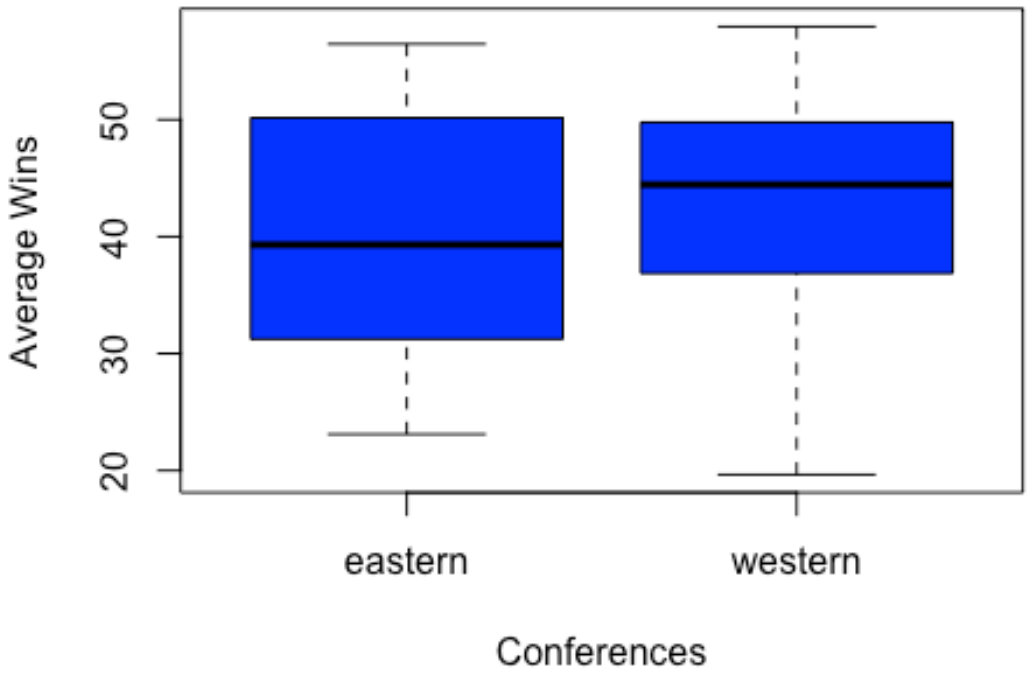

```r
# Scatter plot and correlation of actual vs. simulated
plot(summary[1:30,]$average_wins, summary[1:30,]$actual_wins)
```



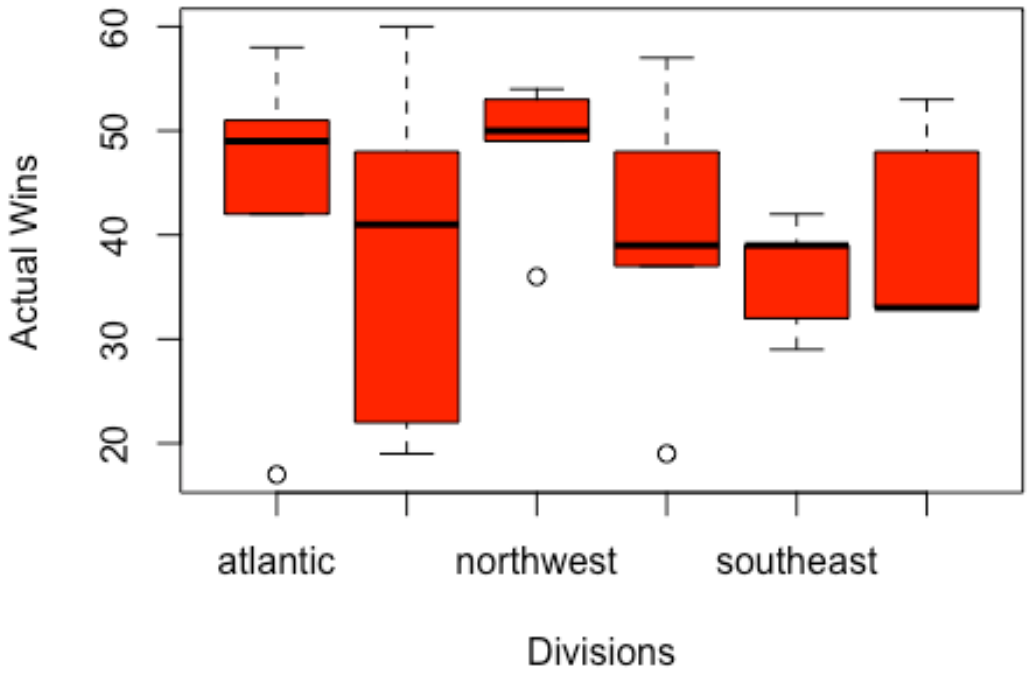

```
cor(summary[1:30,]$average_wins, summary[1:30,]$actual_wins)

## [1] 0.9424123

# Side by side box plots of simulated win distributions for teams in same
division and conference
boxplot(summary$average_wins ~ team_info$division, col="red",
xlab="Divisions", ylab="Average Wins")
```



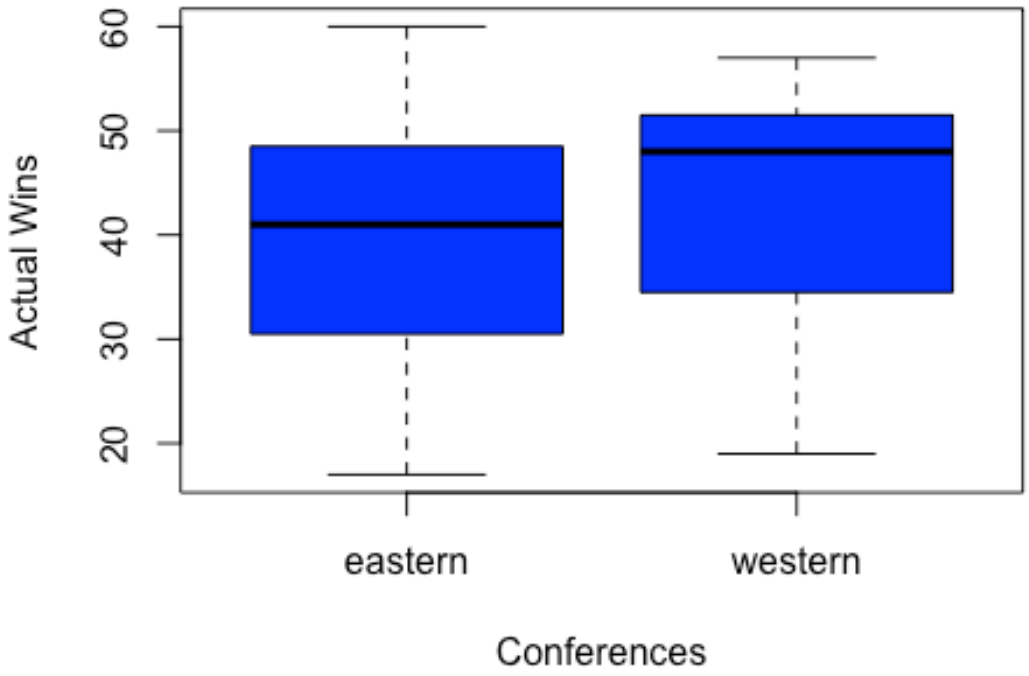

```
boxplot(summary$average_wins ~ team_info$conference, col="blue",
xlab="Conferences", ylab="Average Wins")
```




```
# Side by side box plots of actual win distributions for teams in same
division and conference
boxplot(summary$actual_wins ~ team_info$division, col="red",
xlab="Divisions", ylab="Actual Wins")
```




```
boxplot(summary$actual_wins ~ team_info$conference, col="blue",
xlab="Conferences", ylab="Actual Wins")
```

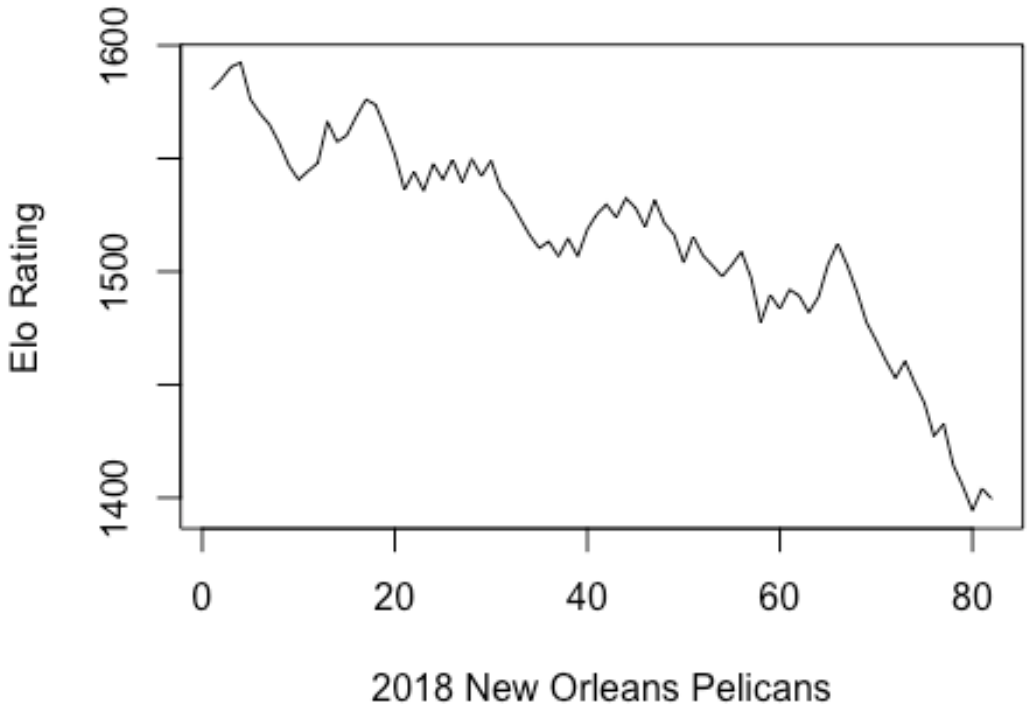


Tracking Elo rating for Underachieving Team: Pelicans

```
scores = read.table('nba_scores.csv', header=TRUE, sep = ',')
elos = read.table('nba_initial_elos.csv', header=TRUE, sep=',')

weight = 8.7
hfa = 78.79

# Select team and season to follow for a period of time
team = "New Orleans Pelicans"
first_season = 2018
last_season = 2018

# Create data frame to store information for team specified above
team_results = data.frame(matrix(ncol = 8, nrow = 0))
colnames(team_results) = c("opponent", "pregame_elo", "win_probability",
"result", "team_score", "opponent_score", "elo_adjustment", "postgame_elo")

# Iterate through all games in the sport's history
for(i in 1:nrow(scores)) {
  # Find indices corresponding to home and away teams for current game
  home_index = which(elos$team == scores$home_team[i])
```

```
  away_index = which(elos$team == scores$away_team[i])

