Predicting Player Season Assists in NHL Hockey

Henry Andrew Wong

26/03/2021

Executive Summary

Goals & Objectives

This project develops machine learning models to predict *Next Season Player Assists* over a National Hockey League (NHL) season, using historical data.

An *assist* is a point awarded to a player for a pass, shot, or deflection that results in a goal by another friendly player. The same goal can award up to two assists for separate friendly players who supported the puck into the net. Assists therefore proxy "playmaking."

The target RMSE (detailed in the *Model Performance* section below) is 10.00. This is arbitrarily set below the standard deviation of 13.76 for season assist totals of all players over 19 NHL seasons.

This summary outlines the major highlights of the project. Code, detailed computations and logic are included in later sections.

Data Acquisition & Wrangling

Model data was sourced from 4 separate .csv files originally from https://www.kaggle.com/martinellis/nhl-game-data:

1. The code in this project automatically downloads source files from a different location, a shared Google Drive folder.
2. Skater game statistics are joined with reference data for players, teams and games.
3. The project proceeds to *player-level* and *season-level* analysis with appropriate aggregations and data slicing.
4. Variables are considered deliberately for model inclusion or exclusion.
5. Details and code are shown in the *Methods & Analysis* section below.

Data Set Selection

The overall logic for dataset partitioning is outlined here. Specific details are presented in the *Training, Testing and Validation Datasets* section below.

1. Since the models predict *Player Season Assists*, the test dataset and the validation dataset should be further forward in time than the training dataset. To be useful, predictions are trained on historical (training) data, and predictions are made on the "future data" (test, followed by validation for the final model only).

2. Because the target response variable is a season-long statistic for a player, all models rely on complete, inseparable season game data, aggregated by season alone or by both season and player.

3. The player game data is separated into 13 whole seasons for training, 3 whole seasons for testing, and 3 whole seasons for validation. Random sampling is used in training models where hardware limits arise.

4. Only the final model (showing the highest predictive power, with the lowest RMSE out of the models tested) is validated against the validation data set.

Models Attempted

The models attempted were:

1. *rt1*: A regression tree using limited variables.
2. *rt2*: A regression tree using many more quantitative variables.
3. *lm1*: A linear model using limited variables.
4. *lm1a*: A linear model using many more quantitative variables.
5. *lm1_rt3*: A regression tree using a linear prediction as an additional predictor.
6. *rt1_lm3*: A linear model using a regression tree prediction as an added predictor.
7. *rf1*: A random forest using random sampling from the training setr.

Each of these experiments is detailed in the *Models* section.

Summary of Results

Detailed computations are referred to here and shown in later sections:

1. The target statistic (next season player assists total) ranges between 0 and 96, with a median of 9, a mean of 13, and a standard deviation of 13.76, for 19 NHL seasons.
2. The target RMSE was set at 10.000, balancing usefulness with difficulty of the task.
3. Model RMSE results ranged between 9.117 and 9.9635.
4. The random forest model returned different RMSEs in a range, depending on iterative sampling in training. A seed value is set for repeatable results.
5. An ensemble model that first produced predictions from a regression tree, then featured the predictions as an input to a linear model using an exhaustive set of season summary variables returned the best RMSE of 9.117.
6. The final model was validated against the latest three NHL seasons on record (excluding the ongoing season on the date of this report). This reported an RMSE of 9.197.

Detailed discussion and quantitative results are covered in the sections below.

Predictions in Context. Why Predict Player Season Assists?

A player's season assist total is a major dimension of player value:

1. An assist counts for a point. This is the same value as a goal when computing points.
2. Some points recorded as assists had a high chance of becoming goals, especially when executed close to the opposing net. One example is when a shot directed at the net is deflected in by a friendly player, awarding the goal to the deflecting player instead of the shooter.
3. Assists not converted from direct shots at the net, such as those purposely bounced off a goalie to create unpredictable chances, qualify as "playmaking." There are many classifications of playmaking. Many of the top players in the NHL are better playmakers than they are goal scorers.
4. A goal can be mapped to up to two assists (ex. if one player passes to another, and the second player passes to the eventual goal scorer).

These attributes make predictions valuable for:

1. Player valuations for trades.
2. Odds making in the casino industry.
3. Advantage hockey pool performance among private groups.
4. Hockey season simulations for commercial or gaming purposes.

There are plenty of applications for the models developed in this project.

Project Setup

Conditional Library Installation & Invocation

This study was performed on R version 3.6.3. All required packages for this project are conditionally installed and unconditionally invoked.

```
# Install packages as required.
# To handle data piping, mutations, etc.
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
# For classification and regression tree models.
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
# Data tables at download.
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
# Tools that help with aggregations and data handling.
if(!require(cgwtools)) install.packages("cgwtools", repos = "http://cran.us.r-project.org")
# To display pretty tables.
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
# Random forest models.
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
# Regression tree models (and classification trees, but not used here).
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
# To support plotting of regression trees.
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
```

```r
if(!require(party)) install.packages("party", repos = "http://cran.us.r-project.org")
# For pretty plots of regression trees.
if(!require(partykit)) install.packages("partykit", repos = "http://cran.us.r-project.org")
# To plot correlations.
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
# Invoke libraries.
library(corrplot)
library(partykit)
library(rpart)
library(rpart.plot)
library(tidyverse)
library(caret)
library(data.table)
library(kableExtra)
library(cgwtools)
library(rpart)
library(randomForest)
```

Core Data Description

The data used in this study is originally from:

https://www.kaggle.com/martinellis/nhl-game-data?select=game_skater_stats.csv#

However, code has been provided to automatically download the same files from a separate Google Drive location. Failing that, there is commented backup code to load the files from R's working directory after downloading them manually from:

https://github.com/henryawong/edX

The following are insights gleaned from exploring the base data in depth and tying it to published league information linked in the *Reference* section.

Field by field descriptions are in the *Field Level Descriptions* section below. These observations, taken together, indicate how difficult the task of predicting within 10.0000 assists is.

1. The player game data covers 19 NHL hockey seasons, from 2000/2001 to 2019/2020. Goaltender statistics are ignored.
2. A typical NHL season consists of 82 games per team, running between October and April, with playoffs running until June. One game constitutes 2 "team-perspective" games (one for each team playing). A "player game" follows the same principle.
3. The target response variable is *Next Season Player Assists*. This is not to be confused with the assist total for the "current season" of the row analyzed. A data structure is constructed to suit this type of analysis. The statistic is the total of assists awarded to one player, over an entire NHL season, no matter the length of the season.
4. The number of league games per season was adjusted when the Vegas Golden Knights were added to league statistics, starting in 2017/2018.
5. Labor disputes between the NHL and the NHLPA (the players' association) caused lockouts in 2004/2005 (season canceled) and 2012/2013 (season shortened to about 60% of a regular season).
6. The COVID-19 pandemic shortened the 2019/2020 season.

7. Injuries, game-day roster changes, and exogenous events can cause some players to miss games and others to gain games as replacement players.
8. The core data used in this project is from skaters (forward and defense players). Goalies are excluded.
9. Nineteen seasons of data will tend to expose mean reversions, outlier seasons for players (unusually good or poor), and stub-ended career statistics (only the first or last few seasons of a player's career).
10. Injuries resulting in game absences generally dampen a player's assist total for a season.
11. A player aging variable is considered. Most hockey players exhibit pronounced changes in playstyle, team role, and performance as they age.

Ice hockey is an especially interesting sport with respect to player age. Players between the ages of 18 up to 45 in the modern era are regularly on the ice together, each with a diverse range of abilities to offer.

Age does not always indicate year-over-year decline. For example, at the ages of 43 and 44, Jaromir Jagr achieved 66 and 46 points in full seasons of play. Most young forwards never make 46 points in their highest seasons.

Data Acquisition & Wrangling

CSV files are automatically downloaded from a publicly shared Google Drive location, and read into starting variables.

Failing automatic download, there is commented code to load manually acquired files from R's working directory, into the variables used in this project. The block can be uncommented and executed.

If manual download is required, the files are available at:
https://github.com/henryawong/edX

Samples of the resulting variable rows are shown.

```
# This code automatically downloads the required files from a publicly
#  shared Google Drive location.
#  There is a commented block at the end of this section for backup
#  if there are problems with the automatic download.

dl_game_skater_stats <- tempfile()
download.file("https://drive.google.com/uc?id=1M7MZS_vTs3rWu7R_cOMbI0Vdp7vQwdtf", dl_game_skater_stats)
game_skater_stats <- read.csv(dl_game_skater_stats)
head(game_skater_stats,3)
```

```
##      game_id player_id team_id timeOnIce assists goals shots hits
## 1 2016020045   8468513       4       955       1     0     0    2
## 2 2016020045   8476906       4      1396       1     0     4    2
## 3 2016020045   8474668       4       915       0     0     1    1
##   powerPlayGoals powerPlayAssists penaltyMinutes faceOffWins faceoffTaken
## 1              0                0              0           0            0
## 2              0                0              2           0            0
## 3              0                0              0           0            0
##   takeaways giveaways shortHandedGoals shortHandedAssists blocked plusMinus
## 1         1         1                0                  0       1         1
## 2         1         2                0                  0       2         0
## 3         2         0                0                  0       0        -1
##   evenTimeOnIce shortHandedTimeOnIce powerPlayTimeOnIce
```

```
## 1           858                  97                 0
## 2          1177                   0               219
## 3           805                   0               110
```

```
dl_player <- tempfile()
download.file("https://drive.google.com/uc?id=1wNzxzlk6n104mTWIN4Em8rf8wW_xJw8K", dl_player)
player_info <- read.csv(dl_player)
head(player_info,3)
```

```
##   player_id firstName     lastName nationality      birthCity primaryPosition
## 1   8466148    Marian        Hossa         SVK StarÃ¡ Lubovna              RW
## 2   8465058    Michal     Rozsival         CZE         Vlasim               D
## 3   8476906    Shayne Gostisbehere         USA Pembroke Pines               D
##             birthDate birthStateProvince height height_cm weight shootsCatches
## 1 1979-01-12 00:00:00               <NA>  6' 1"    185.42    207             L
## 2 1978-09-03 01:00:00               <NA>  6' 1"    185.42    210             R
## 3 1993-04-20 01:00:00                 FL 5' 11"    180.34    180             L
```

```
dl_game <- tempfile()
download.file("https://drive.google.com/uc?id=13YxSU7uTtVaRP1pBYVP8DzSA60tEy9kQ", dl_game)
game <- read.csv(dl_game)
head(game,3)
```

```
##      game_id   season type        date_time_GMT away_team_id home_team_id
## 1 2016020045 20162017    R 2016-10-19T00:30:00Z            4           16
## 2 2017020812 20172018    R 2018-02-07T00:00:00Z           24            7
## 3 2015020314 20152016    R 2015-11-24T01:00:00Z           21           52
##   away_goals home_goals      outcome home_rink_side_start          venue
## 1          4          7 home win REG                right  United Center
## 2          4          3  away win OT                 left KeyBank Center
## 3          4          1 away win REG                right     MTS Centre
##            venue_link venue_time_zone_id venue_time_zone_offset
## 1 /api/v1/venues/null    America/Chicago                     -5
## 2 /api/v1/venues/null   America/New_York                     -4
## 3 /api/v1/venues/null   America/Winnipeg                     -5
##   venue_time_zone_tz
## 1                CDT
## 2                EDT
## 3                CDT
```

```
dl_team <- tempfile()
download.file("https://drive.google.com/uc?id=1qJ1XWYO72JmQQTkTLgIqCHA6niAvQ_Le", dl_team)
team_info <- read.csv(dl_team)
head(team_info,3)
```

```
##   team_id franchiseId    shortName teamName abbreviation             link
## 1       1          23   New Jersey   Devils          NJD  /api/v1/teams/1
## 2       4          16 Philadelphia   Flyers          PHI  /api/v1/teams/4
## 3      26          14  Los Angeles    Kings          LAK /api/v1/teams/26
```

```
# Only uncomment and use this code block if automatic download fails.
#  The CSV files should be downloaded manually from here:
#  https://github.com/henryawong/edX
#  Place them in R's before running the code below.
#
#  ### Code block starts here ###
#  game_skater_stats  <- read.csv("game_skater_stats.csv")
#  player_info  <- read.csv("player_info.csv")
#  team_info  <- read.csv("team_info.csv")
#  game <- read.csv("game.csv")
#  ### Code bock ends here ###
```

Data Assembly

The core data, as loaded above, are listed here. Detailed descriptions of table joins, derived data grains, and individual fields are built up in the section below.

1. *game_skater_stats*: Game-by-game statistics for players including assists, goals, shots, hits, and time on ice.
2. *player_info*: Player details including first name, last name, height, weight, birth date, and primary playing position.
3. *team_info*: Team details, including team names and ids for every NHL team in the league over the 19 most recently completed seasons. This includes teams that have bee added, moved, and / or renamed.
4. *game*: Game data including game date, arena, outcome, and game id.

Methods & Analysis

Data Exploration & Analysis

The data is explored and analyzed for insights useful to selecting predictors, substituting proxy variables, and understanding interactions that can help us build models and interpret their results.

Skater Game Data

This section covers skater data:

1. The core file is skater game statistics. Skaters are forwards and defenders. Forwards can be left wing (LW) or right wing (RW).
2. This is joined to player information such as name, height, weight, and birthdate.
3. Then, team data is joined to team identifiers.

```
# Join skater data with tables for player attributes,
#  game information, and team information.
player_game_data <- game_skater_stats %>%
  left_join(player_info, by="player_id") %>%
  left_join(game, by="game_id") %>%
```

```
  left_join(team_info, by="team_id") %>%
#  Add a points field.
  mutate(points = goals + assists) %>%
#  Eliminate any duplicate rows (from joins or originals)
  unique() %>%
#  Narrow to only wanted fields.
#  Keep other fields in mind for future study.
  select(firstName, lastName, season
       , timeOnIce, assists, goals, points, shots, hits
       , season, game_id, player_id
       , teamName, team_id
       , height, weight, birthDate, nationality, primaryPosition
       , date_time_GMT)

# Sample rows.
head(player_game_data,3)
```

```
##   firstName     lastName   season timeOnIce assists goals points shots hits
## 1      Nick      Schultz 20162017       955       1     0      1     0    2
## 2    Shayne Gostisbehere 20162017      1396       1     0      1     4    2
## 3      Dale        Weise 20162017       915       0     0      0     1    1
##      game_id player_id teamName team_id height weight           birthDate
## 1 2016020045   8468513   Flyers       4  6' 1"    203 1982-08-25 01:00:00
## 2 2016020045   8476906   Flyers       4 5' 11"    180 1993-04-20 01:00:00
## 3 2016020045   8474668   Flyers       4  6' 2"    206 1988-08-05 01:00:00
##   nationality primaryPosition        date_time_GMT
## 1         CAN               D 2016-10-19T00:30:00Z
## 2         USA               D 2016-10-19T00:30:00Z
## 3         CAN              RW 2016-10-19T00:30:00Z
```

Field Level Data Descriptions

The fields in the main data structure are listed with their data types and descriptions:

firstName: Factor: First name of the player.

lastName: Factor: Last name of the player.

season: Integer: Season the game belongs to.

timeOnIce: Integer: Total minutes on the ice (not on bench) played by a player in a game.

assists: Integer: Assists awarded to the player in a game row.

goals: Integer: Goals scored by the player in a game.

points: Integer: Goals + Assists by the player in the game.

shots: Integer: Shots on the opposing net in the game. Missing the net does not count.

hits: Integer: Legal hits by the player on an opposing player (ie. the opponent has the puck).

game_id: Integer: Unique identifier for a game.

player_id: Integer: Unique identifier for a player.

team_id: Integer: Unique identifier for the player's team.

teamName: Factor: Name of the player's team on game day. Can differ in consecutive games if traded, drafted to an expansion team, or if the team moved. Overnight trades have happened.

height: Factor: Player's height in feet and inches.

weight: Integer: Player's weight in pounds.

birthDate: Factor: Player's birth date.

nationality: Factor: Nationality of the player, at birth or by earliest citizenship.

Row & Entity Counts

The joined database covers 853,404 observations (game rows for players) covering 3353 players, across 37 teams, over 19 seasons from 2000/2001 to 2019/2020.

Some observations:

1. Many players only appear in a portion of the 19 seasons window (rookies, retiring players, injured players).
2. One team was added to the league in 2017/2018 (Vegas Golden Knights).
3. The Atlanta Thrashers moved to become the Winnipeg Jets in 2011/2012.
4. Many players were traded to other teams (and sometimes back) during the seasons window.
5. There were labour disputes that resulted in lockouts, canceling the entire 2004/2005 season and shortening the 2012/2013 season to about 60% the normal game count.

```
# Show a data sample.
player_game_data %>% head(3)
```

```
##   firstName     lastName   season timeOnIce assists goals points shots hits
## 1      Nick      Schultz 20162017       955       1     0      1     0    2
## 2    Shayne Gostisbehere 20162017      1396       1     0      1     4    2
## 3      Dale        Weise 20162017       915       0     0      0     1    1
##      game_id player_id teamName team_id height weight          birthDate
## 1 2016020045   8468513   Flyers       4  6' 1"    203 1982-08-25 01:00:00
## 2 2016020045   8476906   Flyers       4 5' 11"    180 1993-04-20 01:00:00
## 3 2016020045   8474668   Flyers       4  6' 2"    206 1988-08-05 01:00:00
##   nationality primaryPosition       date_time_GMT
## 1         CAN               D 2016-10-19T00:30:00Z
## 2         USA               D 2016-10-19T00:30:00Z
## 3         CAN              RW 2016-10-19T00:30:00Z
```

```
# Count player game rows.
nrow(player_game_data)
```

```
## [1] 853404
```

```
# Count players. View as factor and count unique levels.
factor(player_game_data$player_id) %>%
  levels() %>% length()
```

```
## [1] 3353
```

```r
# Count teams.  Use levels function to count as factor.
factor(player_game_data$team_id) %>%
  levels() %>% length()
```

```
## [1] 37
```

```r
# Count seasons.
factor(player_game_data$season) %>%
  levels() %>% length()
```

```
## [1] 19
```

```r
# List seasons, sorted.  Note missing 2004/2005 canceled season.
factor(player_game_data$season) %>%
  levels() %>% sort() %>% data.frame()
```

```
##           .
## 1  20002001
## 2  20012002
## 3  20022003
## 4  20032004
## 5  20052006
## 6  20062007
## 7  20072008
## 8  20082009
## 9  20092010
## 10 20102011
## 11 20112012
## 12 20122013
## 13 20132014
## 14 20142015
## 15 20152016
## 16 20162017
## 17 20172018
## 18 20182019
## 19 20192020
```

```r
# Games counts per season.
names(game)
```

```
##  [1] "game_id"                "season"                 "type"
##  [4] "date_time_GMT"          "away_team_id"           "home_team_id"
##  [7] "away_goals"             "home_goals"             "outcome"
## [10] "home_rink_side_start"   "venue"                  "venue_link"
## [13] "venue_time_zone_id"     "venue_time_zone_offset" "venue_time_zone_tz"
```

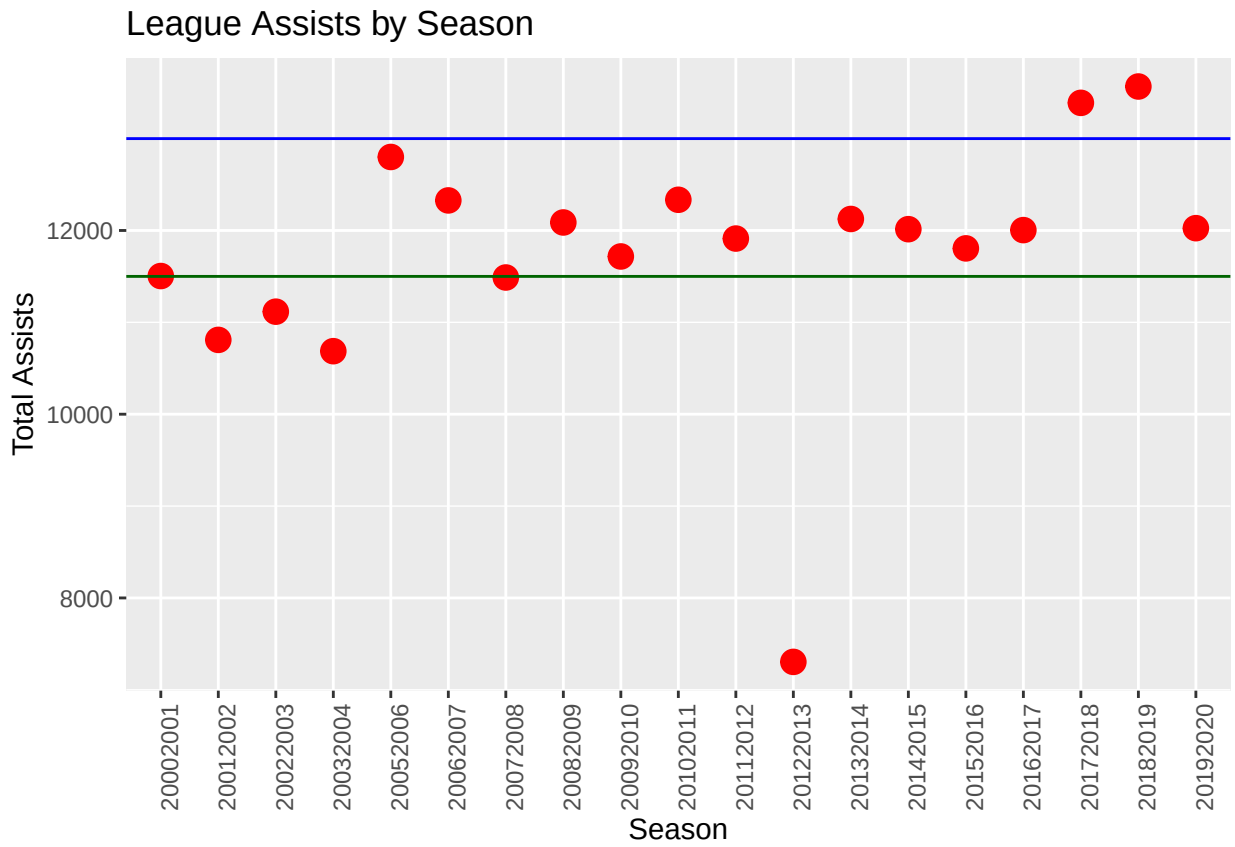
```r
game %>% group_by(season) %>%
  select(game_id) %>% unique() %>%
  summarize(gameCount=n())
```

```
## # A tibble: 19 x 2
##      season gameCount
##       <int>     <int>
##  1 20002001      1230
##  2 20012002      1230
##  3 20022003      1230
##  4 20032004      1230
##  5 20052006      1230
##  6 20062007      1230
##  7 20072008      1230
##  8 20082009      1230
##  9 20092010      1230
## 10 20102011      1319
## 11 20112012      1316
## 12 20122013       806
## 13 20132014      1323
## 14 20142015      1319
## 15 20152016      1321
## 16 20162017      1323
## 17 20172018      1363
## 18 20182019      1360
## 19 20192020      1215
```

League Season Assists

First, season summations are calculated and plotted.

```
# Create a data summary of
#  league assists grouped by season.
#  Summarize by computing sums for goals, points, and assists.
season_summary <- player_game_data %>% group_by(season) %>%
  summarize(sumAssists=sum(assists)
            , sumGoals=sum(goals)
            , sumPoints=sum(points))

# Turn off scientific notation to keep axis labels
#  clean in the following plots and graphs.
options(scipen=999)

# Graph total league assists by included seasons.
#  Make markers large and color them red.
#  Turn the season labels on the x-axis to 90 degrees to avoid crowding.
#  Eliminate the legend.
#  Label with a title and axis labels.
#  Put colored horizontal lines at y values of 13000 and 11500 to
#  help the eye spot ranges.
season_summary %>%
  ggplot(aes(x=as.factor(season), y=sumAssists)) +
          geom_point(color="red", size=4) +
          theme(axis.text.x=element_text(angle=90) ,legend.position = "none") +
          labs(title = "League Assists by Season", y="Total Assists", x="Season") +
          geom_hline(yintercept=13000, colour="blue") +
          geom_hline(yintercept=11500, colour="darkgreen")
```



Insights gained from plotting total league assists by sequential season:

1. Relative to later seasons, league assists clustered lower from 2000/2001 to the end of 2003/2004.

2. 2005/2006 saw a sudden increase in assists, followed by a subsequent drop back to 2000/2001 levels over the next two seasons. The 2005/2006 peak could have reflected changes in rules, equipment, team composition, rest after a canceled season, exuberance returning from a lockout or even playstyle changes. There are too many factors to consider a definitive reason here.

