Monte Carlo Simulation of the 1992-93 NHL Season

Chase Lane

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## Warning in library(package, lib.loc = lib.loc, character.only =
TRUE,
## logical.return = TRUE, : there is no package called 'tidyverse'
```

```
require(stringr)
```

```
## Loading required package: stringr
```

Simulation of the 1992-93 NHL Season

```
#Import updated initial elos file which contains correct conferences
and teams
#Import all NHL games

scores <- read.table("nhl_scores.csv", header=TRUE, sep=",")
elos <- read.table("nhl_initial_elos1993.csv", header=TRUE, sep=",")

#1992-93 Season (Stored as 1992 in dataset)
simulated_season = 1992

#Select all games prior to the season we want simulated
pre_season = scores[which(scores$season < simulated_season &
scores$season >= 1901),]
```

```r
#Select only games in simulated season specifically regular season
games denoted by "r"
season_schedule = scores[which(scores$season == simulated_season &
scores$game_type == "r"),]
#Correct number of games played as each team played a total of 84
games

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

#Adjusting team names in dataset and doing this after obtaining
correct years from original dataset because there would be conflicting
Winnipeg Jets names
#Carolina Hurricanes -> Hartford Whalers
#Colorado Avalance -> Quebec Nordiques
#Dallas Stars -> Minnesota North Stars
#Arizona Coyotes -> Winnipeg Jets
pre_season$home_team <- gsub("Carolina Hurricanes", "Hartford
Whalers", pre_season$home_team)
pre_season$home_team <- gsub("Colorado Avalanche", "Quebec Nordiques",
pre_season$home_team)
pre_season$home_team <- gsub("Dallas Stars", "Minnesota North Stars",
pre_season$home_team)
pre_season$home_team <- gsub("Arizona Coyotes", "Winnipeg Jets",
pre_season$home_team)
pre_season$away_team <- gsub("Carolina Hurricanes", "Hartford
Whalers", pre_season$away_team)
pre_season$away_team <- gsub("Colorado Avalanche", "Quebec Nordiques",
pre_season$away_team)
pre_season$away_team <- gsub("Dallas Stars", "Minnesota North Stars",
pre_season$away_team)
pre_season$away_team <- gsub("Arizona Coyotes", "Winnipeg Jets",
pre_season$away_team)

season_schedule$home_team <- gsub("Carolina Hurricanes", "Hartford
Whalers", season_schedule$home_team)
season_schedule$home_team <- gsub("Colorado Avalanche", "Quebec
Nordiques", season_schedule$home_team)
season_schedule$home_team <- gsub("Dallas Stars", "Minnesota North
Stars", season_schedule$home_team)
season_schedule$home_team <- gsub("Arizona Coyotes", "Winnipeg Jets",
season_schedule$home_team)
season_schedule$away_team <- gsub("Carolina Hurricanes", "Hartford
Whalers", season_schedule$away_team)
season_schedule$away_team <- gsub("Colorado Avalanche", "Quebec
Nordiques", season_schedule$away_team)
season_schedule$away_team <- gsub("Dallas Stars", "Minnesota North
Stars", season_schedule$away_team)
```

```
season_schedule$away_team <- gsub("Arizona Coyotes", "Winnipeg Jets",
season_schedule$away_team)

#Calculating home field advantage
home_wins = 0
games = 0
first_game_index = 9308 #1967-68 season removes extreme early results

# Iterate through games - first index can be changed to eliminate
early seasons where scores are extreme
for(i in first_game_index:nrow(scores)) {
  # Count number of games that do not end in ties
  if(scores$home_score[i] != scores$away_score[i]) { games = games + 1
}

  # Count number of games where home team wins
  if(scores$home_score[i] > scores$away_score[i]) { home_wins =
home_wins + 1 }
}

home_win_prob = home_wins / games  # Calculate home win probability
where outcome was not a tie
hfa = -400*log10(1/home_win_prob - 1)  # Calculate number of Elo
points added to home team
cat("HFA = ", hfa)