  # Find home and away team Elo ratings
  home_elo = elos$rating[home_index]
  away_elo = elos$rating[away_index]

  # Calculate home team win probability
  win_prob = 1 / (10^((away_elo - (home_elo + hfa*scores$neutral[i]))/400) +
1)

  # Calculate actual margin of victory - must be positive
  score_diff = abs(scores$home_score[i] - scores$away_score[i])

  # Determine home team result
  if(scores$home_score[i] > scores$away_score[i]) {
    home_result = 1  # Home team wins
  } else if(scores$home_score[i] < scores$away_score[i]) {
    home_result = 0  # Home team loses
  } else {
    home_result = 0.5  # Tie
  }

  # Calculate amount each team's Elo rating is adjusted by
  home_elo_adjustment = weight * log(score_diff + 1) * (home_result -
win_prob)

  # Adjust Elo ratings - add point to winner and subtract points from loser
  elos$rating[home_index] = elos$rating[home_index] + home_elo_adjustment
  elos$rating[away_index] = elos$rating[away_index] - home_elo_adjustment

  # Add game information to team result data frame for each team game of the
team specified above if team and season both match
  if(scores$season[i] >= first_season & scores$season[i] <= last_season &
(scores$home_team[i] == team | scores$away_team[i] == team)) {
    if(scores$home_team[i] == team) {  # If specified team was at home
      team_results[nrow(team_results) + 1,] = c(scores$away_team[i],
elos$rating[home_index] - home_elo_adjustment, win_prob, home_result,
scores$home_score[i], scores$away_score[i], home_elo_adjustment,
elos$rating[home_index])
    } else {  # If specified team was away
      team_results[nrow(team_results) + 1,] = c(scores$home_team[i],
elos$rating[away_index] + home_elo_adjustment, 1-win_prob, 1-home_result,
scores$away_score[i], scores$home_score[i], -1*home_elo_adjustment,
elos$rating[away_index])
    }
  }

  # Adjust Elo ratings at end of season to regress 1/3 of the way towards
1500