3. All of the seasons from 2005/2006 to 2016/2017 (except the 2012/2013 lockout season) ranged between 11,500 to almost 13,000 assists, league-wide.

4. A major labor dispute shortened the 2012/2013 season to about 60% of a normal season, bringing down the total assist count. This is an outlier year by definition, but its data are included in training to reflect the very realistic possibility of shorter future seasons. The current COVID-19 pandemic has already shortened seasons. Although our predictions are for season player assist totals, season means are also included as predictors (covered in the *Season Aggregation* section below).

5. 2017/2018 and 2018/2019 saw total assists climb and sustain clearly above the prior league totals. This is most likely due to adding the Vegas Golden Knights franchise to the league. In addition to adding another team's worth of assists, the team's first two years of performance were excellent, going to the playoff final in their first franchise year.

6. In 2019/2020 league assists fell back to the sustained average before Vegas joined, but this most likely reflects the shorter season caused by COVID-19 canceled games. With one added team in the league producing assists, a higher league season total is very likely. The 2019/2020 season is likely an impaired version of a new average league total for full seasons. 2019/2020 had 1215 games, versus the 1360 in 2018/2019.

7. Average league-wide total season assists is 11,738 with 2012/2013 included, and 11,984 without that season. Removing the cluster of lower totals before 2005/2006 escalates the mean to 12,215. This suggests that the modern era since 2005 likely has a stable average for total league assists. Thus we will be estimating player season total assists over a backdrop of general league stability in segments.

8. From one perspective, the prediction of player season assists is an allocative exercise within a relatively stable range of league assists per season. Exogenous events like labor disputes can have a reducing effect on league total assists, but it appears these are scale changes from canceled games.

```
# Mean of all league season assist totals.
mean(season_summary$sumAssists)
```

```
## [1] 11737.53
```

```
# Remove 20122013 and recalculate mean.
season_summary %>% filter(season != 20122013) %>%
  summarize(avgSumAssists = mean(sumAssists))
```

```
## # A tibble: 1 x 1
##   avgSumAssists
##           <dbl>
## 1        11984.
```

```
# Remove lower cluster and outlier year and recalculate.
season_summary %>%
  filter(season != 20122013) %>%
  filter(season > 20052006) %>%
  summarize(avgSumAssists = mean(sumAssists))
```

```
## # A tibble: 1 x 1
##   avgSumAssists
##           <dbl>
## 1        12215.
```

Player Season Assists

The number of assists that a player registers in one season is the target response variable. Season player data is aggregated here to:

1. Examine season-level quantities for insights and clues for related predictors.

2. Prepare the data for use in models.

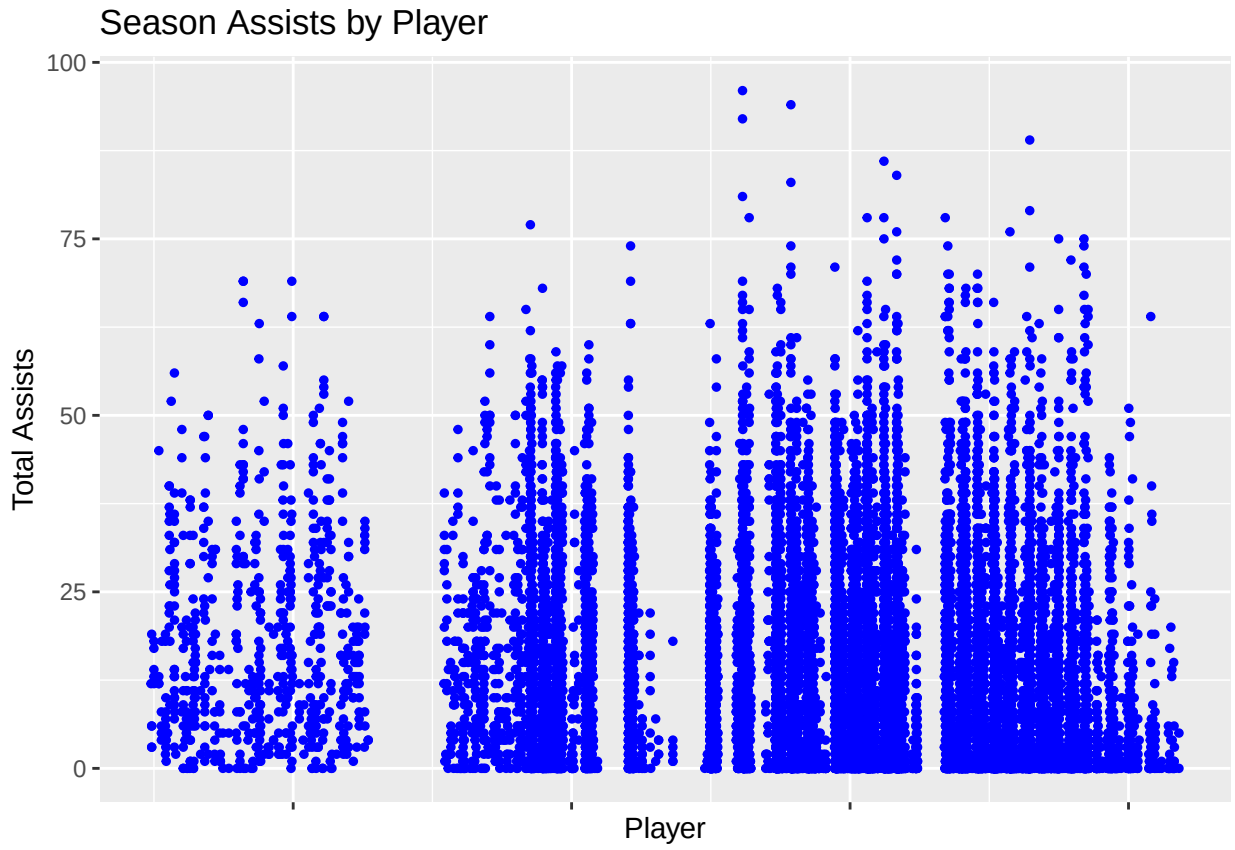
3. Unearth insights.

```
# Prepare data to graph season assist totals per player.
#  Must group by season, then player within a season.
#  Compute total goals, assists and points.
player_season_summary <- player_game_data %>% group_by(season, player_id) %>%
      summarize(sumAssists=sum(assists)
            , sumGoals=sum(goals)
            , sumPoints=sum(points))
```

The plot shows a wide range of season assist totals per player. The player IDs on the x-axis are meaningless here so they are omitted from labeling. A numerical summary is computed for further insight.

Player assist totals are displayed for all players and all seasons here, so any given player has his own seasons compared.

```
# Graph season assists by player.
#  Blank out player IDs on the x-axis, as they are meaningless here.
#  Make points blue and dots small.
player_season_summary %>%  ggplot(aes(x=player_id, y=sumAssists)) +
          geom_point(color="blue", size=1) +
          theme(axis.text.x=element_blank(),legend.position = "none") +
          labs(title = "Season Assists by Player", y="Total Assists", x="Player")
```




```
# Minimum, maximum, and interquartile range.
summary(player_season_summary)
```

```
##      season           player_id         sumAssists       sumGoals
##  Min.   :20002001   Min.   :8444894   Min.   : 0.00   Min.   : 0.000
##  1st Qu.:20052006   1st Qu.:8466240   1st Qu.: 2.00   1st Qu.: 1.000
##  Median :20102011   Median :8470609   Median : 9.00   Median : 4.000
##  Mean   :20100198   Mean   :8469358   Mean   :13.28   Mean   : 7.804
##  3rd Qu.:20152016   3rd Qu.:8474695   3rd Qu.:20.00   3rd Qu.:12.000
##  Max.   :20192020   Max.   :8481813   Max.   :96.00   Max.   :65.000
##    sumPoints
##  Min.   :  0.00
```

```
##  1st Qu.:  3.00
##  Median : 14.00
##  Mean   : 21.09
##  3rd Qu.: 32.00
##  Max.   :130.00
```

```
# Calculate standard deviation.
sd(player_season_summary$sumAssists)
```

```
## [1] 13.75763
```

The season and player_id summaries can be ignored, as they have no quantitative meaning.

Insights:

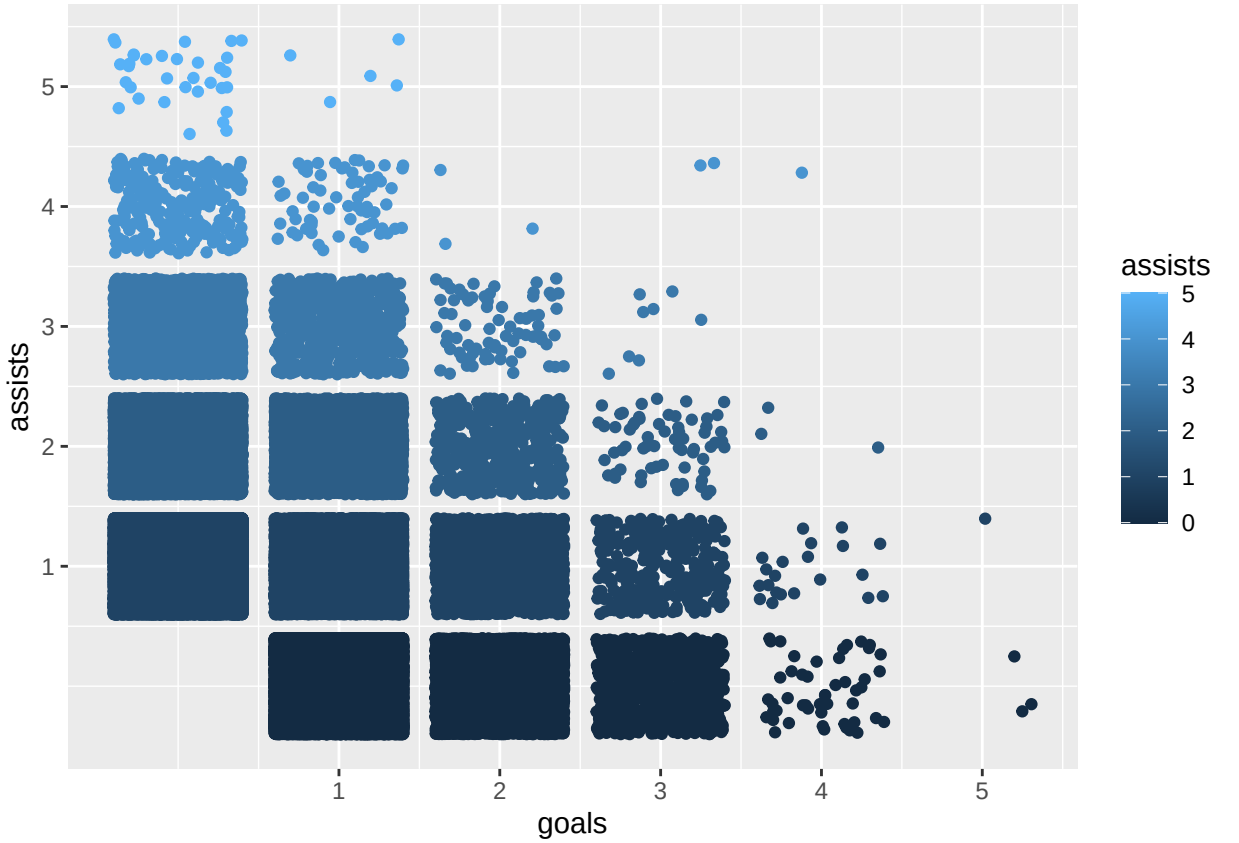
1. The summary of assists shows a mean of 13 but a median of 9, so there are players with higher assist totals pulling up the mean.
2. The interquartile range is bounded from 2 assists in a season to 20 by individual players.
3. The 4th quartile starts at 20, but rises to an extreme value of 96. The elite playmakers of the league extend far and away from the mean in assists.
4. These wide variations bar using a simple mean as an effective predictor. It is obvious that different players reach very different assist totals per season.

Points, Goals & Assists

Points are the sum of goals and assists, whether measured by season, team, player, or any other grouping. Not all great scoring players earn high assist totals, nor vice versa. Some players bias to assists in their point totals (per game or season) while others enjoy goals as the dominant contributor to their points.

In this section, the rows showing zero points for a player in a game are removed, to focus only on point-producing players (at the game level).

```
#  Filter to only games where players produce points.
players_with_points <- player_game_data %>%
  filter(points > 0)

# Graph goals versus assists per player for games where the player
#  earns points.  Use a scatterplot.
#  Goals on the x-axis, assists on the y-axis, points as dot size.
#  Assists in brighter shading.
#  Put markers on every integer on the axes.
#  Create in a graph object for layers added later.
plot_points <- players_with_points %>%
  ggplot(aes(goals, assists, color=assists)) +
  geom_jitter() +
  scale_x_continuous(breaks = seq(1, 5,1)) +
  scale_y_continuous(breaks = seq(1, 5,1))

# Show graph by calling graph object.
plot_points
```



The clear observations from "assists per player game" plotted by "goals per player game" are listed. Some clues may need quantitative drill-down but are limited in this section.

1. There are more games where players register 4 assists than 4 goals.
2. The densely plotted dots at up to 2 goals and 2 assists make it difficult to see clear relationships.
3. In 19 seasons, no player has exceeded 8 points in one game. Sam Gagner's 8-point game consisted of 4 goals and 4 assists.
4. Very few players have scored 5 goals in one game, over the 19 seasons. None of the 5-goal scorers registered an assist in the same game.

The above illustrates a general relationship between assists and goals by a player in the same NHL game.

The correlations calculated show barely any correlation between a player's assists and goals in the same game, when including games where a player earns no points. However, excluding zero-point games shows a strong negative correlation.

This can loosely be interpreted as more assists in a game associating with fewer goals in that game by the same player, and vice versa, in games where the player earned at least 1 point.

```
# Calculate correlation between goals and assists over all player games.
cor(player_game_data$assists, player_game_data$goals)
```

```
## [1] 0.05689724
```

```r
# Correlate them in games where a player earns at least 1 goal or 1 assist.
cor(players_with_points$assists, players_with_points$goals)
```

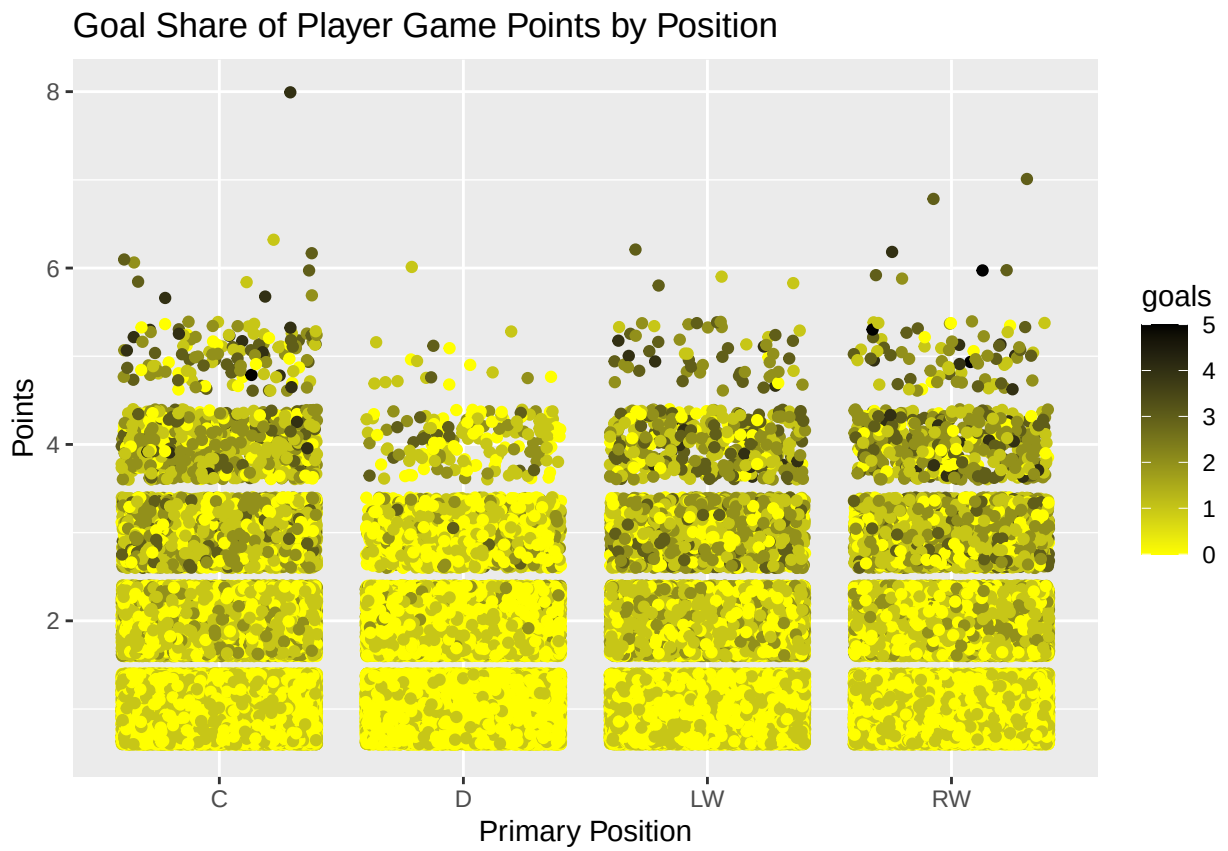
```
## [1] -0.5746525
```

Player Position: Goals vs Assists

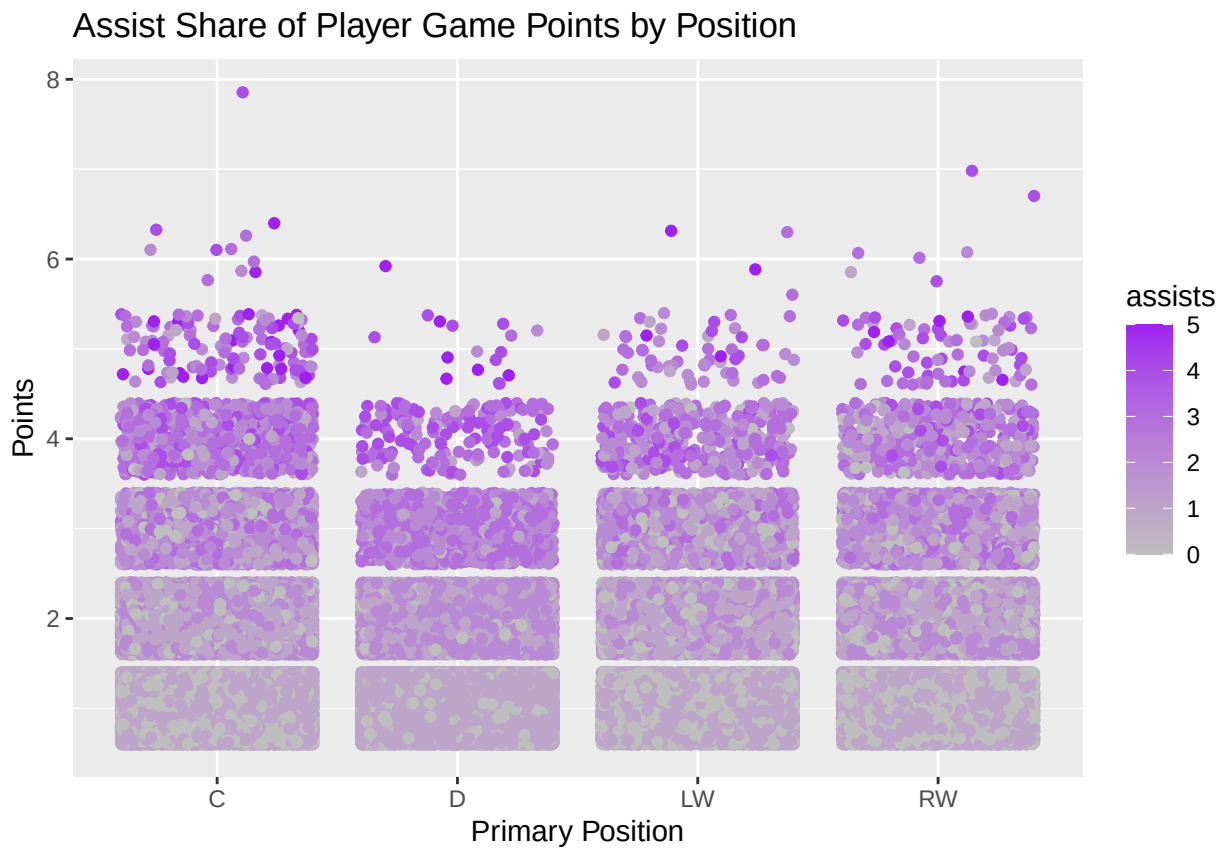
A similar "goal share" versus "assist share" of player game points is plotted by primary playing position.