## HFA =  51.63396

#Calculate optimal k-value

# Iterate through all potential k values that are being tested
starting_weight = 6.8 # Lower bound for weight ranges to be tested -
generally set equal to 0
iterations = 10  # Number of k values to test
step_size = 0.1 # Amount to increment k by at each step


# Initialize data frame to store k values and corresponding error
errors = data.frame(matrix(ncol = 2, nrow = iterations))
colnames(errors) = c("weight", "error")
errors$weight = starting_weight + (1:iterations)*step_size
errors$error = NA
scores <- read.table("nhl_scores.csv", header=TRUE, sep=",")
# Iterate through all potential k values that are being tested
# Iterate through all potential k values that are being tested
for(counter in 1:iterations) {
  weight = starting_weight + counter*step_size  # Calculate k value
for current iteration
  error = 0  # Reset error for current iteration
  elos = read.table("nhl_initial_elos.csv", header=TRUE, sep=",")  #
```

```
Reset initial Elo ratings

  # Iterate through games - first index can be changed to eliminate
early seasons in a league where early results tend to be extreme
  for(i in first_game_index: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
    }

    # Add squared error between home result and predicted probability
of home team winning to SSE
    error = error + (home_result - win_prob)^2

    # 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

    # 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(elos)) {
        if(scores$season[i] >= elos$inaugural_season[j]) {
```

```
          elos$rating[j] = elos$rating[j] - (elos$rating[j] - 1500)/3
        }
      }

      existing_teams = elos[which(elos$inaugural_season <=
(scores$season[i] + 1)),]
      expansion_adjustment = -1*(mean(existing_teams$rating) - 1500)

      for(j in 1:nrow(elos)) {
        if((scores$season[i] + 1) >= elos$inaugural_season[j]) {
          elos$rating[j] = elos$rating[j] + expansion_adjustment
        }
      }
    }
  }
  errors$error[counter] = error  # Store error for current iteration
}

weight = errors$weight[which(errors$error == min(errors$error))]
cat("Optimal k-value = ", weight)

## Optimal k-value =  7.1

#Calculate pre season Elo Ratings
hfa = 51.63396
weight = 7.1

team_info = read.table("nhl_initial_elos1993.csv", header=TRUE,
sep=",")
# 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
      }
    }
  }
}

team_info = team_info[which(team_info$conference != 'NA'),]
team_info[order(-team_info$rating),]
```

```
##                     team        conference division   rating
inaugural_season
## 9    Pittsburgh Penguins   Prince of Wales  Patrick 1561.327
1967
## 8       New York Rangers   Prince of Wales  Patrick 1559.345
1926
## 15    Chicago Blackhawks Clarence Campbell   Norris 1550.520
1926
## 12   Washington Capitals   Prince of Wales  Patrick 1539.122
1974
## 2     Montreal Canadiens   Prince of Wales    Adams 1536.085
1917
## 4      Detroit Red Wings Clarence Campbell   Norris 1532.676
1926
## 13     New Jersey Devils   Prince of Wales  Patrick 1525.655
1974
## 17       St. Louis Blues Clarence Campbell   Norris 1522.265
1967
## 19     Los Angeles Kings Clarence Campbell   Smythe 1517.717
1967
## 20     Vancouver Canucks Clarence Campbell   Smythe 1517.465
1970
## 21        Calgary Flames Clarence Campbell   Smythe 1514.964
1972
## 3          Boston Bruins   Prince of Wales    Adams 1514.132
1924
## 23       Edmonton Oilers Clarence Campbell   Smythe 1511.364
1979
## 5         Buffalo Sabres   Prince of Wales    Adams 1504.632
1970
## 11    New York Islanders   Prince of Wales  Patrick 1495.179
1972
## 22         Winnipeg Jets Clarence Campbell   Smythe 1494.799
1979
## 10   Philadelphia Flyers   Prince of Wales  Patrick 1494.556
1967
## 16 Minnesota North Stars Clarence Campbell   Norris 1484.594
1967
## 14      Hartford Whalers   Prince of Wales    Adams 1473.218
1979
## 1    Toronto Maple Leafs Clarence Campbell   Norris 1462.881
1917
## 18      Quebec Nordiques   Prince of Wales    Adams 1423.990
1979
## 6        Ottawa Senators   Prince of Wales    Adams 1406.667
1992
## 7    Tampa Bay Lightning Clarence Campbell   Norris 1406.667
1992
## 24       San Jose Sharks Clarence Campbell   Smythe 1395.606
1991
```