```

```r
  if(i < nrow(scores) && scores$season[i+1] > scores$season[i]) {  # New season
    for(j in 1:nrow(elos)) {  # For each team
      if(scores$season[i] >= elos$inaugural_season[j]) {  # Check if team existed
        # Move each team's Elo rating back towards 1500 by 1/3 of the difference
        elos$rating[j] = elos$rating[j] - (elos$rating[j] - 1500)/3
      }
    }

    # Identify all teams that existed at beginning of following season
    existing_teams = elos[which(elos$inaugural_season <= (scores$season[i] + 1)),]

    # Calculate amount each team's Elo rating must be adjusted by to make mean 1500
    expansion_adjustment = -1*(mean(existing_teams$rating) - 1500)

    # Perform expansion adjustment on teams that existed at beginning of following season
    for(j in 1:nrow(elos)) {  # For each team
      if((scores$season[i] + 1) >= elos$inaugural_season[j]) {  # Check if team existed
        elos$rating[j] = elos$rating[j] + expansion_adjustment  # Update ratings if so
      }
    }
  }
}
# Graph of Pelicans Elo rating throughout the season
plot(team_results$postgame_elo, type = "l", xlab = paste(first_season, team), ylab = "Elo Rating")
```

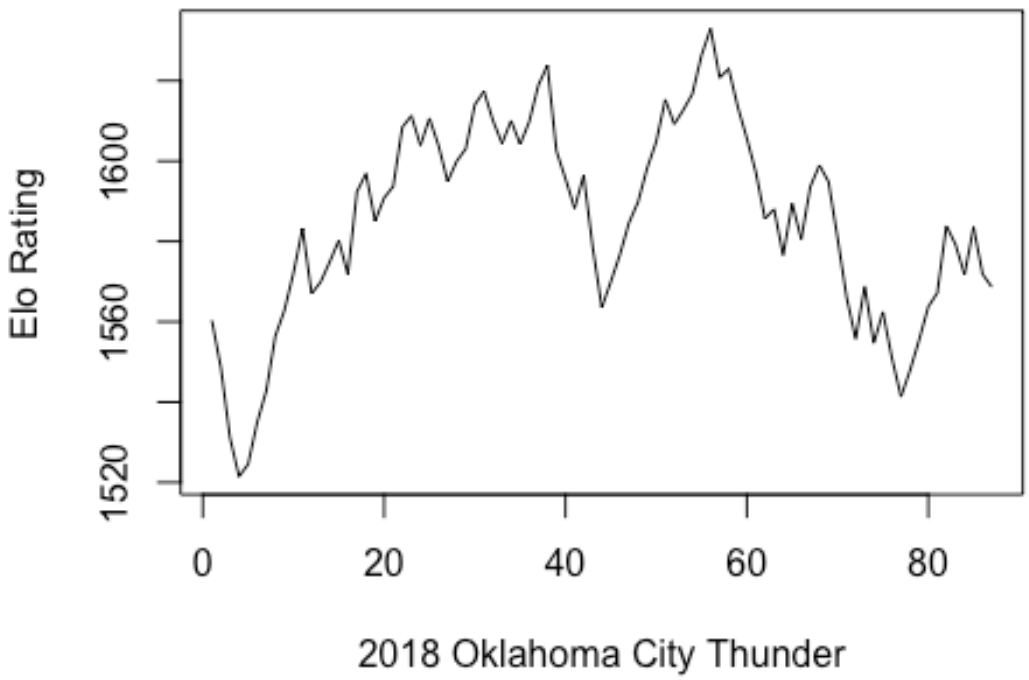


Tracking Elo rating for Average Team: Oklahoma City Thunder

```
scores = read.table('nba_scores.csv', header=TRUE, sep = ',')
elos = read.table('nba_initial_elos.csv', header=TRUE, sep=',')

weight = 8.7
hfa = 78.79

# Select team and season to follow for a period of time
team = "Oklahoma City Thunder"
first_season = 2018
last_season = 2018

# Create data frame to store information for team specified above
team_results = data.frame(matrix(ncol = 8, nrow = 0))
colnames(team_results) = c("opponent", "pregame_elo", "win_probability",
"result", "team_score", "opponent_score", "elo_adjustment", "postgame_elo")