1. The y-axis is denominated in points per player game.
2. The x-axis categorizes players into their primary on-ice positions.
3. In the first plot (yellow to black), the darker the shading, the more goals contributed to the point observation.
4. In the second plot (gray to purple), the darker the shading, the more assists contributed to the point observation.

```r
# Graph points per player per game.
#  Add grouping by player's ice position.
#  Emphasize GOALS with color fading in high contrast from YELLOW to BLACK.
#  Label it clearly as Goal Share of Points.
players_with_points %>% group_by(primaryPosition) %>%
  ggplot(aes(x=primaryPosition, y=points, color=goals)) +
    geom_jitter() +
    scale_color_gradient(low = 'yellow', high = 'black') +
    labs(title = "Goal Share of Player Game Points by Position", x="Primary Position", y="Points")
```




```
# Graph points per player per game.
#  Add grouping by player's ice position.
#  Emphasize ASSISTS with color fading in high contrast from GRAY to PURPLE.
#  Label it clearly as Goal Share of Points.
players_with_points %>% group_by(primaryPosition) %>%
  ggplot(aes(x=primaryPosition, y=points, color=assists)) +
    geom_jitter() +
    scale_color_gradient(low = 'gray', high = 'purple') +
    labs(title = "Assist Share of Player Game Points by Position", x="Primary Position", y="Points")
```



In games where a player earns any points at all:

1. Not surprisingly, Defense (D) player points are mostly assists. When D players make more points in a game, the role of goals increases, especially in their 4th point and higher.
2. By inspection, it appears that D players are almost equally as likely to achieve 1, 2 and even 3-point games as forward players are. However, a much larger share of D points are from assists.
3. When a player of any position registers exactly 1 point in a game, it is most likely from an Assist. D players have the lowest chance of making the point from a goal. Centers have a visibly higher chance, and Wingers have the best chance of gaining the point from a goal, but still more likely from an Assist.
4. While Left Wing (LW) and Right Wing (RW) players' points are also more likely from assists at the 2 point level, Wingers are slightly more likely to gain points from goals than Center and Defense players are.
5. In games where a player achieves 3 points, forwards (C, LW, RW) appear almost equally likely to be from Goals or Assists, while D players maintain a bias to assists.
6. At 4 points, forwards have the advantage over D players, to produce 4 points in one night, regardless of goal or assist content.
7. The density of 4-Point games among Centers is significantly higher than that of LW and RW players.
8. Centers have a strong advantage over other forwards, and a strong advantage over D players for producing both goals and points, in personal 5-point games.
9. Personal 5-Point games favor RW players slightly over LW players.

10. At 6-points, Centers retain a clear advantage for point production, over all other positions, especially D.

11. The top point nights are outliers (and often reflect empty-net goals because losing teams tend to pull their goalies in the final minutes of the game) but appear to favor Right Wingers. It should be noted that the outlying nature of such elite performance could map more to individual player attributes than playing positions.

The varying point production by primary playing position is sufficient to mark position as a predictor to include in models.

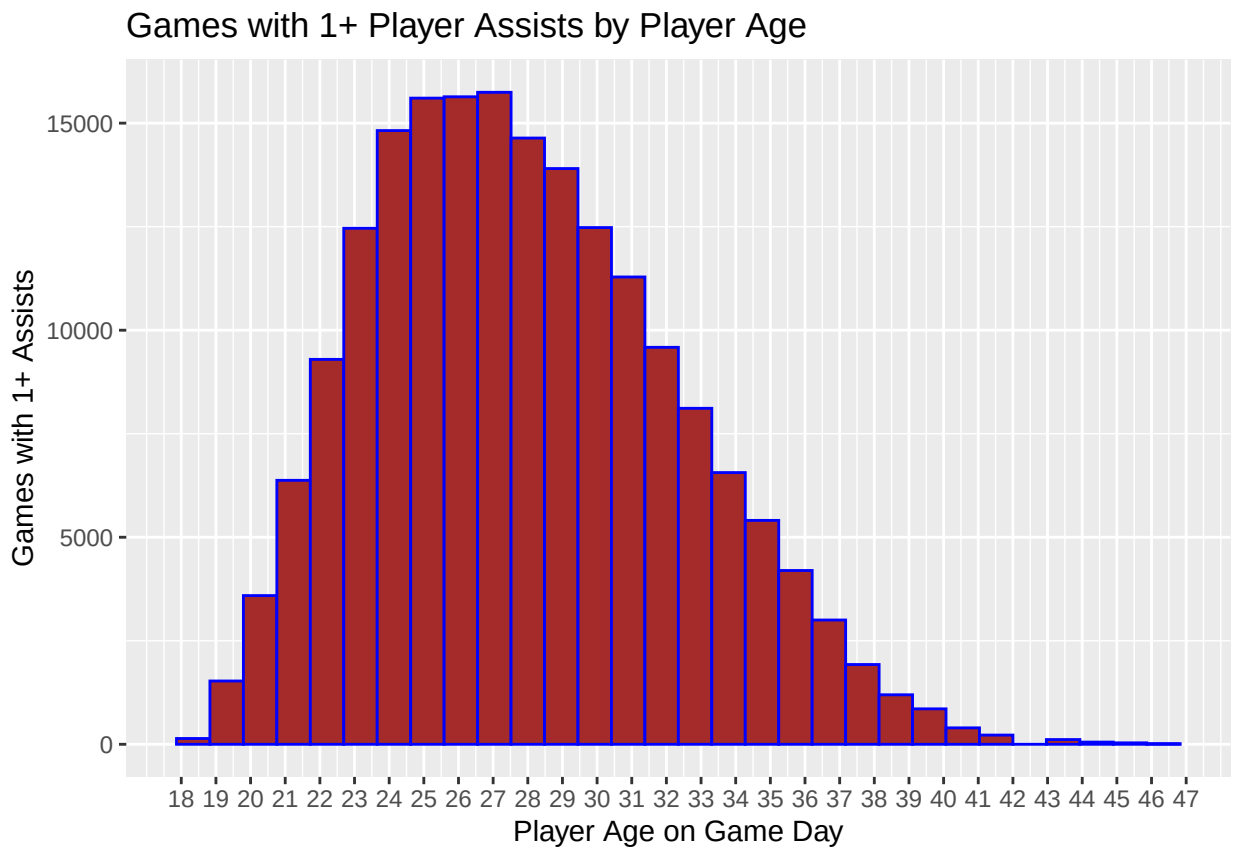
Player Age

Does age on game day affect player season assists? Some playmakers register points immediately upon entering the NHL. Experience on the ice can make other, average players into high performers. Many players take some years to develop higher peaks but surpass those who drafted higher.

This is a well-known phenomenon among hockey fans and adds a dimension of mystery for long-term observers. Player season assists are an accepted indicator of performance. If age influences the production of assists, it should be considered a predictor in models.

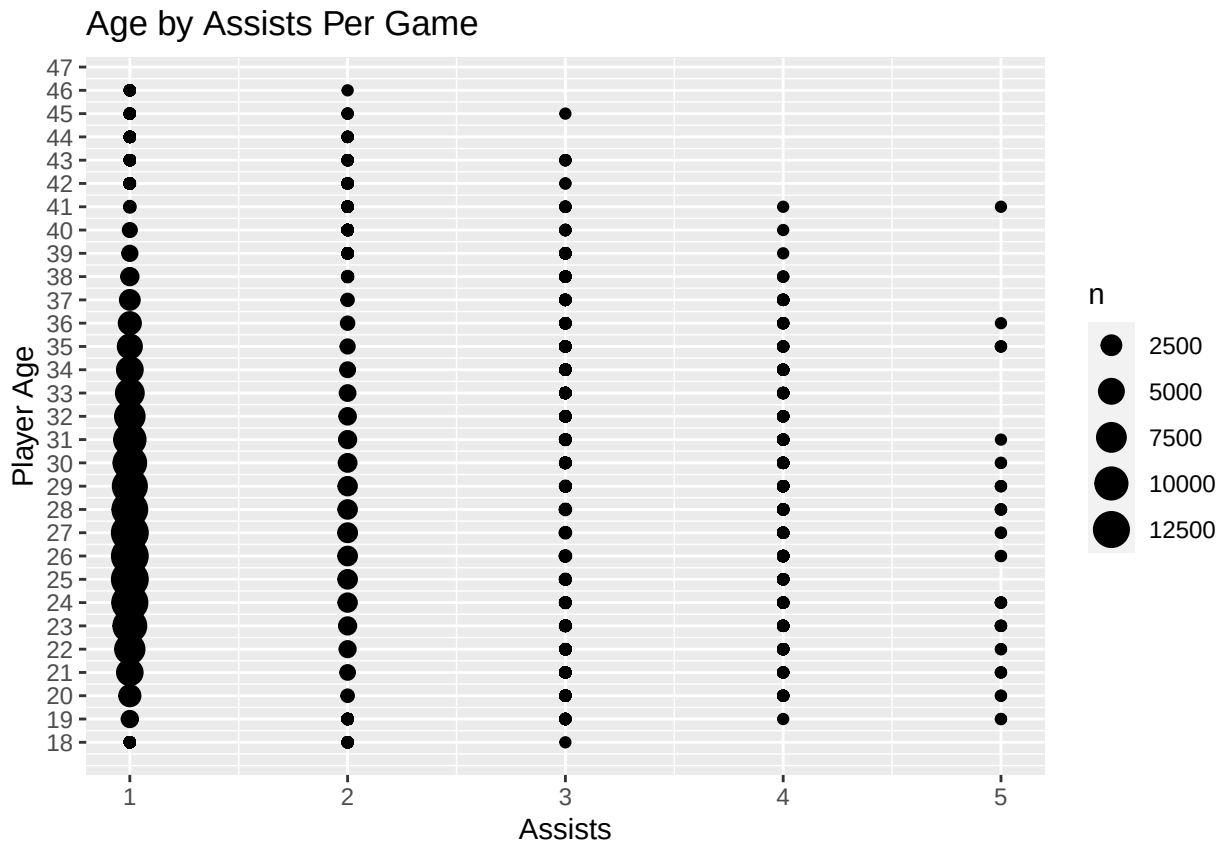
Games with 1 or more assists by a player are plotted against player age on game day. There is an obvious ramp up from age 20 to a peak in the mid-twenties, then a consistent decline into the thirties and forties.

```
# Make a variable for analyzing player age versus assists.
#  Calculate field for age as game day minus player birthday (results in days).
#  Convert the days result to years (leap years are not specially treated here).
player_game_age_assists <- players_with_points %>%
  mutate(ageAtGameDays = as.Date(date_time_GMT)-as.Date(birthDate)) %>%
  mutate(age = as.numeric(round(ageAtGameDays/365))) %>% select(age, assists)

# Since we started with players with POINTS, filter to only assists.
#  Graph games with 1+ assist counts in a histogram with default bins.
#  Add some color and outline to make it readable.
#  Mark every age on the y-axis, from 18 to 47 on the x-axis.
#  Add labels.
player_game_age_assists %>% filter(assists>0) %>%
    ggplot(aes(x=age)) +
      geom_histogram(col="blue", fill="brown") +
      scale_x_continuous(breaks = seq(18,47,1)) +
      labs(title="Games with 1+ Player Assists by Player Age"
           , y="Games with 1+ Assists"
           , x="Player Age on Game Day")
```



Drilling down to specific assist counts per game, another pattern emerges.

```
# Since we started with players with POINTS, filter to only assists.
#  Graph age by game assists counts.
#  Use a point plot.
#  Mark every age on the y-axis, from 18 to 47
#  (using a 1-step sequence) with a tick.
#  Add a dot size for count of games.
#  Add labels.
player_game_age_assists %>% filter(assists>0) %>%
  group_by(age) %>% ggplot(aes(y=age, x=assists)) +
    geom_point() +
    scale_y_continuous(breaks = seq(18,47,1)) +
    geom_count() +
    labs(title = "Age by Assists Per Game", x="Assists", y="Player Age")
```



For all games with registered assists in the database:

1. In games where a player earns only 1 assist, there is a quick ramp-up from age 19 to 20. For young players, this generally reflects their first and second year in the NHL. 18-year-old players are rare, as teams often play their rookies in the minor leagues before promoting them to major league rosters.
2. The ramp-up in 1-assist games beyond age 20 "bulges" into the mid-twenties, where a decline in league assist abundance starts, tapering into the players' early thirties, when most players retire from the league. It is important to note this is not necessarily for individual players, but league players as a group.
3. Games with 2-assists by a player follow roughly the same ramp-up and tapering as with 1-assist games, but in lower proportion.
4. Games with 3-assist show less bulging and are harder to reach for players into their early-forties.
5. 4-assist games start at age 19 and end at age 41. Their abundance among the included age groups is consistently low.
6. Elite 5-assist nights are rare and not abundant for players over 31.

Age is a clear contributor to player assists, and therefore player season assist totals. Player birth date is used as a predictor in this project's models, to represent player age effects.

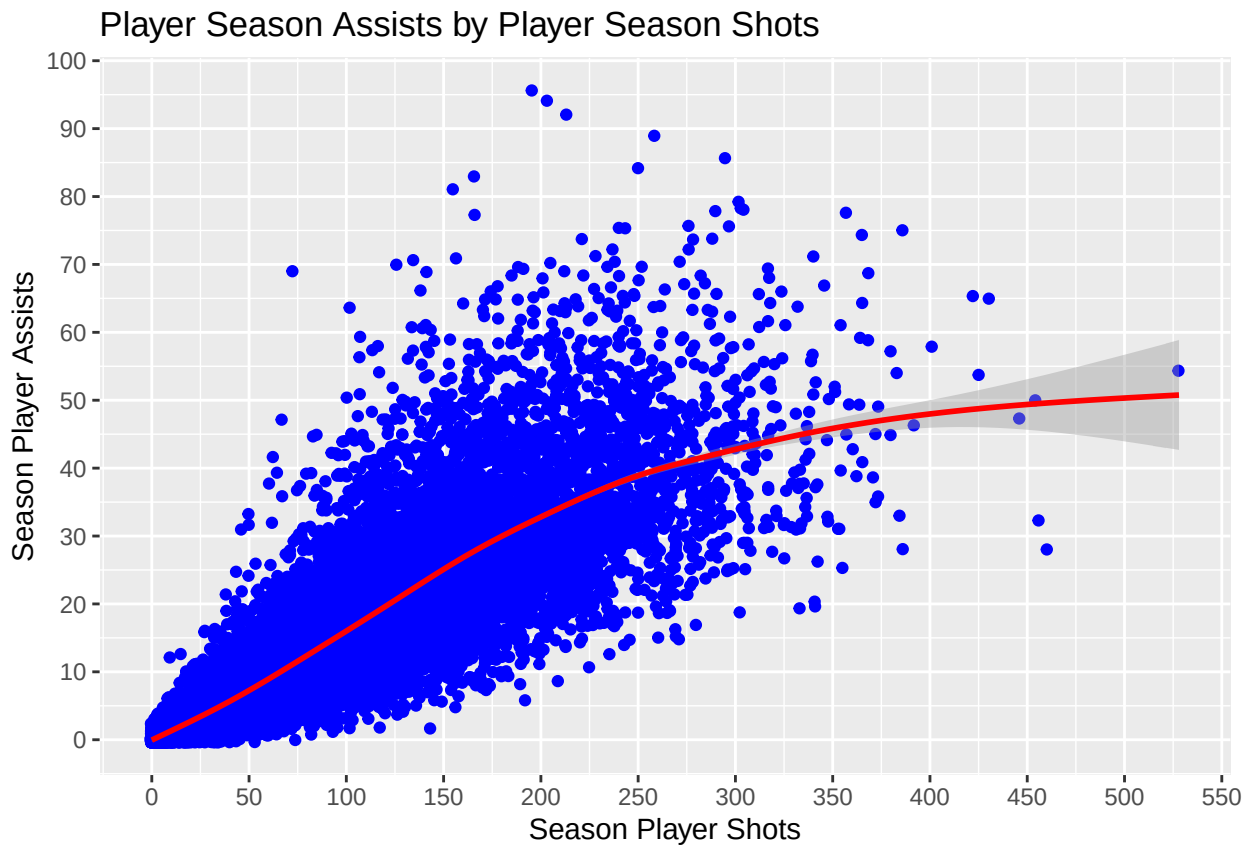
Shots

Shots are distinguished from passes, but not in statistics except by omission. There is no statistic for passes, but some apparent shots can actually be passes.

Shots are usually thought of as the intended progenitor to goals, but they can also result in assists when a teammate deflects, re-shoots, or otherwise legally directs the puck into the opposing net from a shot. The original shooter then earns an assist as opposed to a goal. Thus, shots are intuitively related to assists. This is explored briefly.

Season player assists are graphed against season player shots.

```
# Create a variable summing assists and shots by season by player.
shots_assists <- player_game_data %>% group_by(season, player_id) %>%
  summarize(sumAssists=sum(assists), sumShots=sum(shots), sumGoals=sum(goals))

# Plot player season assists vs player season shots.
#  Graph as scatter (jitter) in color.
#  Add a smoothing line with defaults.
#  Change marks on axes to more often than the default.
#  Start, end, and step for break sequences.
#  Add labels and title.
shots_assists %>%
    ggplot(aes(y=sumAssists, x=sumShots)) +
    geom_jitter(colour="blue") +
    geom_smooth(color="red") +
    scale_y_continuous(breaks = seq(0,100,10)) +
    scale_x_continuous(breaks = seq(0,600,50)) +
    labs(title="Player Season Assists by Player Season Shots"
         ,y="Season Player Assists", x="Season Player Shots")
```



There is a very strong correlation between season player shots and season player assists at 0.86, using all

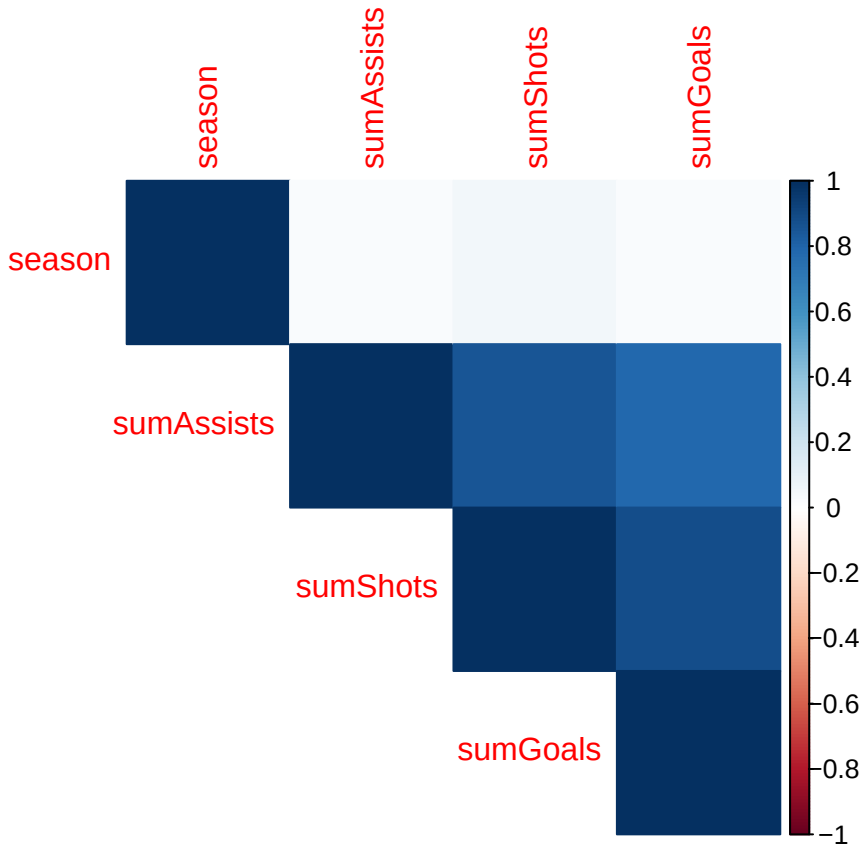
player game rows in the database. Whether the shots are intended as goals (and converted into assists by teammates instead) is not assumed. For example, It is possible that players who earn assists also shoot in proportion.