```
##    points
## 9     119
## 8      79
## 15    106
## 12     93
## 2     102
## 4     103
## 13     87
## 17     85
## 19     88
## 20    101
## 21     97
## 3     109
## 23     60
## 5      86
## 11     87
## 22     87
## 10     83
## 16     82
## 14     58
## 1      99
## 18    104
## 6      24
## 7      53
## 24     24
```

```r
#Simulating actual season with a seed of 45
set.seed(45)
#Number of iterations
iterations = 10000

#Omit teams not in league anymore, create data frames to hold results
team_info = team_info[which(team_info$conference != 'NA'),]
summary = data.frame(matrix(0, ncol = 6, nrow = nrow(team_info)))
colnames(summary) = c("team", "average_points", "playoffs",
"division_titles", "conference_championships", "championships")
summary$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

for(i in 1:iterations) {
  season_stats = team_info[,which(colnames(team_info) !=
"inaugural_season")]
  season_stats$points = 0
  season_stats$rand = runif(nrow(team_info))
```

```r
  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 and tie probabilities
    tie_prob = (1/(sqrt(4*pi))) * exp(-((away_elo - (home_elo +
hfa*season_schedule$neutral[j]))^2/160000))
    win_prob = 1 / (10^((away_elo - (home_elo +
hfa*season_schedule$neutral[j]))/400) + 1) - 0.50*tie_prob
    u = runif(1)

    #In the 92 NHL Season teams could win, lose, tie
    if(u < win_prob) {  # Home team wins in regulation
      season_stats$points[home_index] =
season_stats$points[home_index] + 2
    } else if(u < win_prob + 0.50*tie_prob) {  # Treat as Tie
      season_stats$points[home_index] =
season_stats$points[home_index] + 1
      season_stats$points[away_index] =
season_stats$points[away_index] + 1
    } else if(u > win_prob + tie_prob) {  # Away team wins in
regulation
      season_stats$points[away_index] =
season_stats$points[away_index] + 2
    } else { # Treat as tie
      season_stats$points[home_index] =
season_stats$points[home_index] + 1
      season_stats$points[away_index] =
season_stats$points[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
```

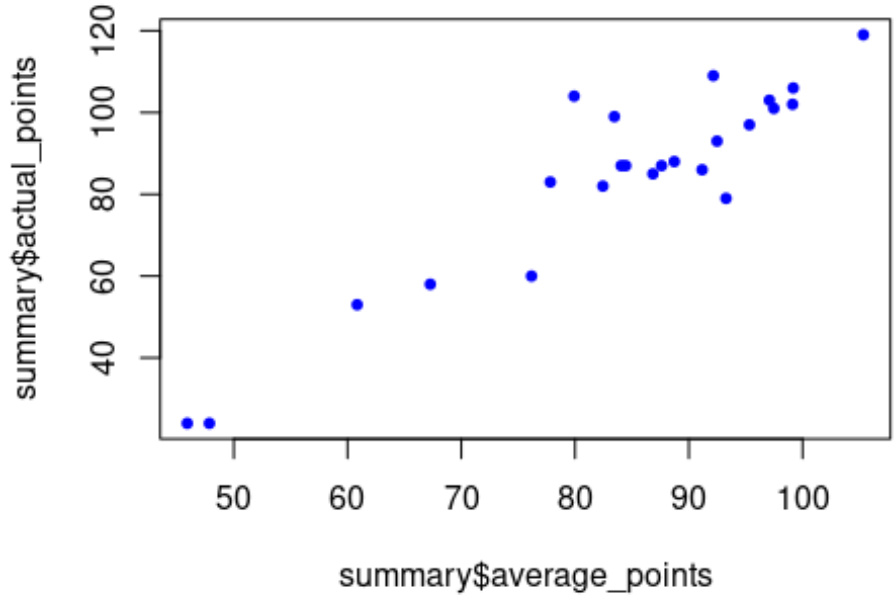
```
    } else {
      home_result = 0.5  # Tie
    }