# Iterate through all games in the sport's history
for(i in 1:nrow(scores)) {
  # Find indices corresponding to home and away teams for current game
  home_index = which(elos$team == scores$home_team[i])
```

```
  away_index = which(elos$team == scores$away_team[i])

  # Find home and away team Elo ratings
  home_elo = elos$rating[home_index]
  away_elo = elos$rating[away_index]

  # Calculate home team win probability
  win_prob = 1 / (10^((away_elo - (home_elo + hfa*scores$neutral[i]))/400) +
1)

  # Calculate actual margin of victory - must be positive
  score_diff = abs(scores$home_score[i] - scores$away_score[i])

  # Determine home team result
  if(scores$home_score[i] > scores$away_score[i]) {
    home_result = 1  # Home team wins
  } else if(scores$home_score[i] < scores$away_score[i]) {
    home_result = 0  # Home team loses
  } else {
    home_result = 0.5  # Tie
  }

  # Calculate amount each team's Elo rating is adjusted by
  home_elo_adjustment = weight * log(score_diff + 1) * (home_result -
win_prob)

  # Adjust Elo ratings - add point to winner and subtract points from loser
  elos$rating[home_index] = elos$rating[home_index] + home_elo_adjustment
  elos$rating[away_index] = elos$rating[away_index] - home_elo_adjustment

  # Add game information to team result data frame for each team game of the
team specified above if team and season both match
  if(scores$season[i] >= first_season & scores$season[i] <= last_season &
(scores$home_team[i] == team | scores$away_team[i] == team)) {
    if(scores$home_team[i] == team) {  # If specified team was at home
      team_results[nrow(team_results) + 1,] = c(scores$away_team[i],
elos$rating[home_index] - home_elo_adjustment, win_prob, home_result,
scores$home_score[i], scores$away_score[i], home_elo_adjustment,
elos$rating[home_index])
    } else {  # If specified team was away
      team_results[nrow(team_results) + 1,] = c(scores$home_team[i],