The correlation between season player assists and goals is also strong at 0.79. As expected, the season shot to season goal correlation is the strongest at 0.88.

```
# Create a correlation matrix to plot.
cor_all <- shots_assists %>% select(sumAssists, sumShots, sumGoals) %>%
  data.frame() %>% cor()

# Show correlations in a table.
cor_all  %>% round(3) %>% kable()
```

	season	sumAssists	sumShots	sumGoals
season	1.000	0.025	0.056	0.026
sumAssists	0.025	1.000	0.856	0.789
sumShots	0.056	0.856	1.000	0.883
sumGoals	0.026	0.789	0.883	1.000

```
# Plot it graphically
corrplot(cor_all, method = "color", type="upper")
```



Shots are a potential predictor of season player assists. However, since shots are also strongly correlated with goals, and assists are strongly correlated with goals, using all of these as predictors in the same model may be redundant or displace an otherwise more important predictor. This is considered in modeling.

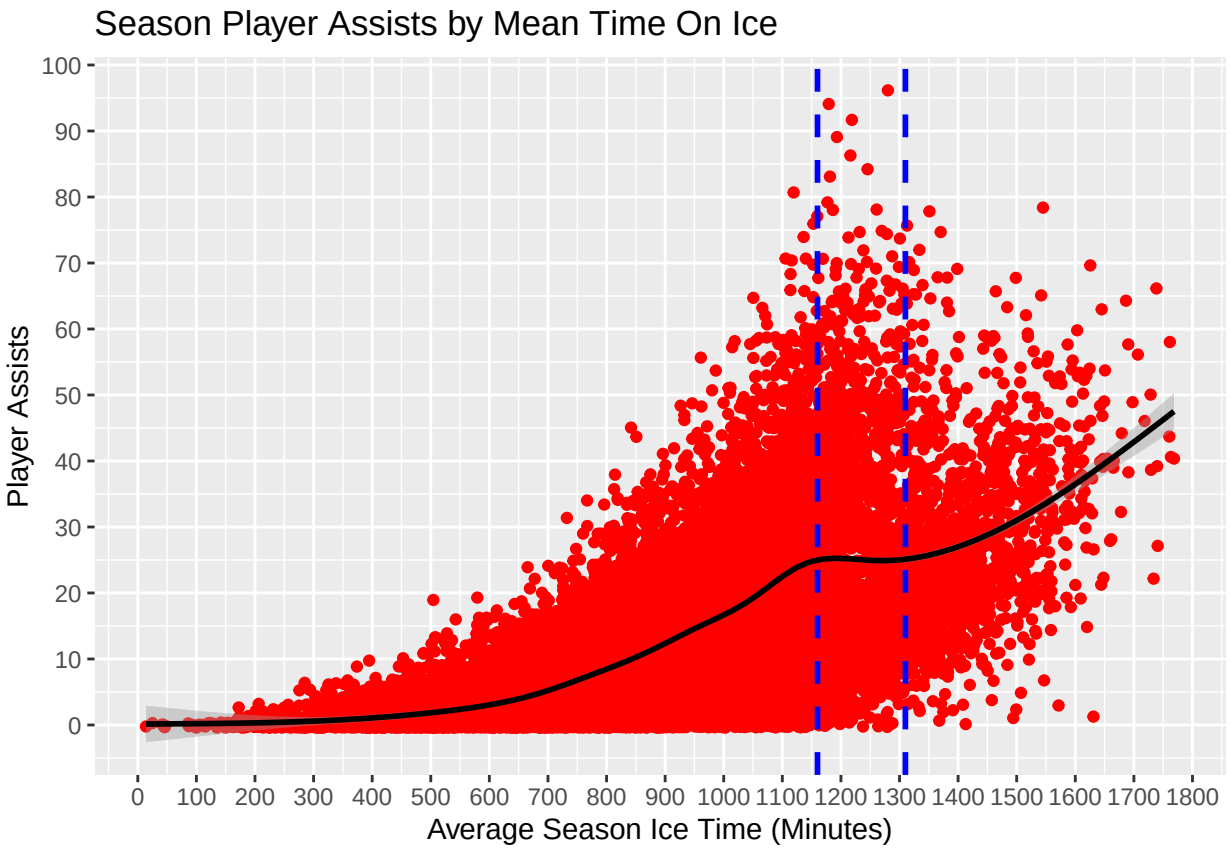
Time On Ice

A player's time on ice per game is marshaled by the coach, so there is an element of coach's judgment in this statistic. Once the player hops in, it is up to the player to execute (ie. generate assists and goals).

```
# First, calculate sum of assists and
#  mean and sum of time on ice by season and player.
toi_assists <- player_game_data %>% group_by(season, player_id) %>%
  summarize(sumAssists=sum(assists), meanTOI=mean(timeOnIce)
            ,sumTOI=sum(timeOnIce))

# Sample rows.
head(toi_assists,3)
```

```
## # A tibble: 3 x 5
## # Groups:   season [1]
##     season player_id sumAssists meanTOI sumTOI
##      <int>     <int>      <int>   <dbl>  <int>
## 1 20002001   8444894         12    863.  51806
## 2 20002001   8444919         19   1253.  96465
## 3 20002001   8445000         13    720.  53248
```

```
# Graph assists versus time on ice aggregates.
#  Use a scatter (jitter) plot with some color.
#  Add a smoothing line to show trending.
#  Change axis ticks.
#  Add vertical lines at 1160 and 1310 meanTOI.
#  to show flat segment.
#  Add labels.
toi_assists %>%
    ggplot(aes(y=sumAssists, x=meanTOI)) +
    geom_jitter(colour="red") +
    geom_smooth(colour="black") +
    scale_y_continuous(breaks = seq(0,120,10)) +
    scale_x_continuous(breaks = seq(0,2000,100)) +
    geom_vline(xintercept = 1160, color="blue"
               , linetype="dashed", size=1) +
    geom_vline(xintercept = 1310, color="blue"
               , linetype="dashed", size=1) +
    labs(title="Season Player Assists by Mean Time On Ice"
         , y="Player Assists",x="Average Season Ice Time (Minutes)")
```



Observations, following the smoothed mean line for assists, as season ice time increases:

1. More ice time is generally associated with more assists.
2. The rate of increase in assists levels to flat at around 1160 minutes of season ice time until about 1310 minutes.
3. A steeper rate of increase starts at about 1310 minutes of season ice time, carrying through to 1750 minutes.

There is a high correlation of ice time to assists, about 0.649. Ice time is included as a predictor in models.

```
# Calculate correlation.
#  Correlation is high so shots are redundant with sumAssists as a predictor.
cor(toi_assists$meanTOI, toi_assists$sumAssists)
```

```
## [1] 0.6488184
```

Nationality

NHL players are from a diverse mix of nationalities. Nationalities are player birthplaces, not the teams they play on.

1. Out of 3353 unique players in the database, Canadians are a full 48.6%.
2. 769 players are Americans, representing the 2nd largest group.

3. The remaining largest groups with over 100 players each are Swedish, Russian, Czech, and Finnish.

Player counts and percentages of league composition are tabled by nationality below:

```
# Count the unique players in the database.  Bear in mind, players
#  come and go over seasons, but this is fairly representative.
unique_player_count <- player_game_data %>%
  select(player_id) %>% unique() %>% nrow()

# Show the database player count.
unique_player_count
```

```
## [1] 3353
```

```
# Group game data by nationality, then player_id.
#  Take the unique combinations (before this step, each row is a player game).
#  Calculate counts and percentages.
player_game_data %>%  select(nationality, player_id) %>%
  unique() %>% group_by(nationality) %>%
  summarize(playerCount=n()
            , natPercent=playerCount/unique_player_count) %>%
              arrange(desc(playerCount))
```

```
## # A tibble: 28 x 3
##    nationality playerCount natPercent
##    <fct>             <int>      <dbl>
##  1 CAN                1630    0.486
##  2 USA                 769    0.229
##  3 SWE                 237    0.0707
##  4 RUS                 187    0.0558
##  5 CZE                 177    0.0528
##  6 FIN                 149    0.0444
##  7 SVK                  70    0.0209
##  8 CHE                  31    0.00925
##  9 DEU                  22    0.00656
## 10 LVA                  15    0.00447
## # ... with 18 more rows
```

When plotting season assists by nationality, we observe:

1. Canadians account for most season assists of any nationality, over the entire 19 seasons. This is not surprising, given 49% of players are Canadian (prevalence at work).
2. Americans account for the second most season assists, followed by Swedish players.
3. Other notables birthplaces are the Czech Republic, Russia, Finland, and Slovakia.
4. The other nationalities fall roughly in line with player abundance, so we examine mean assists by nationality next.

Plot notes:

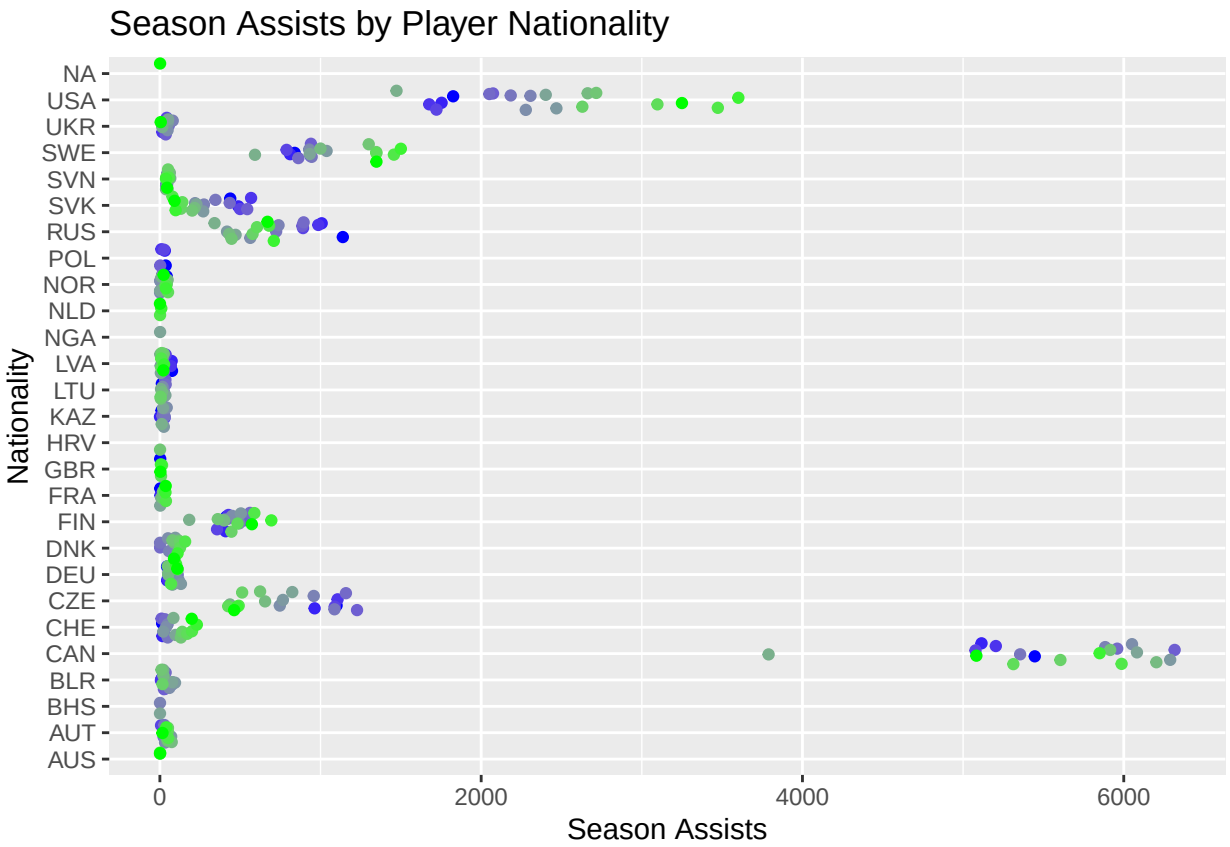
1. NA records, where no nationality is recorded for the player, are preserved to keep the integrity of the denominator (the assists they achieved occurred, and omitting their nationality from the database should not omit the assists).

2. Different colors are different seasons.

```
# Aggregate player game stats by nationality.
#  Filter to only those games where player assists occur.
#  Group by season, then nationality and calculate aggregates.
agg_nationality <- player_game_data %>%
  filter(assists > 0) %>%
  group_by(season, nationality) %>%
  summarize(nation=nationality, gameCount=n(), gamePoints = sum(points)
            , gameAssists = sum(assists), gameGoals = sum(goals)
            , meanPoints = mean(points), meanAssists = mean(assists)
            , meanGoals = mean(goals)) %>%
                unique()

# Sum seasons assists by nationality.
#  Compute average of these seasonal assists sums.
#  Show the top 7.
player_game_data %>%
  select(nationality, season, assists) %>%
  group_by(season, nationality) %>%
    summarize(sumSeasonAssists=sum(assists)) %>%
  group_by(nationality) %>%
    summarize(meanSeasonAssists=mean(sumSeasonAssists)) %>%
    arrange(desc(meanSeasonAssists)) %>% head(7)
```

```
## # A tibble: 7 x 2
##   nationality meanSeasonAssists
##   <fct>                   <dbl>
## 1 CAN                     5607.
## 2 USA                     2402.
## 3 SWE                     1045.
## 4 CZE                      795.
## 5 RUS                      694.
## 6 FIN                      457.
## 7 SVK                      283.
```

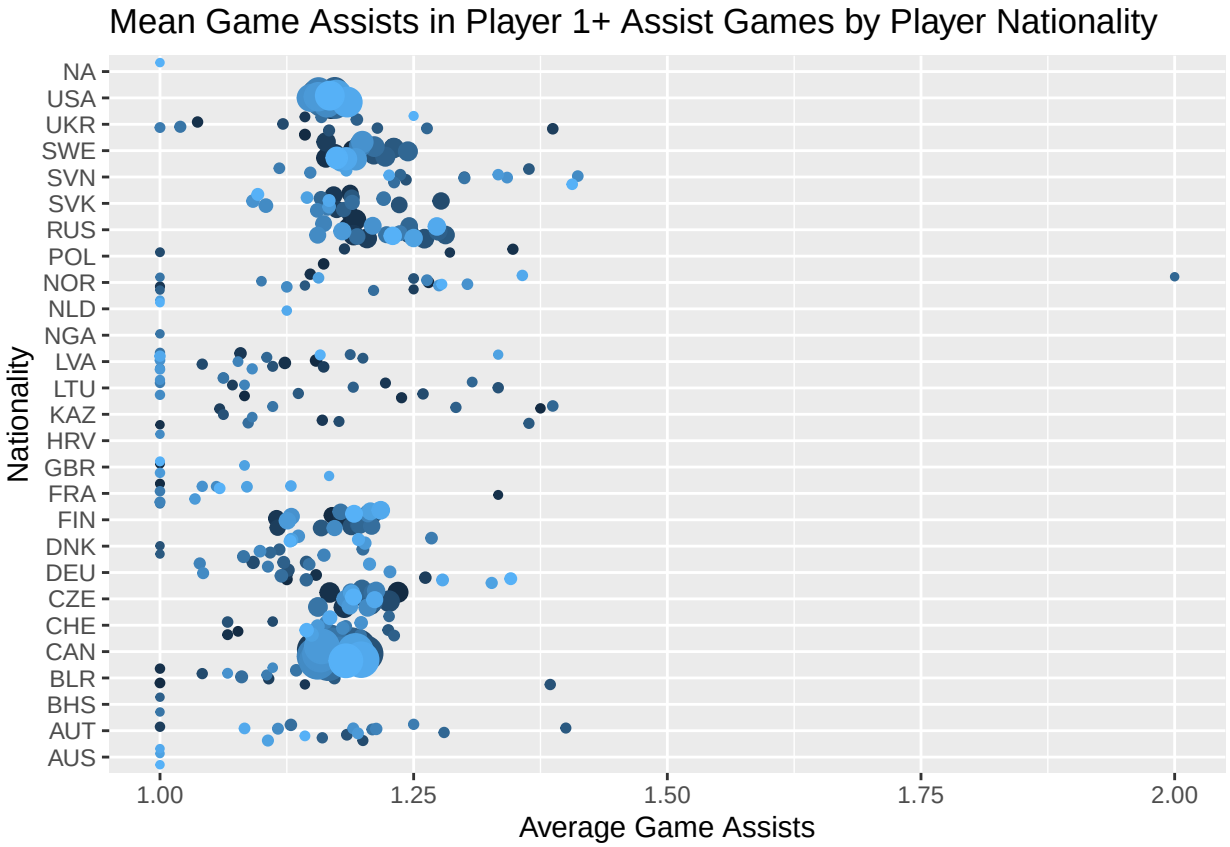
```
# Plot seasonal assist totals by player nationality.
#  Add a scatter plot geom.
#  Show colors to differentiate seasons but hide legend.
#  (Season isn't the focus of this graph).
#  Add title and labels, bearing axis flip in mind.
#  Flip the axes to show nationalities as a readable vertical list.
agg_nationality %>%
  ggplot(aes(x=nationality, y=gameAssists)) +
  geom_jitter(aes(color=season), show.legend = FALSE) +
    scale_colour_gradient(low="blue",high="green") +
    labs(title = "Season Assists by Player Nationality"
       , x="Nationality", y="Season Assists") +
      coord_flip()
```



Examining the mean game assists by nationality reveals some further insights:

1. The only Slovenian player in the NHL accounts for all of that nationality's statistics. This is Anze Kopitar, a known elite player.
2. *Prevalence*: The sheer number of Canadians and Americans dilutes superstar performers' season assist averages to below Kopitar's best seasons for assists. For example, Connor McDavid's personal averages are higher than Kopitar's, but he is a member of a much larger nationality group of 1630 Canadians. The same dilution applies to the USA statistics.
3. Out of the top 10 assist makers of any season of the last 19, 5 are Canadian, 3 are Russian, 1 is Swedish and 1 is American.

```
# Plot mean season game assists by nationality.
#  On a scatter plot, show game count by size, season by color.
#  Hide legend as specific seasons isn't the focus.
agg_nationality %>%
  ggplot(aes(x=nation, y=meanAssists)) +
  geom_jitter(aes(size = gameCount, color=season), show.legend = FALSE) +
  labs(title = "Mean Game Assists in Player 1+ Assist Games by Player Nationality"
       , x="Nationality", y="Average Game Assists") +
      coord_flip()
```




```
# Compute personal averages for Kopitar and McDavid.
#  Filter by the two players (Kopitar has played more seasons).
#  Group in order of seasons, then by the players' names.
#  Sum by this grouping to get seasonal player totals.
#  Regroup by only the player names and calculate mean of
#  their season sum of assists.
player_game_data %>%
  filter(lastName=="Kopitar" | lastName=="McDavid") %>%
  select(firstName, lastName, season, assists) %>%
  group_by(season, firstName, lastName) %>%
  summarize(sumAssists=sum(assists)) %>%
  group_by(firstName, lastName) %>%
  summarize(meanSeasonAssists=mean(sumAssists))
```

```
## # A tibble: 2 x 3
## # Groups:   firstName [2]
##   firstName lastName meanSeasonAssists
##   <fct>     <fct>                <dbl>
## 1 Anze      Kopitar               47.4
## 2 Connor    McDavid               63.8
```

```
# Us the same logic as above, without filters, to
#  show the highest means season assist players.
player_game_data %>%
  select(firstName, lastName, season, assists, nationality) %>%
```

```
  group_by(season, firstName, lastName, nationality) %>%
  summarize(sumAssists=sum(assists)) %>%
  group_by(firstName, lastName, nationality) %>%
  summarize(meanSeasonAssists=mean(sumAssists)) %>%
  arrange(desc(meanSeasonAssists)) %>% head(10)
```

```
## # A tibble: 10 x 4
## # Groups:   firstName, lastName [10]
##    firstName lastName  nationality meanSeasonAssists
##    <fct>     <fct>     <fct>                   <dbl>
##  1 Connor    McDavid   CAN                      63.8
##  2 Sidney    Crosby    CAN                      58.3
##  3 Artemi    Panarin   RUS                      57
##  4 Nicklas   Backstrom SWE                      56.7
##  5 Joe       Thornton  CAN                      56.6
##  6 Mitchell  Marner    CAN                      56.2
##  7 Nikita    Kucherov  RUS                      55
##  8 Patrick   Kane      USA                      53.2
##  9 Ray       Bourque   CAN                      52
## 10 Evgeni    Malkin    RUS                      51.6
```

Nationality appears to have some influence on season player assist production. Some of this is related to the prevalence of some nationalities over others in the league. Nevertheless, nationality may contribute to assist production as different countries have hockey development programs that emphasize different aspects of the game.