    # 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
  }

  summary$average_points = summary$average_points +
season_stats$points

  division_winners = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(division_winners) = c("team", "conference", "division",
"rating", "points", "rand")

  non_division_winners = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(non_division_winners) = c("team", "conference", "division",
"rating", "points", "rand")

  num_wild_cards = 0
  wild_card_teams = data.frame(matrix(ncol = 6, nrow = 0))
  colnames(wild_card_teams) = c("team", "conference", "division",
"rating", "points", "rand")

  #Take the top 4 teams from each division and seed by division 1v4
and 2v3 in each division
  for(div in divisions) {
    div_standings = season_stats[which(season_stats$division == div),]
    div_standings = div_standings[order(-div_standings$points, -
div_standings$rand),]
    division_winners = rbind(division_winners, div_standings[1:4,])
    non_division_winners = rbind(non_division_winners,
div_standings[5:nrow(div_standings),])
  }
  #No wild card teams were included

  division_winners =
division_winners[order(division_winners$conference,
division_winners$division, -division_winners$points, -
division_winners$rand),]
  for(j in 1:nrow(division_winners)) {
    index = which(season_stats$team == division_winners$team[j])
```

```
    summary$playoffs[index] = summary$playoffs[index] + 1
    if(j %% 4 == 1) {  # Only increment division winners by 1 in
division titles
      summary$division_titles[index] = summary$division_titles[index]
+ 1
    }
  }

  games_per_round = c(7, 7, 7, 7)

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

  #NHL
  playoff_bracket[1,] = division_winners[1,]
  playoff_bracket[2,] = division_winners[2,]
  playoff_bracket[3,] = division_winners[3,]
  playoff_bracket[4,] = division_winners[4,]
  playoff_bracket[5,] = division_winners[5,]
  playoff_bracket[6,] = division_winners[6,]
  playoff_bracket[7,] = division_winners[7,]
  playoff_bracket[8,] = division_winners[8,]
  playoff_bracket[9,] = division_winners[9,]
  playoff_bracket[10,] = division_winners[10,]
  playoff_bracket[11,] = division_winners[11,]
  playoff_bracket[12,] = division_winners[12,]
  playoff_bracket[13,] = division_winners[13,]
  playoff_bracket[14,] = division_winners[14,]
  playoff_bracket[15,] = division_winners[15,]
  playoff_bracket[16,] = division_winners[16,]

  #Adjusting Division labels
  playoff_bracket$division[4] = playoff_bracket$division[3]
  playoff_bracket$division[8] = playoff_bracket$division[7]
  playoff_bracket$division[12] = playoff_bracket$division[11]
  playoff_bracket$division[16] = playoff_bracket$division[15]

  playoff_bracket$seed = rep(1:4,4)
  playoff_bracket


  # Divisional rounds
  for(round in 1:2) {
    for(j in 1:4) {
      for(k in 1:(nrow(playoff_bracket)/8)) {
        high_seed_index = 0.25*nrow(playoff_bracket)*j-
(0.25*nrow(playoff_bracket)-k)
        low_seed_index = 0.25*nrow(playoff_bracket)*j-(k-1)
```