elos$rating[away_index] + home_elo_adjustment, 1-win_prob, 1-home_result,
scores$away_score[i], scores$home_score[i], -1*home_elo_adjustment,
elos$rating[away_index])
    }
  }

  # Adjust Elo ratings at end of season to regress 1/3 of the way towards
1500
```

```r
  if(i < nrow(scores) && scores$season[i+1] > scores$season[i]) {  # New season
    for(j in 1:nrow(elos)) {  # For each team
      if(scores$season[i] >= elos$inaugural_season[j]) {  # Check if team existed
        # Move each team's Elo rating back towards 1500 by 1/3 of the difference
        elos$rating[j] = elos$rating[j] - (elos$rating[j] - 1500)/3
      }
    }

    # Identify all teams that existed at beginning of following season
    existing_teams = elos[which(elos$inaugural_season <= (scores$season[i] + 1)),]

    # Calculate amount each team's Elo rating must be adjusted by to make mean 1500
    expansion_adjustment = -1*(mean(existing_teams$rating) - 1500)

    # Perform expansion adjustment on teams that existed at beginning of following season
    for(j in 1:nrow(elos)) {  # For each team
      if((scores$season[i] + 1) >= elos$inaugural_season[j]) {  # Check if team existed
        elos$rating[j] = elos$rating[j] + expansion_adjustment  # Update ratings if so
      }
    }
  }
}
# Graph of Thunder Elo rating throughout the season
plot(team_results$postgame_elo, type = "l", xlab = paste(first_season, team), ylab = "Elo Rating")
```

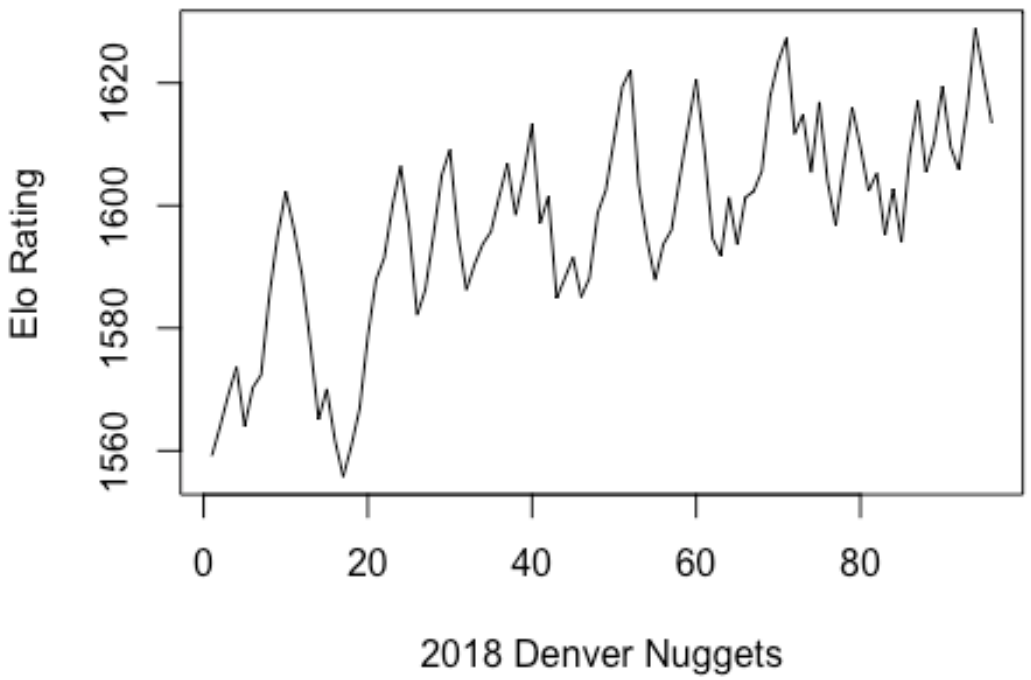


Tracking Elo rating for Overachieving Team: Denver Nuggets

```
scores = read.table('nba_scores.csv', header=TRUE, sep = ',')
elos = read.table('nba_initial_elos.csv', header=TRUE, sep=',')

weight = 8.7
hfa = 78.79

# Select team and season to follow for a period of time
team = "Denver Nuggets"
first_season = 2018
last_season = 2018

# Create data frame to store information for team specified above
team_results = data.frame(matrix(ncol = 8, nrow = 0))