Team

Team season totals are computed for each of assists, goals and hits. The top 10 of each category are displayed.

```
# Group player game data by season and team identifiers
#  Calculate sums and averages for assists, goals and hits.
team_seasons <- player_game_data %>%
  group_by(season, team_id, teamName) %>%
  unique() %>% summarize(seasonAssists=sum(assists)
                         ,seasonGoals=sum(goals)
                         ,seasonHits=sum(hits)
                         ,seasonAvgAssists=mean(assists)
                         ,seasonAvgGoals=mean(goals)
                         ,seasonAvgHits=mean(hits)) %>%
                          ungroup()


# Sort by highest season assists by team. Show top 10.
team_seasons %>%
  arrange(desc(seasonAssists)) %>%
  head(10) %>% select(season, teamName, seasonAssists)
```

```
## # A tibble: 10 x 3
##      season teamName   seasonAssists
##       <int> <fct>              <int>
##  1 20162017 Penguins             611
```

```
##  2 20182019 Sharks               599
##  3 20172018 Capitals             585
##  4 20172018 Lightning            576
##  5 20182019 Bruins               560
##  6 20182019 Lightning            557
##  7 20132014 Blackhawks           555
##  8 20192020 Lightning            551
##  9 20102011 Canucks              550
## 10 20172018 Jets                 549
```

```
# Sort by highest season assists by team. Show top 10.
team_seasons %>%
  arrange(desc(seasonGoals)) %>%
  head(10) %>% select(season, teamName, seasonGoals)
```

```
## # A tibble: 10 x 3
##      season teamName       seasonGoals
##       <int> <fct>                <int>
##  1 20162017 Penguins               355
##  2 20182019 Sharks                 347
##  3 20172018 Capitals               342
##  4 20172018 Lightning              340
##  5 20182019 Bruins                 336
##  6 20182019 Lightning              327
##  7 20172018 Jets                   326
##  8 20102011 Bruins                 325
##  9 20172018 Golden Knights         325
## 10 20142015 Lightning              324
```

```
# Sort by highest season assists by team. Show top 10.
team_seasons %>%
  arrange(desc(seasonHits)) %>%
  head(10) %>% select(season, teamName, seasonHits)
```

```
## # A tibble: 10 x 3
##      season teamName     seasonHits
##       <int> <fct>             <int>
##  1 20132014 Kings              3691
##  2 20112012 Rangers            3084
##  3 20142015 Islanders          3002
##  4 20112012 Kings              2995
##  5 20142015 Ducks              2959
##  6 20132014 Blue Jackets       2915
##  7 20142015 Capitals           2880
##  8 20192020 Islanders          2873
##  9 20192020 Lightning          2851
## 10 20162017 Penguins           2812
```

Observations:

1. The 2016/2017 Penguins ranked at the top for both season goals and assists. They won the Stanley Cup Championship that year. There is potentially a relationship. Some of this will owe to more games played than other teams that did not progress through as many series. However, since no other team from the same season ranks in the top 10, there is likely a correlation.

2. The 2013/2014 Kings topped the last 19 seasons with the most hits by any team in any season, by a vast margin. They won the Stanley Cup that season. They did not rank in goals or assists that year. If nothing else, this demonstrates there is more than one way to win a championship. Note the 2016/2017 Penguins ranked 10th in hits for the same championship season they topped assist and goals.

3. On the top 10 list for seasons in the 19, the Lightning appeared 3 times in the top 10 for assists, 3 times for goals, and once for hits. They won the Stanley Cup Championship in 2019/2020.

4. If this project targeted Stanley Cup probabilities for teams, all three of these variables would be highly relevant. This is testament to the value of the target response variable.

5. When *team season assist means* are calculated, their distribution is wider than the distribution of overall *team assist means* without regard to season (measured across all seasons). Teams have been added and moved, but their impact on any one season is diluted in the scheme of all seasons.

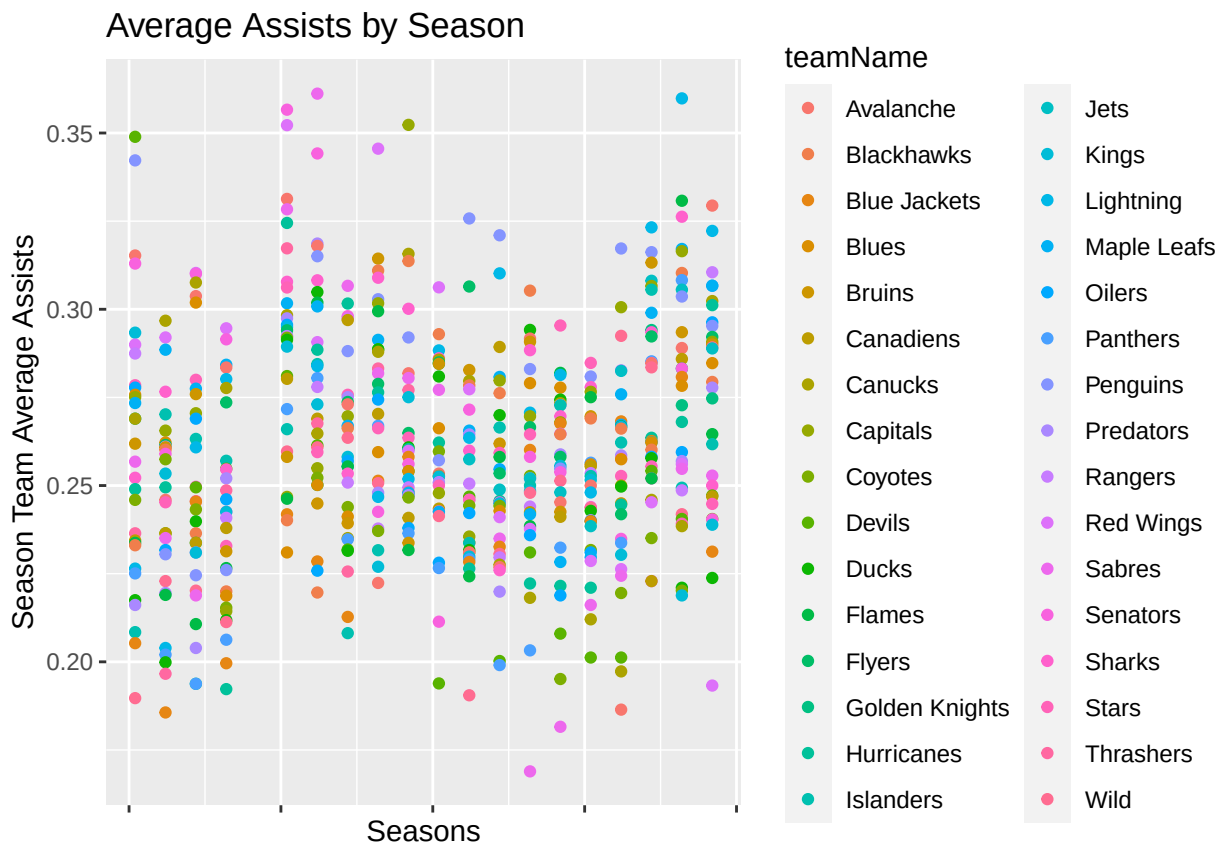
```
# Compute average season team assists.
#  Eliminate null records from previous grouping operations.
ta <- team_seasons %>% select(season, teamName, seasonAvgAssists) %>%
  filter(!is.na(teamName)) %>%
  arrange(desc(seasonAvgAssists))

# Summarize assists.
ta$seasonAvgAssists %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1689  0.2398  0.2578  0.2606  0.2825  0.3612
```

```
# Standard deviation with season separation.
ta$seasonAvgAssists %>% sd()
```

```
## [1] 0.0321092
```

```
# Graph season average assists by team
ta %>% ggplot(aes(x=season, y=seasonAvgAssists, color=teamName)) +
       geom_point() +
       theme(axis.text.x = element_blank()) +
       labs(y="Season Team Average Assists"
            , x="Seasons"
            , title = "Average Assists by Season")
```




```
# Average team assists over 19 seasons.
all_team_assists <-player_game_data %>%
  group_by(teamName) %>%
  summarize(meanTeamAssists=mean(assists)) %>%
  arrange(desc(meanTeamAssists)) %>%
  filter(!is.na(teamName))

# List team assists over 19 seasons.
all_team_assists %>% arrange(desc(meanTeamAssists))
```

```
## # A tibble: 32 x 2
##    teamName       meanTeamAssists
##    <fct>                    <dbl>
##  1 Penguins                 0.288
##  2 Golden Knights           0.281
##  3 Sharks                   0.278
##  4 Capitals                 0.276
##  5 Lightning                0.276
##  6 Red Wings                0.275
##  7 Maple Leafs              0.275
##  8 Avalanche                0.273
##  9 Senators                 0.272
## 10 Jets                     0.272
## # ... with 22 more rows
```

```
# Distribution of assists over 19 seasons.
all_team_assists$meanTeamAssists %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2403  0.2527  0.2598  0.2617  0.2722  0.2879
```

```
# SD of assists,
all_team_assists$meanTeamAssists %>% sd()
```

```
## [1] 0.01286576
```

The differences in team assists means over 19 seasons are minimal.

```
# Delete team identifiers from core data.
player_game_data <- player_game_data %>% select(-team_id, -teamName)
```

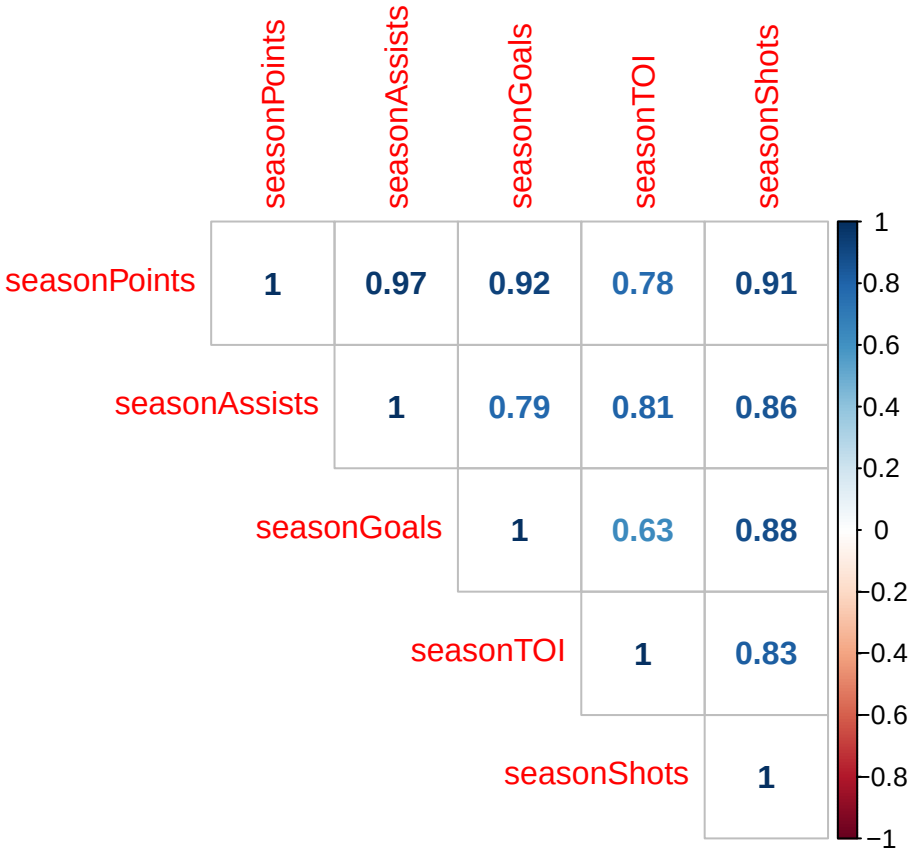
Variable Correlations

The quantity dimensions are plotted in a correlation matrix below. This can help identify which predictors could be redundant if used in a model together. Highly correlated dimensions can cause some models to run less efficiently or worse, displace more relevant predictors.

```
# Create a subset of the player game data to correlate.
#  Make it a tibble.
cc <- player_game_data %>%
  select(timeOnIce, assists, goals, shots, hits, points) %>%
  as_tibble()

# Create an object grouped by season, then player.
#  Aggregate the stats to correlate.
cd <- player_game_data %>% group_by(season, player_id) %>%
  summarize(  pointGamesCount=n(), seasonPoints = sum(points)
            , seasonAssists = sum(assists), seasonGoals = sum(goals)
            , seasonTOI = sum(timeOnIce), seasonShots=sum(shots) )%>%
              unique()

# Create a correlation matrix for all fields
cm <- cor(cd[,-c(1,2,3)])
cm
```

```
##               seasonPoints seasonAssists seasonGoals seasonTOI seasonShots
## seasonPoints     1.0000000     0.9652071   0.9219667 0.7771922   0.9149618
## seasonAssists    0.9652071     1.0000000   0.7886230 0.8064765   0.8562079
## seasonGoals      0.9219667     0.7886230   1.0000000 0.6331262   0.8834332
## seasonTOI        0.7771922     0.8064765   0.6331262 1.0000000   0.8278657
## seasonShots      0.9149618     0.8562079   0.8834332 0.8278657   1.0000000
```

```
# Plot correlation matrix.
corrplot(cm, method = "number", type = "upper")
```



Data Preparation for Models

The above data wrangling and exploration created many variables for specific section usage. This section focuses on preparing the data specifically for modeling.

Season Aggregations

First, season data is summed up for the meaningful dimensions explored.

```
# Select the fields we need for models.
#  Group first by season, then player.
#  Total and average based on this grouping hierarchy.
season_data <- player_game_data %>%
  select(player_id, firstName, lastName, season
                              , game_id, nationality
                              , points, goals, assists, shots
                              , hits, timeOnIce) %>%
  group_by(season, player_id) %>% summarise(sumPoints=sum(points)
                                          , sumAssists=sum(assists)
                                          , sumGoals=sum(goals)
                                          , sumShots=sum(shots)
                                          , sumTimeOnIce=sum(timeOnIce)
                                          , meanAssists=mean(assists)
                                          , meanGoals=mean(goals)
```

```
                                          , meanShots=mean(shots)
                                          , meanTimeOnIce=mean(timeOnIce))
# Show sample rows.
season_data %>% arrange(desc(sumPoints)) %>% head(4)
```

```
## # A tibble: 4 x 11
## # Groups:   season [3]
##   season player_id sumPoints sumAssists sumGoals sumShots sumTimeOnIce
##    <int>     <int>     <int>      <int>    <int>    <int>        <int>
## 1 2.02e7   8476453       130         89       41      258       102616
## 2 2.01e7   8466138       125         96       29      195       103696
## 3 2.01e7   8467875       124         74       50      365       121587
## 4 2.01e7   8448208       123         69       54      368       108609
## # ... with 4 more variables: meanAssists <dbl>, meanGoals <dbl>,
## #   meanShots <dbl>, meanTimeOnIce <dbl>
```

Next, "current season" and "next season" player assist totals are put on the same row so that R models can target "next season player total assists" to predict. It would be pointless to predict "current season total assists" as they are already known at the end of each season. The purpose of our model is to *predict* player assist totals for the entire subsequent season, once the current season totals are available:

1. A "next season assist total" column is defined for each player. The season player totals table is joined back to a copy of itself with the seasons incremented, resulting in single rows having "current season assists" and "next season assists" together.
2. Since the 2004/2005 season was canceled (there is no data), the incremental "next season" for 2003/2004 must be set to 2005/2006.
3. Player retirements and injuries create some NA values for "next season" totals. These are dealt with at the modeling step.
4. The final season in the training data will technically derive "next season" player assist totals from the first season of the test dataset (explained in the *Training, Testing & Validation Datasets* section below).
5. The validation set contains the 3 most recent complete seasons, but 2019/2020 does not have a complete "next season" to draw on yet, as of this writing. The 2018/2019 records pull in sumAssists from 2019/2020 as nextSumAssists, but the rest of the 2019/2020 data is ignored.

The technical steps to accomplish the above are commented comprehensively in the code.

```
# Prepare a "next season" data set.
#  Create a join_season field to avoid confusion.
#  This will represent the "prior season" in the join at a later step.
#  Only for season 20052006, the prior season is forced to
#  20032004 because 20042005 was canceled.
#  Rename season field to next_season because that's what it
#  will become once joined back to the main data, below.
#  Select only the needed fields to avoid duplication in the join.
next_season_data <- season_data %>%
  mutate(join_season=ifelse(season==20052006, 20032004, season-10001)) %>%
  rename(next_season=season, nextSumAssists=sumAssists) %>%
  select(join_season, next_season, player_id, nextSumAssists)
```

```r
# Preview some rows.
next_season_data %>% head(3)
```

```
## # A tibble: 3 x 4
## # Groups:   next_season [1]
##   join_season next_season player_id nextSumAssists
##         <dbl>       <int>     <int>          <int>
## 1    19992000    20002001   8444894             12
## 2    19992000    20002001   8444919             19
## 3    19992000    20002001   8445000             13
```

```r
# Join main data back to next season data.
agg_data <- season_data %>%
  full_join(next_season_data, by=c("season"="join_season", "player_id"="player_id"))

# Display relevant rows to demonstrate prediction data structure.
#  Note season versus next season.
agg_data %>% arrange(desc(sumAssists)) %>%
  select(player_id, season, sumAssists, next_season, nextSumAssists)
```

```
## # A tibble: 20,656 x 5
## # Groups:   season [20]
##    player_id   season sumAssists next_season nextSumAssists
##        <int>    <dbl>      <int>       <int>          <int>
##  1   8466138 20052006         96    20062007             92
##  2   8467876 20102011         94    20112012             70
##  3   8466138 20062007         92    20072008             67
##  4   8476453 20182019         89    20192020             79
##  5   8471218 20172018         86    20182019             75
##  6   8471675 20062007         84    20072008             48
##  7   8467876 20092010         83    20102011             94
##  8   8466138 20152016         81    20162017             45
##  9   8476453 20192020         79          NA             NA
## 10   8471215 20082009         78    20092010             49
## # ... with 20,646 more rows
```

```r
# Filter out records with null next_seasons.  They have no
#  values for next season assist totals to
#  compare predictions against.  These will be 20192020 stats.
agg_data <- agg_data %>% filter(!is.na(next_season))

# Unique player count in new dataset.
agg_data$player_id %>% unique %>% length()
```

```
## [1] 3353
```

```r
# Confirm that unique player count has not changed from the
#  original data set.
agg_data$player_id %>%
  unique %>%
  length() == player_game_data$player_id %>%
```

```
              unique() %>%
              length()
```

```
## [1] TRUE
```

```
# Because of the construction method of these data sets, the year
#  before a rookie joins will show the previous year (when he did not play)
#  with a "next year" nextSumAssists value for him but no "current year"
#  sumAssists or any other predictors.  Remove these records.
#  Filter out rows with null current year predictors generated by data
#  handling.  These nulls were not in the original set.
agg_data <- agg_data %>% filter(!is.na(sumAssists))

# Calculate ending row count.
nrow(agg_data)
```

```
## [1] 12918
```

There are 12,918 summary rows ready for partitioning.

Training, Testing & Validation Datasets

Since we are creating a model to predict *Next Season Player Assists* the test dataset and the validation dataset should be further ahead in time than the training dataset. For the model to be useful, it should be tested using "historical data" to predict "future data."

While there are theoretical use cases, it would be completely unrealistic to use future data to predict past hockey season results (which we already have in hand) or to randomize the timing of the data used to test and validate.

Player age, experience, rule changes between seasons, team additions, and position changes (often after trades) are relevant predictors, as explored above, and we live with linear time progression. While using an older version of a player to predict his younger version's performance could be useful if we were predicting missing historical data, this study is about predicting future results. Data is partitioned with this in mind, as follows:

1. The test set uses seasons 2014/2015, 2015/2016, and 2016/2017. The 2016/2017 season introduced a new team to the league, adjusting league game counts. This forced the models to deal with slightly changed aggregations and accounts for a little more than a third of the test data.
2. The validation set consists of the most recent seasons in the database: 2017/2018, 2018/2019, and 2019/2020 (also a short season due to COVID-19). The inclusion of a short season in the validation set forces the final model to validate under real world conditions, where seasons have suddenly shortened because of lockouts and a pandemic. The final season of 2019/2020 contributes "next season" player season assist totals to the 2018/2019 season.
3. The 2019/2020 season itself could not be used to validate because there is no complete 2020/2021 data to compare predicted totals against (although this would be a great real world application once the data is available at season's end).
4. The training set uses all seasons that were not in the test set or validation set.
5. Random sampling *within the training set* is used to address hardware limits, only where required.

6. Only the final model (showing the highest predictive value with the lowest RMSE against the test set) is validated against the validation data set.

```
# Establish range covered by data set
# 2000 to 2019.
agg_data$season %>% unique()
```

```
##  [1] 20002001 20012002 20022003 20032004 20052006 20062007 20072008 20082009
##  [9] 20092010 20102011 20112012 20122013 20132014 20142015 20152016 20162017
## [17] 20172018 20182019
```

```
# Set 20172018 to 20192020 as validation
validation <- agg_data %>%
  filter(season==20172018 | season==20182019 | season==20192020)

# Use 20142015 to 20162017 as test.
test_set <- agg_data %>%
  filter(season==20142015 | season==20152016 | season==20162017)

# The rest is for training.
train_set <- agg_data %>%
  anti_join(test_set) %>%
  anti_join(validation)

# Clear NA records generated by previous grouping.
train_set <- train_set %>% filter(!is.na(sumAssists))
test_set <- test_set %>% filter(!is.na(sumAssists))
validation <- validation %>% filter(!is.na(sumAssists))

# There will be some rookies in the test set years and validation set
#  years that will not be in the training years.  Unfortunately, because
#  of time linearity, we must remove them because prediction of rookie
#  stats with no history is tough for our models, which are intended to
#  predict season assists from using historical data for known players.
#
# If we put the rookie or retired player data into the training set the seasonal
#  aggregates for certain groupings will be disturbed, so we omit them altogether.
#  Execute on kept validation rows.
validation <- validation %>% semi_join(train_set, by = "player_id")

# Do the same for the training set.
test_set <- test_set %>% semi_join(train_set, by = "player_id")

# Show sample rows in training set.
train_set %>% head(3)
```

```
## # A tibble: 3 x 13
## # Groups:   season [1]
##   season player_id sumPoints sumAssists sumGoals sumShots sumTimeOnIce
##    <dbl>     <int>     <int>      <int>    <int>    <int>        <int>
## 1 2.00e7   8444919        20         19        1       69        96465
## 2 2.00e7   8445000        33         13       20      119        53248
## 3 2.00e7   8445176        79         45       34      225        90473
```

```
## # ... with 6 more variables: meanAssists <dbl>, meanGoals <dbl>,
## #   meanShots <dbl>, meanTimeOnIce <dbl>, next_season <int>,
## #   nextSumAssists <int>
```

```
# Rowcounts
c(nrow(train_set), nrow(test_set), nrow(validation))
```

```
## [1] 9248 1561  702
```

Models

RMSE Model Performance Measure

This project uses the residual mean squared error (RMSE) method as a performance measure for models. This formula represents the "error level" of the model being tested by comparing predictions with actuals.