```
        high_seed_elo = playoff_bracket$rating[high_seed_index]
        low_seed_elo = playoff_bracket$rating[low_seed_index]
        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[round]/2)), 1-rep(low_seed_home_win_prob,
floor(games_per_round[round]/2)))
        u = runif(games_per_round[round])
        high_seed_wins = sum(u < win_probs)/games_per_round[round]

        if(high_seed_wins > 0.50) {
          next_round = rbind(next_round,
playoff_bracket[high_seed_index,])
        } else{
          next_round = rbind(next_round,
playoff_bracket[low_seed_index,])
        }
      }
    }

    playoff_bracket = next_round
    playoff_bracket = playoff_bracket[order(playoff_bracket$division,
playoff_bracket$seed),]
    next_round = NULL
  }

  # Conference championships
  playoff_bracket = playoff_bracket[order(playoff_bracket$conference,
playoff_bracket$seed, -playoff_bracket$points, -
playoff_bracket$rand),]
  for(j in 1:2) {
    high_seed_index = 2*j-1
    low_seed_index = 2*j
    high_seed_elo = playoff_bracket$rating[high_seed_index]
    low_seed_elo = playoff_bracket$rating[low_seed_index]
    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[3])
    high_seed_wins = sum(u < win_probs)/games_per_round[3]

    if(high_seed_wins > 0.50) {
      next_round = rbind(next_round,
```

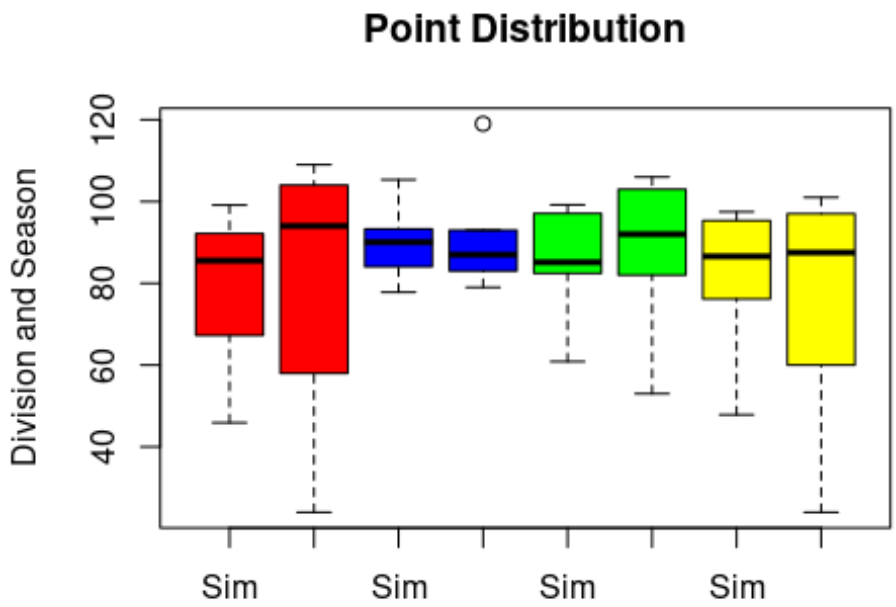
```
playoff_bracket[high_seed_index,])
    } else{
      next_round = rbind(next_round, playoff_bracket[low_seed_index,])
    }
  }

  playoff_bracket = next_round
  playoff_bracket = playoff_bracket[order(playoff_bracket$division,
playoff_bracket$seed),]
  next_round = NULL

  # Stanley Cup Finals
  playoff_bracket = playoff_bracket[order(-playoff_bracket$points, -
playoff_bracket$rand),]

  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[4])
  high_seed_wins = sum(u < win_probs)/games_per_round[4]

  if(high_seed_wins > 0.50) {
    champion = playoff_bracket[1,]
  } else{
    champion = playoff_bracket[2,]
  }

  for(team in playoff_bracket$team) {
    index = which(season_stats$team == team)
    summary$conference_championships[index] =
summary$conference_championships[index] + 1
  }

  index = which(season_stats$team == champion$team)
  summary$championships[index] = summary$championships[index] + 1
  histories[i,] = season_stats$points
}
summary$average_points = summary$average_points/iterations