colnames(team_results) = c("opponent", "pregame_elo", "win_probability",
"result", "team_score", "opponent_score", "elo_adjustment", "postgame_elo")

# Iterate through all games in the sport's history
for(i in 1:nrow(scores)) {
  # Find indices corresponding to home and away teams for current game
  home_index = which(elos$team == scores$home_team[i])
```

```
  away_index = which(elos$team == scores$away_team[i])

  # Find home and away team Elo ratings
  home_elo = elos$rating[home_index]
  away_elo = elos$rating[away_index]

  # Calculate home team win probability
  win_prob = 1 / (10^((away_elo - (home_elo + hfa*scores$neutral[i]))/400) +
1)

  # Calculate actual margin of victory - must be positive
  score_diff = abs(scores$home_score[i] - scores$away_score[i])

  # Determine home team result
  if(scores$home_score[i] > scores$away_score[i]) {
    home_result = 1  # Home team wins
  } else if(scores$home_score[i] < scores$away_score[i]) {
    home_result = 0  # Home team loses
  } else {
    home_result = 0.5  # Tie
  }

  # Calculate amount each team's Elo rating is adjusted by
  home_elo_adjustment = weight * log(score_diff + 1) * (home_result -
win_prob)

  # Adjust Elo ratings - add point to winner and subtract points from loser
  elos$rating[home_index] = elos$rating[home_index] + home_elo_adjustment
  elos$rating[away_index] = elos$rating[away_index] - home_elo_adjustment

  # Add game information to team result data frame for each team game of the
team specified above if team and season both match
  if(scores$season[i] >= first_season & scores$season[i] <= last_season &
(scores$home_team[i] == team | scores$away_team[i] == team)) {
    if(scores$home_team[i] == team) {  # If specified team was at home
      team_results[nrow(team_results) + 1,] = c(scores$away_team[i],
elos$rating[home_index] - home_elo_adjustment, win_prob, home_result,