1. The errors for each prediction-actual pair are squared and summed, partly to ensure both negative and positive errors are equally weighted.
2. The square root of this sum represents the average error produced by the model in units of the response variable. In this case, the units are *Player Season Assists* over an entire NHL hockey season.
3. In this project, the target RMSE is 10.00. This is less than the standard deviation of 13.76 for player season assists of all players in the database of 19 NHL seasons.
4. The interquartile range of season player assists is between 2 and 20 assists, as shown in the statistics summary.

```
# Summarize player season stats.
player_season_summary$sumAssists %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    9.00   13.28   20.00   96.00
```

```
# Calculate standard deviation of all player seasons.
sd(player_season_summary$sumAssists)
```

```
## [1] 13.75763
```

A function is created to evaluate models:

```
# Build the RMSE function to evaluate models.
RMSE <- function(actual_season_assists, predicted_season_assists){
  sqrt(mean((actual_season_assists - predicted_season_assists)^2))
}
```

Aggregation Terminology

All statistics in the modeling section are summarized by season and player. Singular terms are used for convenience but they refer to their season player aggregates. For example, *total shots* in this section refers to the total shots for a player over an entire season.

rt1: Regression Tree Model: Time on Ice, Shots, Assists

The first model attempted is a regression tree model using 3 predictors: mean time on ice, total shots, and total assists. This model was chosen as a starting point because:

1. When these predictors were analyzed (at the player game level, before aggregation to player season level), they showed potential clustering with wide variations within groups. For example, game assists by elite players have a group average wildly above the mean of all other players. A regression tree can partition such groups, first on the basis of the strongest separating dimension, then onto the resulting binary splits. Branches are recursively split into further branches by the same method (minimizing the total SSE at each level). A regression tree model was chosen for these properties. Model tuning based on results is discussed in sections below.
2. A decision tree model can help rank the most influential predictors for further iterations using regression tree models or other types of models.
3. These quantitative predictors are suitable for a regression tree using analysis of variance (ANOVA) to output a numerical estimate (as opposed to classification trees to output a classification).
4. Computation should be relatively reasonable using all 13 seasons of training data without resorting to sampling.
5. Decision trees can handle quantities of different scales. The player season level summations computed both sum and mean measures. While these are denominated in the same units (assists), they are technically of different scales because the denominator used in calculating means depends on an observation count for a player in a season whereas a sum is over the entire season. The sum is more sensitive to canceled games, shortened seasons, and player injuries

```
# Train regression tree model 1 using ice time, shots, and assists.
#  Game level predictors for season stats.
fit_rt1 <- rpart(nextSumAssists~meanTimeOnIce + sumShots + sumAssists
                , method="anova", data=train_set)
```

The resulting model rules are printed and diagrammed below.

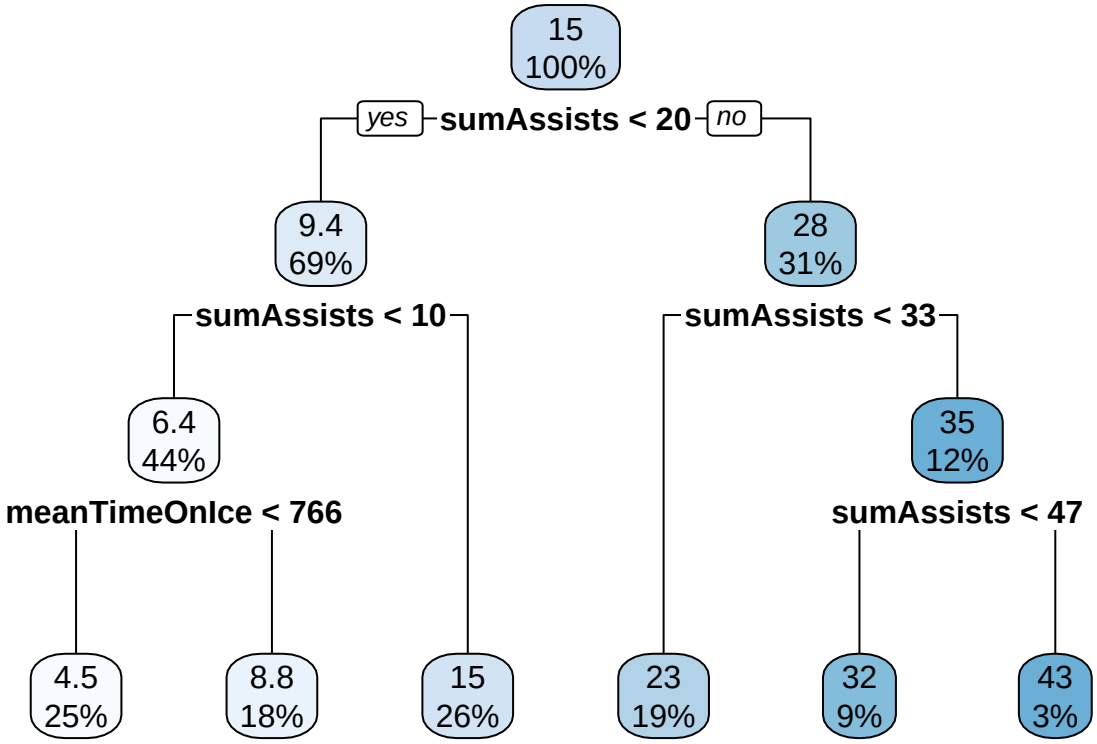
```
# Print decision rules, two different ways.
print(fit_rt1)
```

```
## n= 9248
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 9248 1730500.00 15.059260
##    2) sumAssists< 19.5 6389  517302.10  9.407106
##      4) sumAssists< 9.5 4025  201858.10  6.350311
##        8) meanTimeOnIce< 766.3211 2318   74921.62  4.528473 *
##        9) meanTimeOnIce>=766.3211 1707  108795.30  8.824253 *
##      5) sumAssists>=9.5 2364  213799.50 14.611680 *
##    3) sumAssists>=19.5 2859  552969.40 27.690100
##      6) sumAssists< 32.5 1747  226659.60 23.113340 *
##      7) sumAssists>=32.5 1112  232225.10 34.880400
##       14) sumAssists< 46.5 811  132554.30 31.848340 *
##       15) sumAssists>=46.5 301   72126.25 43.049830 *
```

```
rpart.rules(fit_rt1)
```

```
##  nextSumAssists
##             4.5 when sumAssists <  10       & meanTimeOnIce <  766
##             8.8 when sumAssists <  10       & meanTimeOnIce >= 766
##            14.6 when sumAssists is 10 to 20
##            23.1 when sumAssists is 20 to 33
##            31.8 when sumAssists is 33 to 47
##            43.0 when sumAssists >=       47
```

```
# Show tree logic.
rpart.plot(fit_rt1)
```



Analysis of the resulting tree:

1. If the model went no further than the root node, a prediction of 15 would be generated for every player.
2. If current season assists are lower than 20, the final prediction would be 9.4 if this were a terminal node. If so, 69% of all predictions would be 9.4, with the remaining 31% estimated at 28 next season player assists. But the model does not stop after the first question of whether assists are lower than 20.
3. For the next branch on the left, if current season assists total less than 10 for a player, the prediction is 6.4 if the tree were to end here, but it keeps going.

4. Without listing every step here, the tree produces 6 possible predictions (listed below). Each player will receive one of these predictions for the next season assists field.

The RMSE falls within 9.81895 player season assists. This is already lower than the target of 10.00.

```
# Generate predictions against test set.
predictions_rt1 <- predict(fit_rt1, newdata=test_set)

# Show possible prediction outputs.
unique(predictions_rt1) %>% head()
```

```
## [1] 23.113337  4.528473  8.824253 14.611675 43.049834 31.848335
```

```
# Calculate RMSE.  Store for summary at end.
(rmse_rt1 <- RMSE(test_set$nextSumAssists, predictions_rt1))
```

```
## [1] 9.81895
```

Variable importance is a relative ranking rather than an absolute score. Removing one variable from the model can affect the re-ranking and scoring of another, especially if they are correlated. The rank cannot be read as an absolute importance of variables, just a ranking for a particular model.

1. Total player season assists from the current season is ranked as the most important predictor.
2. Total shots is the second most important variable. Although we know intuitively that assists can come from shots, they can also come from passes and deflections. Theoretically, player shots are intended to convert to goals more than to assists.
3. Average time on ice is third. Coach judgment is built into ice-time as coaches select game day rosters and authorize their ice times.

```
# Show variable importance in descending rank.
varImp(fit_rt1) %>% arrange(desc(Overall))
```

```
##                 Overall
## sumAssists    0.9429841
## sumShots      0.6406305
## meanTimeOnIce 0.5692116
```

Drilling down further on how the rpart package determined the model:

1. The reported complexity parameter (CP) for each split shows the RSS benefit (how much the sum of the squared errors between model prediction and actuals is reduced) at each added split. This is expressed as a percentage of improvement over the current RSS (calculated as if the current split was the final model, with the improvement as a proportion).
2. The largest RSS benefit was achieved at the first split. Since no minimum CP was specified for the model as a parameter, the model continued splitting until any further benefit ran out.
3. A tree with 5 splits returns the lowest standard deviation (xstd). Five splits from the root node end in 6 nodes of possible prediction, as above.

4. The error at each split is scaled to 1.00000 in the error column and cascaded down proportionately to further splits.

5. The rpart functions automatically selected the optimal number of splits (5 splits with 6 final nodes at the bottom of the final tree, each with one prediction), based on an internal cross-validation process in the rpart package and descending relative errors for the feature examined by each split, to determine the optimal complexity parameter.

6. Of all splits tried, 5 splits returned the highest R-square, representing the strongest model.

```
# Complexity parameter cross-validation.
summary(fit_rt1)
```

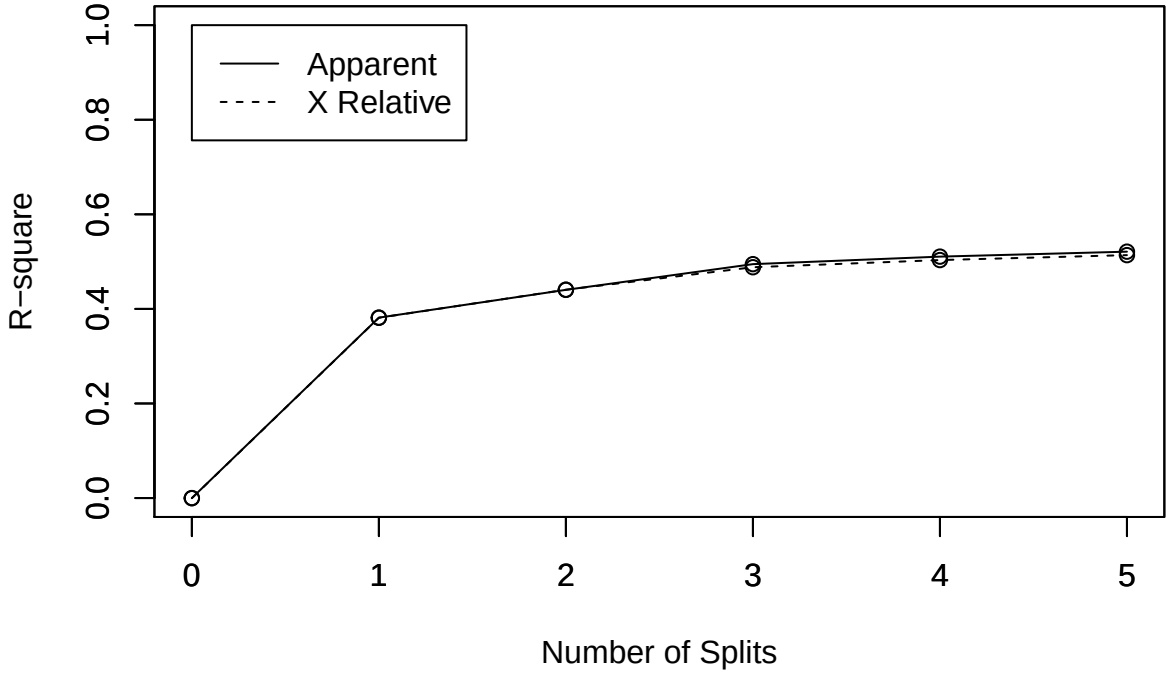
```
## Call:
## rpart(formula = nextSumAssists ~ meanTimeOnIce + sumShots + sumAssists,
##     data = train_set, method = "anova")
##   n= 9248
##
##           CP nsplit rel error    xerror        xstd
## 1 0.38152451      0 1.0000000 1.0002957 0.018892863
## 2 0.05873711      1 0.6184755 0.6187569 0.012397213
## 3 0.05436857      2 0.5597384 0.5594151 0.011983863
## 4 0.01591708      3 0.5053698 0.5118758 0.010644500
## 5 0.01048320      4 0.4894527 0.4967373 0.010015473
## 6 0.01000000      5 0.4789695 0.4863127 0.009874346
##
## Variable importance
##    sumAssists      sumShots meanTimeOnIce
##            59            30            12
##
## Node number 1: 9248 observations,    complexity param=0.3815245
##   mean=15.05926, MSE=187.1215
##   left son=2 (6389 obs) right son=3 (2859 obs)
##   Primary splits:
##       sumAssists    < 19.5     to the left,  improve=0.3815245, (0 missing)
##       sumShots      < 103.5    to the left,  improve=0.3277715, (0 missing)
##       meanTimeOnIce < 923.796  to the left,  improve=0.2406423, (0 missing)
##   Surrogate splits:
##       sumShots      < 127.5    to the left,  agree=0.864, adj=0.559, (0 split)
##       meanTimeOnIce < 1042.113 to the left,  agree=0.754, adj=0.203, (0 split)
##
## Node number 2: 6389 observations,    complexity param=0.05873711
##   mean=9.407106, MSE=80.96762
##   left son=4 (4025 obs) right son=5 (2364 obs)
##   Primary splits:
##       sumAssists    < 9.5      to the left,  improve=0.1964897, (0 missing)
##       sumShots      < 51.5     to the left,  improve=0.1732726, (0 missing)
##       meanTimeOnIce < 749.7221 to the left,  improve=0.1475223, (0 missing)
##   Surrogate splits:
##       sumShots      < 66.5     to the left,  agree=0.846, adj=0.585, (0 split)
##       meanTimeOnIce < 923.2859 to the left,  agree=0.702, adj=0.194, (0 split)
##
## Node number 3: 2859 observations,    complexity param=0.05436857
##   mean=27.6901, MSE=193.4136
```

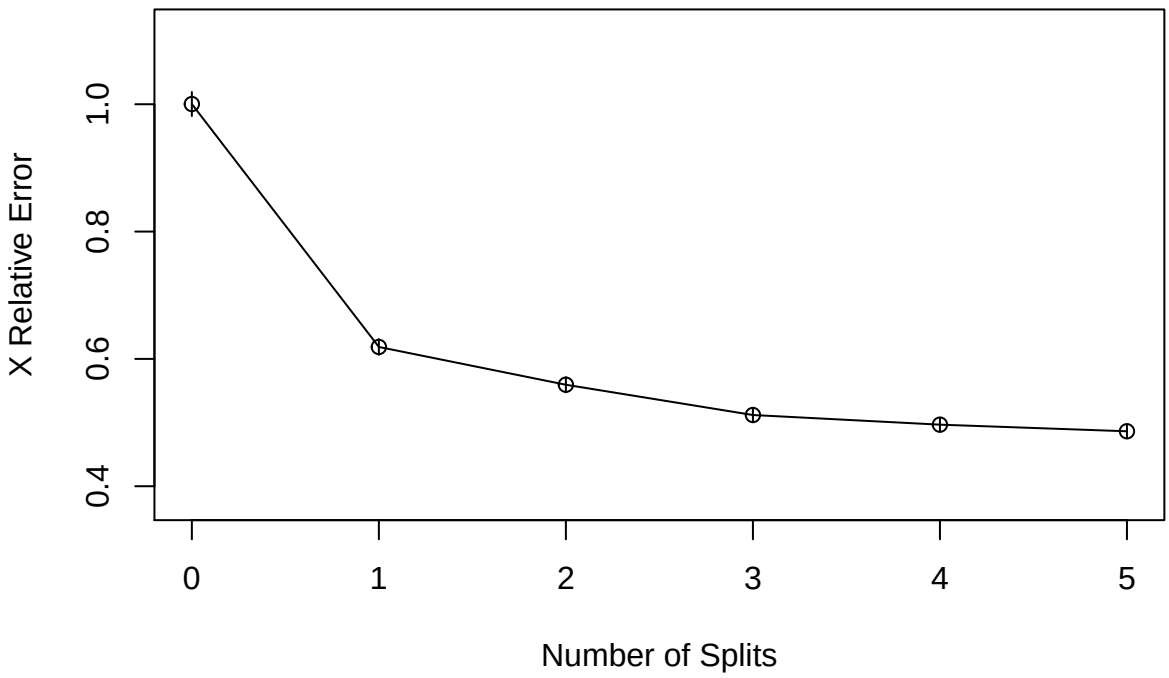
```
##   left son=6 (1747 obs) right son=7 (1112 obs)
##   Primary splits:
##       sumAssists    < 32.5     to the left,  improve=0.17014460, (0 missing)
##       sumShots      < 178.5    to the left,  improve=0.06145009, (0 missing)
##       meanTimeOnIce < 1079.272 to the left,  improve=0.06123062, (0 missing)
##   Surrogate splits:
##       sumShots      < 217.5    to the left,  agree=0.697, adj=0.222, (0 split)
##       meanTimeOnIce < 1116.87  to the left,  agree=0.623, adj=0.031, (0 split)
##
## Node number 4: 4025 observations,    complexity param=0.0104832
##   mean=6.350311, MSE=50.15107
##   left son=8 (2318 obs) right son=9 (1707 obs)
##   Primary splits:
##       meanTimeOnIce < 766.3211 to the left,  improve=0.08987090, (0 missing)
##       sumAssists    < 2.5      to the left,  improve=0.07621403, (0 missing)
##       sumShots      < 32.5     to the left,  improve=0.06597091, (0 missing)
##   Surrogate splits:
##       sumAssists < 4.5      to the left,  agree=0.667, adj=0.214, (0 split)
##       sumShots   < 27.5     to the left,  agree=0.626, adj=0.117, (0 split)
##
## Node number 5: 2364 observations
##   mean=14.61168, MSE=90.43973
##
## Node number 6: 1747 observations
##   mean=23.11334, MSE=129.7422
##
## Node number 7: 1112 observations,    complexity param=0.01591708
##   mean=34.8804, MSE=208.8355
##   left son=14 (811 obs) right son=15 (301 obs)
##   Primary splits:
##       sumAssists    < 46.5     to the left,  improve=0.11861120, (0 missing)
##       meanTimeOnIce < 1126.415 to the left,  improve=0.02994549, (0 missing)
##       sumShots      < 223.5    to the left,  improve=0.01216541, (0 missing)
##   Surrogate splits:
##       sumShots      < 337.5    to the left,  agree=0.737, adj=0.03, (0 split)
##       meanTimeOnIce < 1682.962 to the left,  agree=0.732, adj=0.01, (0 split)
##
## Node number 8: 2318 observations
##   mean=4.528473, MSE=32.32167
##
## Node number 9: 1707 observations
##   mean=8.824253, MSE=63.73478
##
## Node number 14: 811 observations
##   mean=31.84834, MSE=163.4456
##
## Node number 15: 301 observations
##   mean=43.04983, MSE=239.6221
```

```
# Plot R-square by number of splits.
rsq.rpart(fit_rt1)
```

```
##
## Regression tree:
```

```
## rpart(formula = nextSumAssists ~ meanTimeOnIce + sumShots + sumAssists,
##     data = train_set, method = "anova")
##
## Variables actually used in tree construction:
## [1] meanTimeOnIce sumAssists
##
## Root node error: 1730500/9248 = 187.12
##
## n= 9248
##
##         CP nsplit rel error  xerror      xstd
## 1 0.381525      0   1.00000 1.00030 0.0188929
## 2 0.058737      1   0.61848 0.61876 0.0123972
## 3 0.054369      2   0.55974 0.55942 0.0119839
## 4 0.015917      3   0.50537 0.51188 0.0106445
## 5 0.010483      4   0.48945 0.49674 0.0100155
## 6 0.010000      5   0.47897 0.48631 0.0098743
```