#Retrieving season results with residuals
summary$actual_points = team_info$points
summary$residuals = summary$actual_points - summary$average_points
```

```r
#Select team, division, average points in simulation, actual points,
and residual
season <- summary %>% select(1, 2, 7, 8)
season$division = team_info$division
season= season[,c(1,5,2,3,4)]
#season[order(season$division, -season$average_points),]

summary$division = team_info$division
#Comparing simulated season vs actual season
#Simulated points vs actual points
plot(summary$average_points, summary$actual_points, col="blue",
pch=20)
```

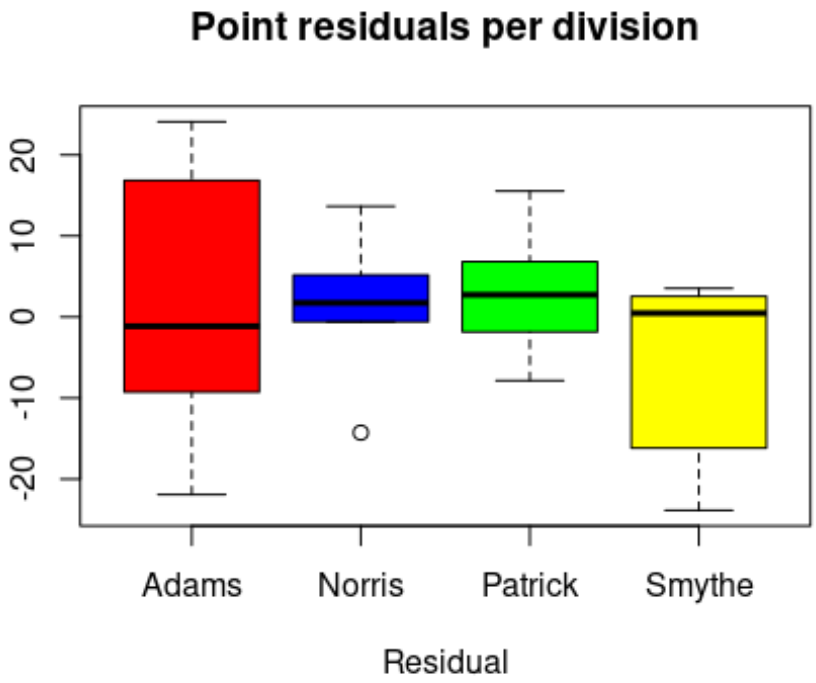



```r
cor(summary$average_points, summary$actual_points)

## [1] 0.927405

#Season
#Splitting up teams by division
adams = summary[which(summary$division == "Adams"),]
norris = summary[which(summary$division == "Norris"),]
patrick= summary[which(summary$division == "Patrick"),]
smythe = summary[which(summary$division == "Smythe"),]