scores$home_score[i], scores$away_score[i], home_elo_adjustment,
elos$rating[home_index])
    } else {  # If specified team was away
      team_results[nrow(team_results) + 1,] = c(scores$home_team[i],
elos$rating[away_index] + home_elo_adjustment, 1-win_prob, 1-home_result,
scores$away_score[i], scores$home_score[i], -1*home_elo_adjustment,
elos$rating[away_index])
    }
  }

  # Adjust Elo ratings at end of season to regress 1/3 of the way towards
1500
```

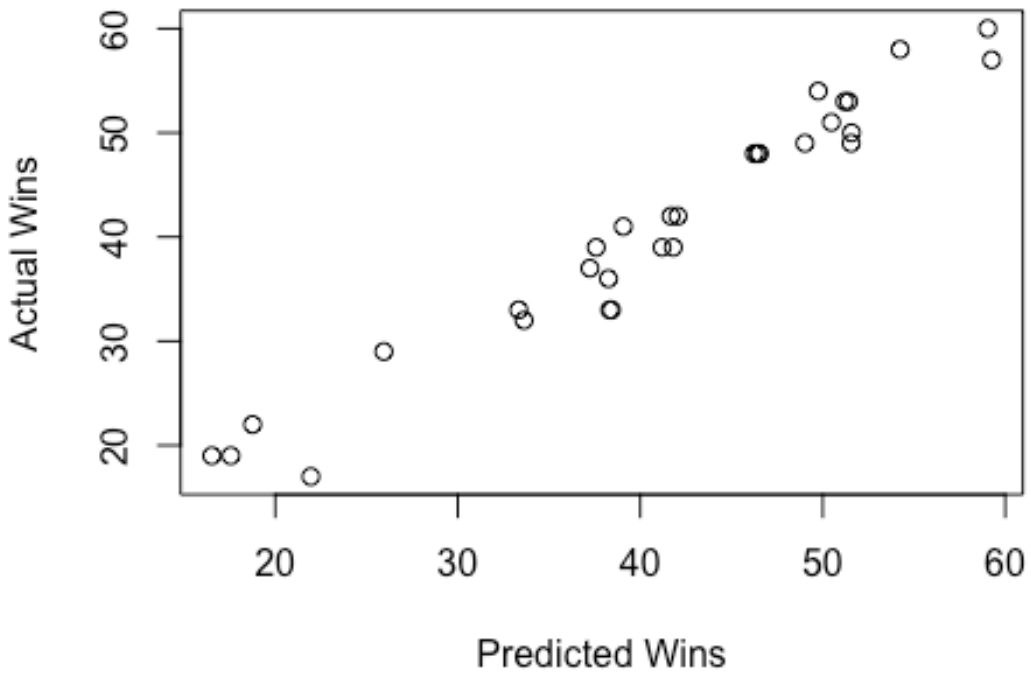
```r
  if(i < nrow(scores) && scores$season[i+1] > scores$season[i]) {  # New
season
    for(j in 1:nrow(elos)) {  # For each team
      if(scores$season[i] >= elos$inaugural_season[j]) {  # Check if team
existed
        # Move each team's Elo rating back towards 1500 by 1/3 of the
difference
        elos$rating[j] = elos$rating[j] - (elos$rating[j] - 1500)/3
      }
    }

    # Identify all teams that existed at beginning of following season
    existing_teams = elos[which(elos$inaugural_season <= (scores$season[i] +
1)),]

    # Calculate amount each team's Elo rating must be adjusted by to make
mean 1500
    expansion_adjustment = -1*(mean(existing_teams$rating) - 1500)

    # Perform expansion adjustment on teams that existed at beginning of
following season
    for(j in 1:nrow(elos)) {  # For each team
      if((scores$season[i] + 1) >= elos$inaugural_season[j]) {  # Check if
team existed
        elos$rating[j] = elos$rating[j] + expansion_adjustment  # Update
ratings if so
      }
    }
  }
}
# Graph of Nuggets Elo rating throughout the season
plot(team_results$postgame_elo, type = "l", xlab = paste(first_season, team),
ylab = "Elo Rating")
```



Relevant Statistics: Four Factors in Basketball

```
# Build a model using the four factors in basketball to predict win totals
and compare against simulated win total