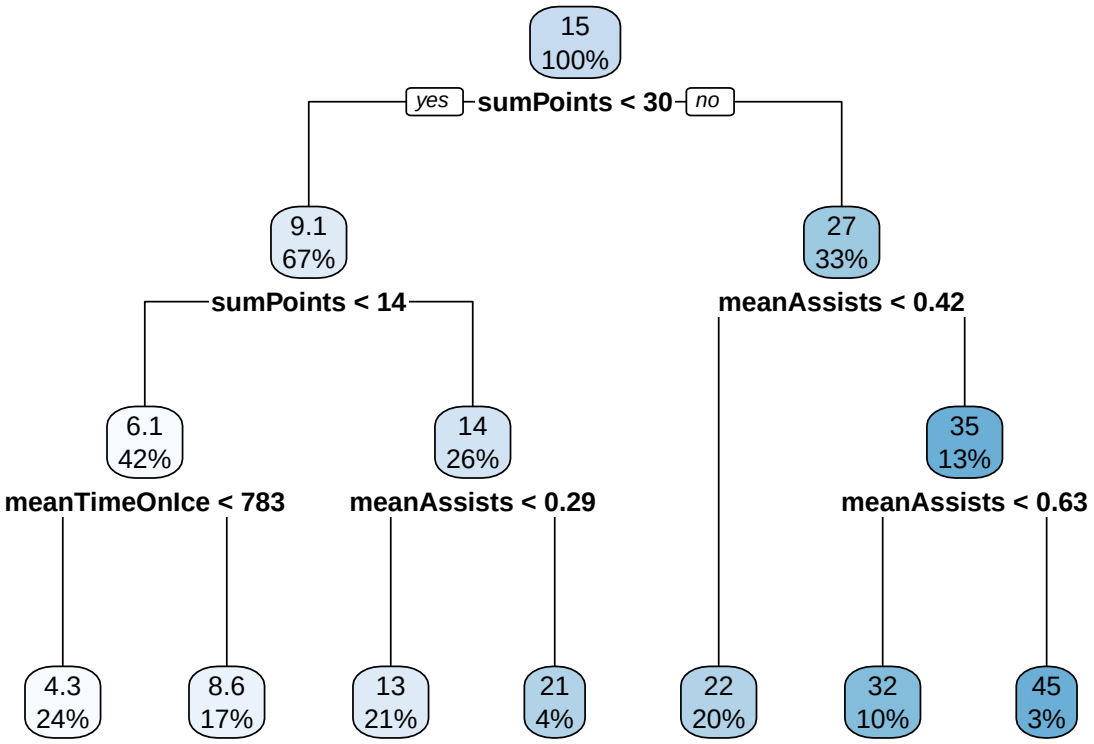




rt2: Regression Tree Model: All Predictors in Training Set

Whereas the first regression tree model was minimalist in variable selection, the second regression tree emphasized using all available variables.

```
# Train RT model 2
fit_rt2 <- rpart(nextSumAssists~. , method="anova", data=train_set)

# Plot decision rules.
rpart.plot(fit_rt2)
```




```
# Rank variable importance.
varImp(fit_rt2) %>% arrange(desc(Overall))
```

```
##                 Overall
## meanAssists   1.00416684
## sumPoints     0.95813799
## sumAssists    0.91697619
## sumShots      0.48769629
## meanTimeOnIce 0.45001484
## sumGoals      0.31193419
## meanShots     0.19900638
## meanGoals     0.06667976
## season        0.00000000
## player_id     0.00000000
## sumTimeOnIce  0.00000000
## next_season   0.00000000
```

This model results in:

1. Seven possible predictions ranging from 4.3 to 45 are generated. This is similar to the first model's range of 4.5 to 43, but with one more possible prediction.
2. The mean of assists becomes the most important predictor (dethroning sum of assists), followed by sum of points and sum of assists. Between the first and second models, current season assists, either in mean or sum form are the most important predictor for next season assists.

3. The sum of shots appears for the first time as a predictor, followed by mean time on ice.

It appears including all our quantitative summaries as predictors produces a more specific regression tree but drops predictive power somewhat. The RMSE is higher at 9.963 in this model, versus 9.819 in the first model.

This means manually selecting which predictors to include influenced model effectiveness (versus providing rpart with all variables and letting it pick). We included all of the variables from the first model, plus more, yet the model generated a tree with inferior predictive power (as measured by RMSE). The predictions could be more accurate for some groups than others.

```
# Predict using testing set.
predictions <- predict(fit_rt2, newdata=test_set)

# Calculate and store RMSE.
(rmse_rt2 <- RMSE(test_set$nextSumAssists, predictions))
```

```
## [1] 9.963498
```

```
# List possible predictions.
predictions %>% unique()
```

```
## [1] 22.486179  4.303057  8.619257 12.675050 32.035789 20.788918 45.258333
```

```
# Compute distribution of possible player season assists.
train_set$nextSumAssists %>% summary()
```

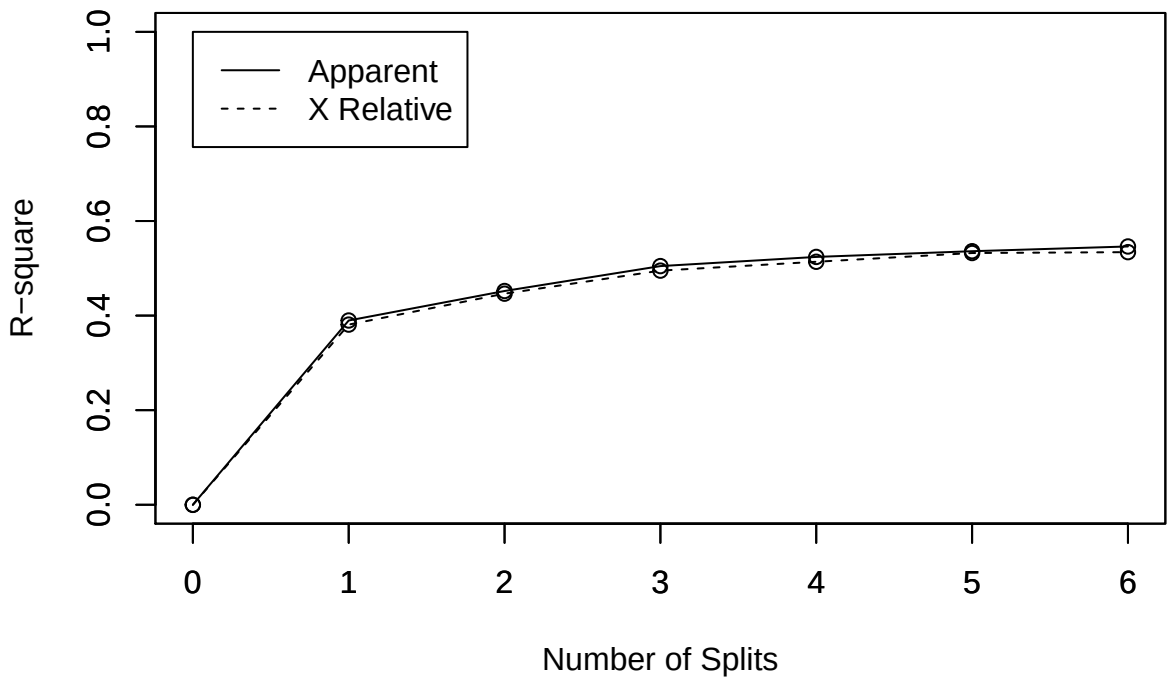
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    4.00   12.00   15.06   22.00   96.00
```

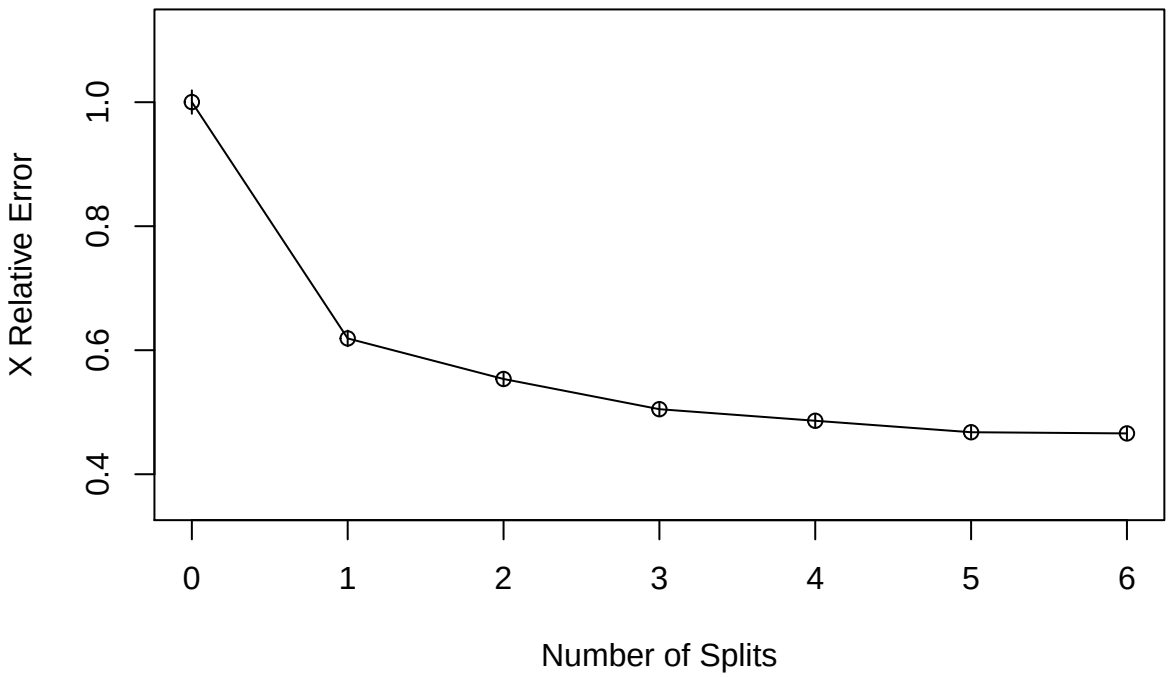
```
# Print complexity parameter analysis.
printcp(fit_rt2)
```

```
##
## Regression tree:
## rpart(formula = nextSumAssists ~ ., data = train_set, method = "anova")
##
## Variables actually used in tree construction:
## [1] meanAssists   meanTimeOnIce sumPoints
##
## Root node error: 1730500/9248 = 187.12
##
## n= 9248
##
##         CP nsplit rel error  xerror      xstd
## 1 0.389630      0   1.00000 1.00024 0.0188883
## 2 0.062387      1   0.61037 0.61902 0.0126711
## 3 0.052681      2   0.54798 0.55363 0.0111131
## 4 0.019357      3   0.49530 0.50489 0.0105823
## 5 0.012110      4   0.47594 0.48620 0.0097925
## 6 0.010039      5   0.46383 0.46771 0.0094060
## 7 0.010000      6   0.45380 0.46583 0.0094013
```

```r
# Plot R-sqare.
rsq.rpart(fit_rt2)
```

```
##
## Regression tree:
## rpart(formula = nextSumAssists ~ ., data = train_set, method = "anova")
##
## Variables actually used in tree construction:
## [1] meanAssists   meanTimeOnIce sumPoints
##
## Root node error: 1730500/9248 = 187.12
##
## n= 9248
##
##         CP nsplit rel error  xerror      xstd
## 1 0.389630      0   1.00000 1.00024 0.0188883
## 2 0.062387      1   0.61037 0.61902 0.0126711
## 3 0.052681      2   0.54798 0.55363 0.0111131
## 4 0.019357      3   0.49530 0.50489 0.0105823
## 5 0.012110      4   0.47594 0.48620 0.0097925
## 6 0.010039      5   0.46383 0.46771 0.0094060
## 7 0.010000      6   0.45380 0.46583 0.0094013
```





A closer look at this regression tree shows:

1. 6 splits (7 terminal nodes) optimized the complexity parameter and produced an R-square just under 0.5.
2. Possible future experiments could include iteratively removing individual predictors to trying for a better model.

lm1: Linear Model: Season Goals, Season Assists

A linear model using total goals and total assists as predictors returns:

1. An RMSE of 9.3793, better than both regression trees, using one fewer predictors.
2. The interquartile residuals range between -5.149 and 4.430. These are relatively small deviations from actuals.
3. However, the minimum and maximum residuals are many times those in the interquartile range, errors more than half the assists produced by the best players in a season.
4. The multiple R-squared of 0.5472 is fairly explanatory of the variation in next season player assist totals, but not especially high.
5. Both coefficients are important to the linear model, by their low p-values.

```r
# Train LM model 1
fit_lm1 <- lm(nextSumAssists ~ sumGoals + sumAssists, data=train_set)

# Use the model to predict.
pred <- predict(fit_lm1, newdata=test_set)

# Compute RMSE.
(rmse_lm1 <- RMSE(test_set$nextSumAssists, pred))
```

```
## [1] 9.379286
```

```r
# Summarize model.
summary(fit_lm1)
```

```
##
## Call:
## lm(formula = nextSumAssists ~ sumGoals + sumAssists, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.494  -5.149  -1.802   4.430  56.448
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  3.91401    0.14243   27.48 <0.0000000000000002 ***
## sumGoals     0.20740    0.01584   13.10 <0.0000000000000002 ***
## sumAssists   0.61736    0.01095   56.36 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.207 on 9245 degrees of freedom
## Multiple R-squared:  0.5472, Adjusted R-squared:  0.5471
## F-statistic:  5586 on 2 and 9245 DF,  p-value: < 0.00000000000000022
```

lm1a: Linear Model: All Quantitative Predictors

A second linear model was built, using all available quantitative predictors. The hope is more predictors will improve this model over the last. The player_id field is omitted because it crashes the model with over 3300 coefficients.

1. Mean goals and mean assists have the highest absolute coefficients, but their impact on the prediction for a player depends on that player's mean goals and assists for the current season of a data row.
2. The low total model p-value (probability that the differences are by chance) indicates this model produces statistically significant results. The p-values for all included predictors except sum shots is below 0.001.
3. The p-value for sum shots is below 0.01, so it is relevant, but its coefficient of -0.00921005 does not weigh meaningfully into estimates except at the extremes, where it can make a difference. The most elite playmakers can earn 90+ assists in a season, so this coefficient can make the difference of up to 1 assist in the prediction. Since 90 * -0.00921005 = -0.83, this coefficient can round the estimate down by a maximum of 1 unit. The fact of its relevance makes it worthy of inclusion for adjusting high end player estimates.

4. The multiple R-squared suggests more than half the variance in next season player total assists is explained by this model.

5. The RMSE of this model is 9.1898, an improvement over the last model's 9.3793. This seems to come mostly from replacing sum goals with mean goals, and further adding effects from mean assists, mean shots, and sum points, as compared to the first model.

```
# Tain a model dropping the player_id predictor.
fit_lm1a <- lm(nextSumAssists ~ . -player_id, data=train_set)
predictions_lm1a <- predict(fit_lm1a, newdata=test_set)

# Compute new RMSE.  Not much different.
(rmse_lm1a <- RMSE(test_set$nextSumAssists, predictions_lm1a))
```

```
## [1] 9.189817
```

```
# Show model details.
fit_lm1a
```

```
##
## Call:
## lm(formula = nextSumAssists ~ . - player_id, data = train_set)
##
## Coefficients:
##   (Intercept)         season      sumPoints     sumAssists       sumGoals
##  -42.39880725    -0.00029865     0.09498260     0.38834349             NA
##      sumShots   sumTimeOnIce    meanAssists      meanGoals      meanShots
##   -0.00921005    -0.00004359     7.11912976     7.82127338     1.99393331
## meanTimeOnIce    next_season
##    0.00861859     0.00030042
```

```
# Show linear model details
summary(fit_lm1a)
```

```
##
## Call:
## lm(formula = nextSumAssists ~ . - player_id, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.772  -5.176  -0.884   4.269  53.581
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate    Std. Error t value             Pr(>|t|)
## (Intercept)   -42.398807247  46.160644631  -0.919              0.35838
## season         -0.000298649   0.000036183  -8.254 < 0.0000000000000002 ***
## sumPoints       0.094982598   0.034798524   2.730              0.00636 **
## sumAssists      0.388343486   0.045362481   8.561 < 0.0000000000000002 ***
## sumGoals                 NA            NA      NA                   NA
## sumShots       -0.009210053   0.006279771  -1.467              0.14251
## sumTimeOnIce   -0.000043588   0.000009494  -4.591 0.000004462639665324 ***
## meanAssists     7.119129757   1.315145043   5.413 0.000000063463403041 ***
```

```
## meanGoals       7.821273378   1.795257777   4.357 0.000013350719215426 ***
## meanShots       1.993933315   0.352545910   5.656 0.000000015972933496 ***
## meanTimeOnIce   0.008618585   0.000732599  11.764 < 0.0000000000000002 ***
## next_season     0.000300423   0.000036690   8.188 0.000000000000000301 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.789 on 9237 degrees of freedom
## Multiple R-squared:  0.5877, Adjusted R-squared:  0.5872
## F-statistic:  1317 on 10 and 9237 DF,  p-value: < 0.00000000000000022
```

lm1_rt3: Regression Tree with Linear Prediction Input

Perhaps the linear predictions can be improved by inputting them as a predictor into a regression tree. This is attempted next, using all predictors in the exhaustive regression tree above (using all features plus the linear prediction):

1. Predictions from the the best linear model are added as a feature to the training set and the test set.
2. A regression tree is run against all of the variables, including the linear prediction.
3. Predictions are made with this new ensemble model, and the RMSE is computed.

```
# Run predictions using linear model with exhaustive variables.
pred_lm <- predict(fit_lm1a, newdata = train_set)

# Add pred_lm to train set.
train_set_lm <- cbind(train_set, pred_lm)

# Rename added field.
train_set_lm <- train_set_lm %>%
  rename(lm_nextSumAssists="...14")

# See sample rows.
train_set_lm %>% head(3)
```

```
## # A tibble: 3 x 14
## # Groups:   season [1]
##   season player_id sumPoints sumAssists sumGoals sumShots sumTimeOnIce
##    <dbl>     <int>     <int>      <int>    <int>    <int>        <int>
## 1 2.00e7   8444919        20         19        1       69        96465
## 2 2.00e7   8445000        33         13       20      119        53248
## 3 2.00e7   8445176        79         45       34      225        90473
## # ... with 7 more variables: meanAssists <dbl>, meanGoals <dbl>,
## #   meanShots <dbl>, meanTimeOnIce <dbl>, next_season <int>,
## #   nextSumAssists <int>, lm_nextSumAssists <dbl>
```

```
# Add pred_lm to test set.
pred <- predict(fit_lm1a, newdata=test_set)
test_set_lm <- cbind(test_set, pred)

# Rename added field.
test_set_lm <- test_set_lm %>%
```

```
  rename(lm_nextSumAssists="...14")

# See relevant rows in sample rows.
train_set_lm %>%
  select(player_id, sumAssists, nextSumAssists, lm_nextSumAssists) %>%
  head(3)
```

```
## # A tibble: 3 x 5
## # Groups:   season [1]
##     season player_id sumAssists nextSumAssists lm_nextSumAssists
##      <dbl>     <int>      <int>          <int>             <dbl>
## 1 20002001   8444919         19              3              15.0
## 2 20002001   8445000         13             17              13.6
## 3 20002001   8445176         45             13              38.9
```

```
# Build regression tree model using linear predictions as a feature.
fit_lm1_rt3 <- rpart(nextSumAssists~., method="anova", data = train_set_lm)

# Make predictions using this new model.
predictions_ensemble <- predict(fit_lm1_rt3, newdata = test_set_lm)

# List possible predictions.
predictions_ensemble %>% unique
```

```
## [1] 23.442993  5.859267 13.758036 33.407362 45.347280
```

```
# Test and score.
(rmse_lm1_rt3 <- RMSE(test_set_lm$nextSumAssists, predictions_ensemble))
```

```
## [1] 9.70372
```

```
# Plot the tree.
varImp(fit_lm1_rt3) %>% arrange(desc(Overall))
```

```
##                      Overall
## lm_nextSumAssists 0.95444378
## sumAssists        0.80051088
## meanAssists       0.79606153
## sumPoints         0.76665263
## sumShots          0.48900766
## meanTimeOnIce     0.07882063
## season            0.00000000
## player_id         0.00000000
## sumGoals          0.00000000
## sumTimeOnIce      0.00000000
## meanGoals         0.00000000
## meanShots         0.00000000
## next_season       0.00000000
```

Results:

1. The RMSE is higher than using either liner model alone.
2. Since this regression tree results in 5 possible predictions, it appears the linear prediction's RMSE has been impaired by forcing continuous scale predictions into one of the five possible predictions output by the regression tree. It might be more useful to try using regression tree predictions in a linear model instead. This is done in the next model.
3. The linear prediction was the most important variable in the regression tree.

rt1_lm3: Linear Model with Regression Prediction Input

For convenience, we use the predictions generated from the first regression tree (with the best RMSE of all regression trees thus far) as an input to a new linear model:

1. Predictions from the regression tree model are added as a feature to the training set and the test set.
2. A linear model is run against all of the variables, including the regression tree prediction.
3. Predictions are made with this new ensemble model, and the RMSE is computed.