boxcolor = c("Red", "Blue", "Green", "Yellow")
#Creating boxplot to show simulated vs actual point distribution
```

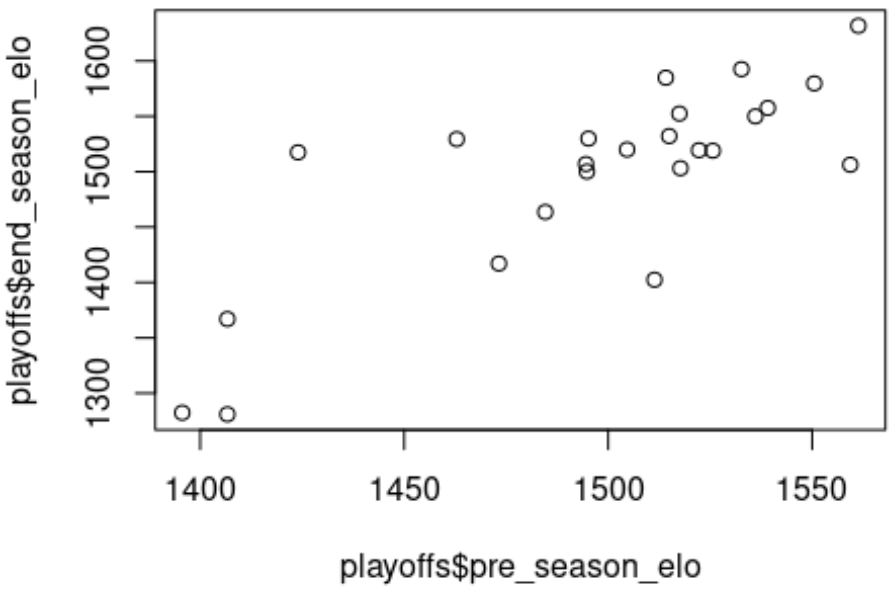
```
boxplot(adams$average_points, adams$actual_points,
patrick$average_points, patrick$actual_points, norris$average_points,
norris$actual_points, smythe$average_points, smythe$actual_points,
names=c("Sim", "Actual", "Sim", "Actual", "Sim", "Actual", "Sim",
"Actual"), ylab="Division and Season", main="Point Distribution",
col=c("Red", "Red", "Blue", "Blue", "Green", "Green", "Yellow",
"Yellow"))
```



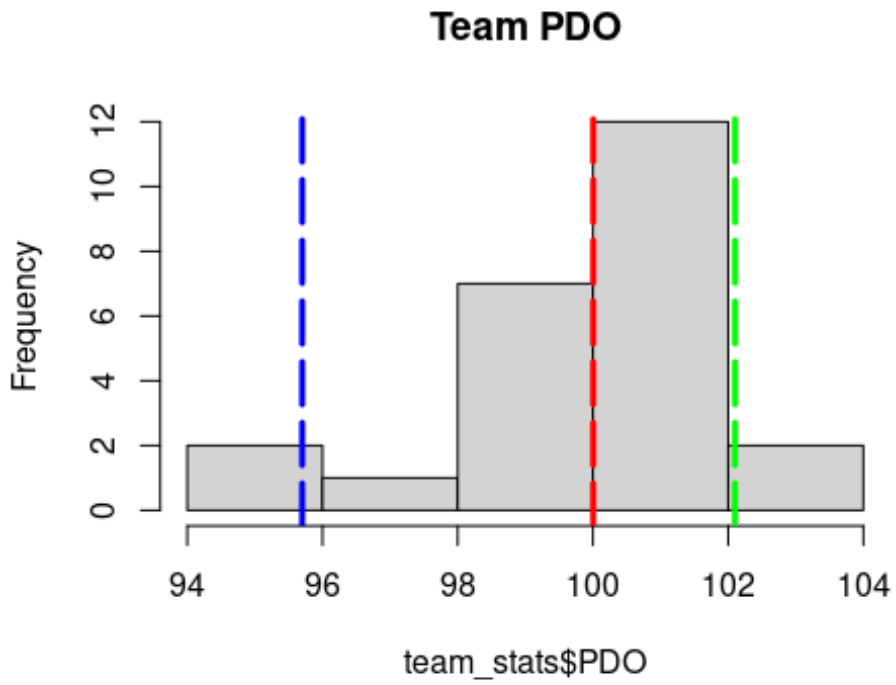