# Using readxl package to read excel file from basketball-reference.com
require(readxl)

## Loading required package: readxl

df = read_excel("nba_team_stats.xls")

# Fit a multiple regression model that regresses the team's winning
percentage on the offensive and defensive four factors (8 predictors in
total)
model = lm(W ~ 0 + `eFG%` + `TOV%` + `ORB%` + `FT/FGA` + `Opp eFG%` + `Opp
TOV%` + `DRB%` + `Opp FT/FGA`, data = df)
summary(model)

##
## Call:
## lm(formula = W ~ 0 + `eFG%` + `TOV%` + `ORB%` + `FT/FGA` + `Opp eFG%` +
##     `Opp TOV%` + `DRB%` + `Opp FT/FGA`, data = df)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4227 -2.0544  0.4189  1.7116  4.2466
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## `eFG%`         383.4149    34.6523  11.065 1.86e-10 ***
## `TOV%`          -5.1166     0.8936  -5.726 9.27e-06 ***
## `ORB%`           1.3785     0.3098   4.449 0.000201 ***
## `FT/FGA`        74.1521    31.2434   2.373 0.026784 *
## `Opp eFG%`    -404.3042    29.2111 -13.841 2.45e-12 ***
## `Opp TOV%`       1.7327     0.7015   2.470 0.021752 *
## `DRB%`           0.6208     0.2754   2.254 0.034474 *
## `Opp FT/FGA`    -0.7841    50.3830  -0.016 0.987723
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.998 on 22 degrees of freedom
## Multiple R-squared:  0.9964, Adjusted R-squared:  0.9951
## F-statistic: 756.9 on 8 and 22 DF,  p-value: < 2.2e-16

# Saving predicted win probabilities and residuals to the data frame
df$predicted = predict(model, df)
df$resid = resid(model)

# Displaying the three teams analyzed
three_teams = subset(df, df$Team == "New Orleans Pelicans" | df$Team ==
"Oklahoma City Thunder" | df$Team == "Denver Nuggets")

# Scatter plot of actual win percentage (Y-axis) against predicted win
percentage based on the model
plot(df$predicted, df$W, xlab = "Predicted Wins", ylab = "Actual Wins")
```




```
cor(df$predicted, df$W)
```

```
## [1] 0.9761532
```