```
# First, run prediction with RT1 against train set.
pred_rt1 <- predict(fit_rt1, newdata = train_set)

# Add predictions to train set.
train_set_rt1 <- cbind(train_set, pred_rt1)

# Rename added field.
train_set_rt1 <- train_set_rt1 %>%
  rename(rt1_nextSumAssists="...14")

# See sample rows.
train_set_rt1 %>% head(3)
```

```
## # A tibble: 3 x 14
## # Groups:   season [1]
##   season player_id sumPoints sumAssists sumGoals sumShots sumTimeOnIce
##    <dbl>     <int>     <int>      <int>    <int>    <int>        <int>
## 1 2.00e7   8444919        20         19        1       69        96465
## 2 2.00e7   8445000        33         13       20      119        53248
## 3 2.00e7   8445176        79         45       34      225        90473
## # ... with 7 more variables: meanAssists <dbl>, meanGoals <dbl>,
## #   meanShots <dbl>, meanTimeOnIce <dbl>, next_season <int>,
## #   nextSumAssists <int>, rt1_nextSumAssists <dbl>
```

```
# Add rt1 predictions to test set.
pred_rt1 <- predict(fit_rt1, newdata=test_set)
test_set_rt1 <- cbind(test_set, pred_rt1)

# Rename added field.
test_set_rt1 <- test_set_rt1 %>%
  rename(rt1_nextSumAssists="...14")
# See relevant fields in sample rows.
test_set_rt1 %>%
  select(player_id, sumAssists, nextSumAssists, rt1_nextSumAssists) %>%
  head(3)
```

```
## # A tibble: 3 x 5
## # Groups:   season [1]
##     season player_id sumAssists nextSumAssists rt1_nextSumAssists
##      <dbl>     <int>      <int>          <int>              <dbl>
## 1 20142015   8448208         30             41               23.1
## 2 20142015   8460542         21              6               23.1
## 3 20142015   8460720          0              0                4.53
```

```
# Build linear model using rt1 prediction as input.
fit_rt1_lm3 <- lm(nextSumAssists~., data = train_set_rt1)

# See new model.
summary(fit_rt1_lm3)
```

```
##
## Call:
## lm(formula = nextSumAssists ~ ., data = train_set_rt1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.135  -5.089  -0.788   4.275  52.536
##
## Coefficients: (1 not defined because of singularities)
##                          Estimate      Std. Error t value             Pr(>|t|)
## (Intercept)        -1271.394131070   132.256708765  -9.613 < 0.0000000000000002
## season                -0.000311136     0.000036025  -8.637 < 0.0000000000000002
## player_id              0.000190996     0.000019281   9.906 < 0.0000000000000002
## sumPoints              0.102822714     0.034661887   2.966              0.003020
## sumAssists             0.367034560     0.050918458   7.208  0.00000000000061112
## sumGoals                        NA              NA      NA                   NA
## sumShots              -0.010932273     0.006252493  -1.748              0.080417
## sumTimeOnIce          -0.000032394     0.000009594  -3.376              0.000737
## meanAssists            7.746347290     1.313443964   5.898  0.00000000381466938
## meanGoals              7.799895549     1.786156820   4.367  0.00001274200239082
## meanShots              1.979189805     0.350812799   5.642  0.00000001733178817
## meanTimeOnIce          0.008430113     0.000758352  11.116 < 0.0000000000000002
## next_season            0.000293550     0.000036512   8.040  0.00000000000000101
## rt1_nextSumAssists     0.013607029     0.036800583   0.370              0.711577
##
## (Intercept)        ***
## season             ***
## player_id          ***
## sumPoints          **
## sumAssists         ***
## sumGoals
## sumShots           .
## sumTimeOnIce       ***
## meanAssists        ***
## meanGoals          ***
## meanShots          ***
## meanTimeOnIce      ***
## next_season        ***
## rt1_nextSumAssists
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.744 on 9235 degrees of freedom
## Multiple R-squared:  0.592,  Adjusted R-squared:  0.5915
## F-statistic:  1117 on 12 and 9235 DF,  p-value: < 0.00000000000000022
```

```
# Make predictions using this new model.
predictions_ensemble2 <- predict(fit_rt1_lm3, newdata = test_set_rt1)

# Test and score.
(rmse_rt1_lm3 <- RMSE(test_set_rt1$nextSumAssists, predictions_ensemble2))
```

```
## [1] 9.116968
```

Results:

1. Although the p-value of the regression tree prediction indicates this variable is not important overall, it appears to have made a difference in the overall RMSE of 9.117, beating out the exhaustive linear model alone at 9.190. We have a new, winning RMSE.
2. Examining the 0.013607029 coefficient for rt1_nextSumAssists: Since 0.5 rounds to 1, if the final estimates are rounded to the nearest unit, this new coefficient will begin to impact new estimates where the regression tree predicted 36.75 or higher ($0.5/0.013607029 = 36.75$). Since the next highest regression tree prediction was 43.05, when looking at prediction counts, this impacts 64 estimates made by the regression tree alone, on the test set.
3. When the regression tree estimate alone is below 31.85 (the next lowest prediction of the 6 possible regression tree outputs), the new coefficient rt_nextSumAssists would have to be supported by other coefficients in the same positive direction to force an estimate up one full assist unit. With the 0.013607029 coefficient, the ensemble model estimate can only move a prediction of 31.85 by $0.013607029 * 31.85 = 0.433$ by itself. This rounds to 0 assists without the help of another (positively correlated) coefficient in the model.
4. Given the above, it is likely prediction errors for higher end players (those estimated 43.05 by the regression tree model) were reduced using this ensemble versus regression tree or linear model alone, in order to produce the lower overall RMSE, as compared to the regression tree alone.

```
# Table the counts of predictions by RT1 over 36.75.
#  There are 64 estimates this will impact.
table(test_set_rt1$rt1_nextSumAssists)
```

```
##
## 4.52847282139776 8.82425307557118 14.6116751269036  23.113337149399
##              230              250              420              387
## 31.8483353884094 43.0498338870432
##              210               64
```

The 9.117 RMSE of this model makes it the new leader.

rf1: Random Forest Model

The relative success of individual regression tree models above inspires trying a random forest model.

1. Random forests will select a subset of predictors for each tree, with a default of 500 trees in ours.
2. Perhaps the random choice of dimensions per tree can identify a better predictive model than the active choices for the regression trees.
3. Random forests can accept quantitative and categorical predictors.

Design of this experiment followed this reasoning:

1. Several attempts to use the full training set in random forests failed on hardware limits. Random sampling from the training set substitutes for using the entire training set, in order to relieve workloads.
2. The model is provided with player_id as the most direct indicator of player-specific statistics.

```
# Sample 10 percent of the training set.
#  Create a sample index randomly.
set.seed(1)
sample_index <- createDataPartition(y = train_set$nextSumAssists, times = 1, p = 0.25, list = FALSE)
#  Use the index to sample from training.
sample2 <- train_set[sample_index,]
```

```
## Warning: The 'i' argument of ''['()' can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_warnings()' to see where this warning was generated.
```

```
# Run a random forest model.
#  Runs in roughly 10 minutes on an 8-core CPU with 16G of RAM.
fit_rf1 <- randomForest(nextSumAssists ~ sumAssists + sumPoints + sumShots +
                        meanAssists + meanShots +
                        sumTimeOnIce + meanTimeOnIce, data=sample2
                        , na.action=na.fail)

# Show random forest summary.
fit_rf1
```

```
##
## Call:
##  randomForest(formula = nextSumAssists ~ sumAssists + sumPoints +      sumShots + meanAssists + meanS
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 82.05126
##                     % Var explained: 56.33
```

```
# Show important variables.
varImp(fit_rf1) %>%
  arrange(desc(Overall))
```

```
##                Overall
## sumPoints     96203.36
## sumAssists    76878.09
## meanAssists   71444.30
## sumShots      52898.49
## meanShots     45853.97
## meanTimeOnIce 37347.03
## sumTimeOnIce  33115.55
```

The random forest model did the following:

1. Built 500 individual regression trees using 2 randomly selected features for each tree. It computed predictions from each of these trees and used the default consensus method for final predictions.
2. It used 2 variabled per split, different from the regression trees models above.
3. The R-square (% of variance explained) is significantly over 50% indicating some predictive power competitive with previous models.
4. The most important variable was sum points, in contrast to previous models.

```
# Use model to predict.
predictions <- predict(fit_rf1, newdata=test_set)

# Calculate and save RMSE.
(rmse_rf1 <- RMSE(test_set$nextSumAssists, predictions))
```

```
## [1] 9.469249
```

The random forest returns a competitive RMSE, but not the best one:

1. A large sample, 25 percent of the training set, was used to train the model.
2. Perhaps larger samples or more variables can improve the RMSE.
3. Future work may include tuning the number of trees, modifying the complexity parameter to be more stringent, or setting minimums for observations in terminal nodes. Including predictions from other models as inputs could also produce interesting results.

Conclusions & Evaluation of Final Model

Final Model

The most promising model by RMSE ranking is the *Linear Model with a Regression Tree Prediction Feature*. Its development benefited from insights developed in data exploration and running other models:

1. The regression tree step used meanTimeOnIce, sumShots, sumAssists.
2. The regression tree predictions used as a feature in a linear model that used all aggregated predictors. Including all the variables in this step was inspired by the stronger results gained from the regression tree that used more variables (run before this ensemble).

Model RMSE's are summarized and ranked from best to worst. Although all the models beat the target RMSE of 10.000, the winning model stands out.

```
# Put RMSE scores into a vector.
rmse_score <- c(rmse_rt1, rmse_rt2, rmse_lm1, rmse_lm1a
                , rmse_lm1_rt3, rmse_rf1, rmse_rt1_lm3)

# Put RMSE descriptions in a vector.
rmse_description <- c ("Regression Tree 1"
                       ,"Regression Tree 2"
                       ,"Linear Model 1"
                       ,"Linear Model 2"
                       ,"Regression Tree 3, Linear Feature"
                       ,"Random Forest Model"
                       ,"Linear Model, Regression Tree Feature"
                       )

# Display result as a table, sorted by ascending RMSE.
#  Best result on top.
rmse_table <- cbind(rmse_description, round(rmse_score,4)) %>%
  data.frame() %>% arrange(rmse_score)

# Change column titles.
rmse_table <- rmse_table %>%
  rename(RMSE= V2, Model = rmse_description)

# Display results in a formatted table.
rmse_table %>% kable()
```

Model	RMSE
Linear Model, Regression Tree Feature	9.117
Linear Model 2	9.1898
Linear Model 1	9.3793
Random Forest Model	9.4692
Regression Tree 3, Linear Feature	9.7037
Regression Tree 1	9.819
Regression Tree 2	9.9635

Final Model Validation

The linear model with regression tree inputs returned the strongest RMSE of 9.117, so it is validated against the validation set below. Because this is an ensemble model, it must be processed through several steps to validate:

1. First, the regression tree model is used to produce predictions using the validation set.
2. Next, the regression tree predictions are combined into the validation set as an additional feature.
3. Finally, the linear model is run against the validation set.

```
# First, run prediction with RT1 against validation set.
pred_v <- predict(fit_rt1, newdata = validation)
```

```r
# Add predictions to validation set.
validation_set <- cbind(validation, pred_v)

validation_set
```

```
## # A tibble: 702 x 14
## # Groups:   season [2]
##    season player_id sumPoints sumAssists sumGoals sumShots sumTimeOnIce
##     <dbl>     <int>     <int>      <int>    <int>    <int>        <int>
##  1 2.02e7   8464989        24         12       12      108        59760
##  2 2.02e7   8465009        27         19        8      170       117303
##  3 2.02e7   8466138        36         23       13       75        51733
##  4 2.02e7   8466139        52         21       31      217        90498
##  5 2.02e7   8468001        14         11        3       86        50886
##  6 2.02e7   8468493        24         20        4       77       113956
##  7 2.02e7   8468498        15         14        1       71       118328
##  8 2.02e7   8468508        51         35       16      198        81613
##  9 2.02e7   8468509        27         23        4       76        87794
## 10 2.02e7   8468611        13          8        5       58        36573
## # ... with 692 more rows, and 7 more variables: meanAssists <dbl>,
## #   meanGoals <dbl>, meanShots <dbl>, meanTimeOnIce <dbl>, next_season <int>,
## #   nextSumAssists <int>, ...14 <dbl>
```

```r
# Rename added field to suit the next step of the final model.
#  Since we used this variable name in the original conception
#  of the linear step, we can't vary from it.
validation_set <- validation_set %>%
  rename(rt1_nextSumAssists="...14")

# See relevant fields in sample rows.
validation_set %>% select(player_id, sumAssists
                , nextSumAssists, rt1_nextSumAssists) %>%
                  head(3)
```

```
## # A tibble: 3 x 5
## # Groups:   season [1]
##     season player_id sumAssists nextSumAssists rt1_nextSumAssists
##      <dbl>     <int>      <int>          <int>              <dbl>
## 1 20172018   8464989         12             13               14.6
## 2 20172018   8465009         19             13               14.6
## 3 20172018   8466138         23             41               23.1
```

```r
# Use the final linear model to make final predictions.
predictions_validation <- predict(fit_rt1_lm3, newdata = validation_set)

# Test and score.
(rmse_validation <- RMSE(validation_set$nextSumAssists, predictions_validation))
```

```
## [1] 9.19705
```

The winning model is the only model run against the validation data set. Among those methods tested, an *exhaustive linear model using a regression tree prediction as an additional feature* demonstrated the best

predictive ability for *Next Season Player Assists*. Moreover, the results appear to have struck a balance between predictive power and overtraining, as validation turned up a comparable RMSE compared to test.

However, this barely scratches the surface of other available models and more sophisticated ensemble techniques. There is plenty of work to do from here forward, discussed below.

Potential Impact

The final model in this report can be used for private hockey pools where friends compete by picking players or in the riskier proposition of making hockey bets in jurisdictions where allowed. It could also contribute to overall player valuations for trades, but not without extensive expansion.

The world of hockey analytics is wide and deep. The final model in this report suggests seasonal player statistics are indeed predictable through modeling, and that results can be achieved with fairly accessible modeling techniques in a field that was deeply arcane and inaccessible to fans only a decade ago.

Model Performance

Summary

The final model is useful for predicting player season assists overall, as represented by the low relative RMSE. On the scale of season player assists, which ranges from 0 to 90+, the final RMSE is low. It may be counter-productive to aim any lower, as natural variation in player performance, slump seasons, injuries and exogenous events demand some flexibility.

However, if the level of error must be accepted, perhaps the distribution of errors can be targeted for improvement. An overall low RMSE does not mean the model's predictive power for individual players is equal, nor as good as it can be. The RMSE is an average measure for the whole set from which predictions are computed.

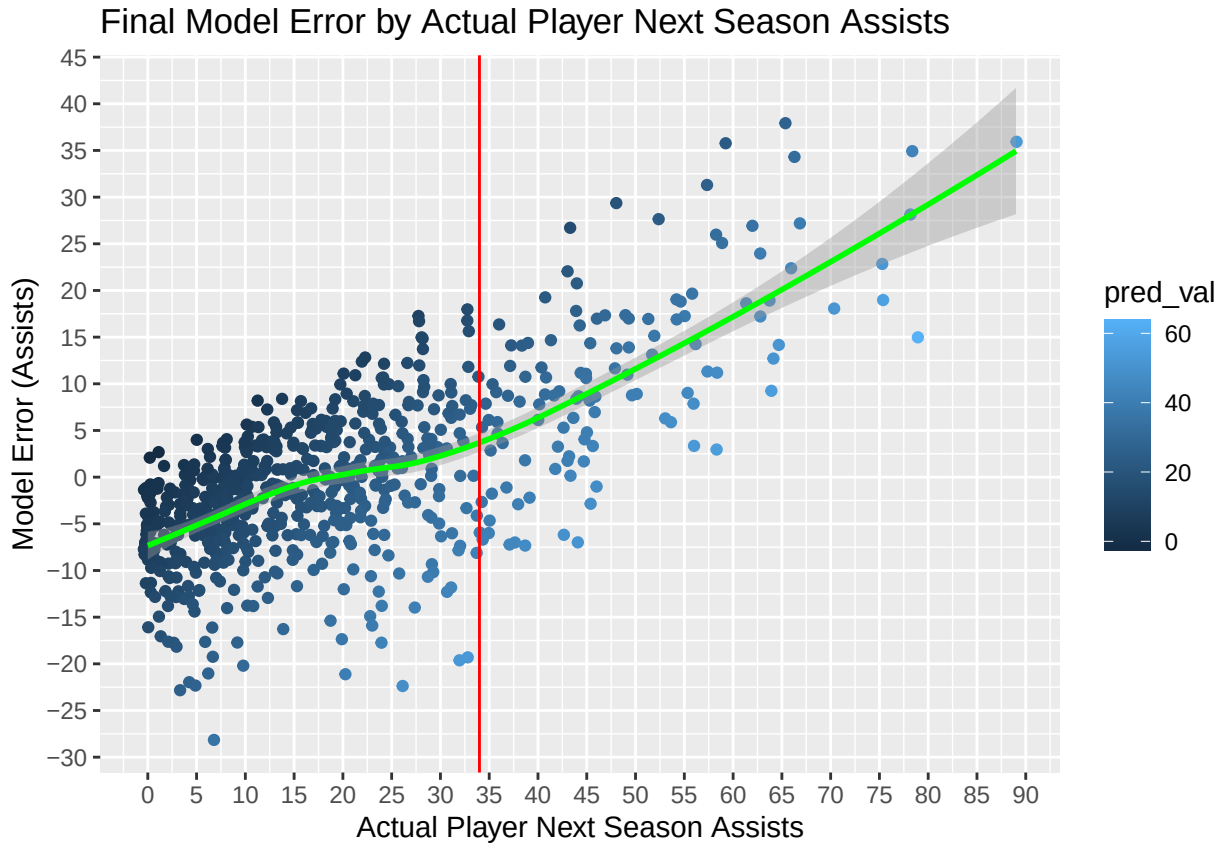
For example, It may be acceptable to trade off better predictive power for the vast majority of average players, in favor of better predictions for high performers and elite players. This can be a guidepost for future work.

Analysis of Errors

An analysis of the errors in our final model follows, graphed against actual player season assists.

```
# Put validation predictions and actuals side-by-side in a table.
validatedPredictions <- cbind(validation, predictions_validation)

# Rename predictions field
validatedPredictions <- validatedPredictions %>%
  rename(pred_val="...14")

# Create an object with new fields for (rounded)
#  predictions and errors.
predictionError <- validatedPredictions %>%
  unique %>%
  mutate(pred=round(pred_val,0)
         ,err=round(nextSumAssists,0)-round(pred_val,0)) %>%
          arrange(abs(err))
```

```
# Graph the errors in a scatter plot.
#  Show predictions in color.
#  Draw a green smooth mean line.
#  Adjust scale markers with sequences.
#  Add labels.
predictionError %>% ggplot(aes(x=nextSumAssists, y=err, colour=pred_val)) +
  geom_jitter() +
  geom_smooth(color="green") +
  scale_y_continuous(breaks = seq(-45,45,5)) +
  scale_x_continuous(breaks = seq(0,100,5)) +
  geom_vline(xintercept = 34, color="red") +
  labs(y="Model Error (Assists)"
      ,x="Actual Player Next Season Assists"
      ,title="Final Model Error by Actual Player Next Season Assists")
```



1. Some player seasons with actuals below 34 (left of the red line) suffer large negative errors from this model. These are where predictions of next season assists are higher than actual next season assists, relative to other estimates by this model. Further investigation may uncover methods to reduce errors of *overestimating* in this zone.

2. The highest severity (not necessarily highest count) errors are among players with very high "next season" assist counts. Here, the problem is *underestimating* the target. In hockey, these players generate perhaps the most volatile assist records. Assist production endemic to their individual play is already volatile, but the effect of missed games, shorter seasons beyond their control, and injuries is more pronounced for these players if assist production is consistent. Their mean assists may not suffer

if they are consistent per game, but their season totals will. A possible solution might imply splitting the data for use in different models.

3. If the errors represent short seasons or injuries, perhaps applying regularization to the linear part of the final model, to reduce the large impact of a few observations, could reduce these errors in general. Another approach would be to arbitrarily reduce the number of outliers allowed in the training data, reducing their effect on model fit in training. This ought to be done carefully because assumed outliers may in fact just be infrequent observations.

4. In general, the mean error is low and the standard deviation of errors is not extreme for the statistic being predicted. There is room for improvement, but prediction may be close to a boundary where targeting a lower RMSE would mean overfitting and less flexibility for future seasons.

```
# Compute mean of errors.
mean(predictionError$err)
```

```
## [1] 0.1196581
```

```
# Compute standard deviation of errors.
sd(predictionError$err)
```

```
## [1] 9.195138
```

Future Work

Some examples of future work could include:

1. Deepening the use of random forests by calibrating them to use more trees and more predictors per tree. This would require more computing power, but could yield stronger results. In this study, the random forest we tried produced a competitive RMSE in the 9.47 range using a 25% sample of the training set. Perhaps a larger sample or enough computation power to use the entire training set, or better parameter calibration can yield a model to beat 9.117.

2. Trying other combinations of inputting the predictions from one model into another could yield stronger results. This would probably benefit from a deeper analysis of the error terms and distribution.

3. An emphasis on targeting error distributions, or even sectioning off the data with quantitative boundaries and applying different models to each range may be an option.

Reference

NHL Game Data Used in this Report

https://www.kaggle.com/martinellis/nhl-game-data

NHL Lockout Seasons

https://thehockeywriters.com/nhl-2004-05-lockout-impacts-2020/ https://thehockeywriters.com/2012-nhl-lockout/

League Expansion 2017: Vegas Golden Knights:

https://www.nhl.com/news/nhl-expansion-history/c-281005106

Anze Kopitar: Slovenian Playmaker Extraordinaire:

https://www.nhl.com/player/anze-kopitar-8471685

Connor McDavid: Generational NHL Player:

https://www.nhl.com/player/connor-mcdavid-8478402

Sam Gagner: 8 Point Game

https://www.nhl.com/news/gagner-joins-elite-company-with-8-point-night/c-615027

Jaromir Jagr: Elite Performaces at Age 42 and 43

https://www.nhl.com/player/jaromir-jagr-8448208

Vegas Golden Knights to Stanley Cup final in First Season

https://www.nhl.com/news/vegas-golden-knights-advance-to-stanley-cup-final/c-298711702