```
#Boxplot for point residuals per division
boxplot(adams$residuals, patrick$residuals, norris$residuals,
smythe$residuals, names=c("Adams", "Norris", "Patrick", "Smythe"),
ylad = "Division", main="Point residuals per division", col=boxcolor,
xlab="Residual")
```



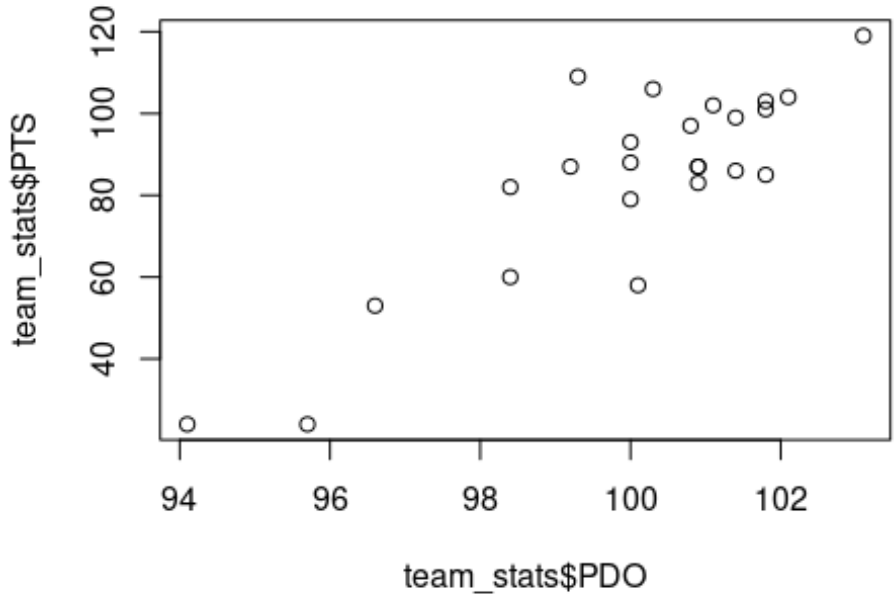

```
#Playoffs
playoffs <- summary %>% select(1, 3, 4, 5, 6, 9)
playoffs = playoffs[,c(1,6,2,3,4,5)]
#Insert pre season elo and end of season elos
playoffs$pre_season_elo = team_info$rating
playoffs$end_season_elo = season_stats$rating
#Calculate elo difference after season
playoffs$elo_difference = playoffs$end_season_elo -
playoffs$pre_season_elo
playoffs = playoffs[order(playoffs$division, -playoffs$champions),]

plot(playoffs$pre_season_elo, playoffs$end_season_elo)
```




```
#Taking a closer look at 3 teams
#Close to expectations - Caps, Underachiever - Sharks, Overachiever -
Nordiques
#Filter just these 3 teams from regular season and playoff dataframes
season_subset = summary[c(12,24,18),]
teams_subset = playoffs[c(14, 24, 4),]
teams_subset$simulation_points = season_subset$average_points
teams_subset$actual_points = season_subset$actual_points

#Calculating PDO based on team metrics
#Team metrics scraped from hockey reference
team_stats <- read.table("teamstats92.csv", header=TRUE, sep=",")
#Renaming Columns
team_stats <- team_stats%>% rename(Team = X)
team_stats <- team_stats %>% mutate_at("Team", str_replace_all,
'[[:punct:]]', "")

#PDO = Shooting + Save Percentages
team_stats$PDO = team_stats$S. + (team_stats$SV. * 100)


hist(team_stats$PDO, main="Team PDO")
abline(v=mean(team_stats$PDO), col="red", lwd=3, lty=5)
abline(v=95.7, col ="blue", lwd=3, lty=5)
abline(v=102.1, col ="green", lwd=3, lty=5)
```




```
plot(team_stats$PDO, team_stats$PTS)
```




```
cor(team_stats$PDO, team_stats$PTS)
```

```
## [1] 0.857758
```

```
model = lm(PTS ~ PDO, data=team_stats)
summary(model)
```

```
##
## Call:
## lm(formula = PTS ~ PDO, data = team_stats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.946  -6.425   0.258   5.650  31.954
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -903.636    126.218  -7.159 3.54e-07 ***
## PDO            9.876      1.262   7.827 8.48e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.82 on 22 degrees of freedom
## Multiple R-squared:  0.7357, Adjusted R-squared:  0.7237
## F-statistic: 61.25 on 1 and 22 DF,  p-value: 8.479e-08
```

```
#Taking a look at interesting results
 # histories[order(-histories$`Pittsburgh Penguins`),]
#Simulation couldn't handle the sharks
 # histories[order(-histories$`San Jose Sharks`),]
```