Lock Line Analysis

Eric Fastner

November 4, 2018

During the 2017-2018 season, I came across a set of articles by Tyler Dellow that really piqued my interest. It involved analyzing game-by-game results for all teams from the 05-06 season onwards to identify the minimum number of points that any playoff team had achieved during that time frame, as well as the maximum number of points that any non-playoff team had achieved during the same time frame. The result was 2 lines, the lock line and the point of no return.

As I've found myself wondering about the implications of this lock line and point of no return compared to various teams from each season, I thought it might be interesting to run a similar analysis myself. I've laid out my initial methodology below.

Methodology

The goal of this project is to create 2 lines:

Lock Line: The minimum number of points that any team has achieved and still made the playoffs after each game (1-82) of the season
Point of no Return: The maximum number of points that any team has achieved and still missed the playoffs after each game (1-82) of the season

I plan to use raw PBP data to create each of these lines, which means that I will need to take the following steps to create each line:

1. Summarize the raw PBP results into game results
2. Summarize results by team, identify the number of points earned by each team based on the results as well as add a value to denote what game of the season it is for each team (1 to 82)
3. Repeat for all seasons with available PBP data and combine into one file
4. Identify which teams made the playoffs and which teams missed the playoffs for each season
5. Group the data set by game number, find the minimum point total for all playoff teams and the maximum for all non-playoff teams

Step 1: Summarize the raw PBP results into game results

First, I'll read in the full 2017-2018 season's enhanced PBP file provided by Corsica

```
require(plyr)
require(dplyr)
require(readr)

raw_20172018 <- read_csv("~/Data_Sets/Hockey/Enhanced PbP/pbp_20172018.csv")

head(raw_20172018, n = 10)
```

```
## # A tibble: 10 x 57
##    game_id event_index season game_date  session game_period game_seconds
##      <int>       <int>  <int> <date>     <chr>         <int>        <int>
##  1  2.02e9           1 2.02e7 2017-10-04 R                 1            0
##  2  2.02e9           2 2.02e7 2017-10-04 R                 1            0
##  3  2.02e9           3 2.02e7 2017-10-04 R                 1            0
```

```
##  4  2.02e9           4 2.02e7 2017-10-04 R                 1            0
##  5  2.02e9           5 2.02e7 2017-10-04 R                 1            0
##  6  2.02e9           6 2.02e7 2017-10-04 R                 1           12
##  7  2.02e9           7 2.02e7 2017-10-04 R                 1           28
##  8  2.02e9           8 2.02e7 2017-10-04 R                 1           28
##  9  2.02e9           9 2.02e7 2017-10-04 R                 1           37
## 10  2.02e9          10 2.02e7 2017-10-04 R                 1           37
## # ... with 50 more variables: event_type <chr>, event_description <chr>,
## #   event_detail <chr>, event_team <chr>, event_player_1 <chr>,
## #   event_player_2 <chr>, event_player_3 <chr>, event_length <int>,
## #   coords_x <int>, coords_y <int>, players_substituted <chr>,
## #   home_on_1 <chr>, home_on_2 <chr>, home_on_3 <chr>, home_on_4 <chr>,
## #   home_on_5 <chr>, home_on_6 <chr>, away_on_1 <chr>, away_on_2 <chr>,
## #   away_on_3 <chr>, away_on_4 <chr>, away_on_5 <chr>, away_on_6 <chr>,
## #   home_goalie <chr>, away_goalie <chr>, home_team <chr>,
## #   away_team <chr>, home_skaters <int>, away_skaters <int>,
## #   home_score <int>, away_score <int>, game_score_state <chr>,
## #   game_strength_state <chr>, highlight_code <chr>, event_distance <dbl>,
## #   event_angle <dbl>, event_rinkside <chr>, event_circle <int>,
## #   event_zone <chr>, home_zone <chr>, prob_goal <dbl>, prob_save <dbl>,
## #   adj_home_corsi <dbl>, adj_home_fenwick <dbl>, adj_home_shot <dbl>,
## #   adj_home_goal <dbl>, adj_away_corsi <dbl>, adj_away_fenwick <dbl>,
## #   adj_away_shot <dbl>, adj_away_goal <dbl>
```

Obviously, all that I'm really interested from this file is the final score. Each game has an event_type value of "GEND" to denote the end of a game that should come in handy. Unfortunately, it's not as easy to come by as originally expected:

```
(31 * 82)/2
```

```
## [1] 1271
```

```
nrow(filter(raw_20172018, event_type == "GEND" & session == "R"))
```

```
## [1] 1269
```

Yikes. It appears that at best, we're missing 2 games when we just filter on each "GEND" event during the regular season. I think that instead of digging through and finding which games are missing a "GEND" event, it's probably better to just identify the end of a game manually. Here's my plan for each type of game ending:

1. **Regulation Ending:** event_type == "PEND", game_period == 3, and home_score != away_score
2. **OT Ending:** event_type == "GOAL", game_period == 4
3. **Shootout Ending:** event_type == "SOC", game_period == 5

The issue with some of the games in this PBP file is that this methodology isn't always 100% correct as the PBP record keeping is a little.....Messy. The most frequent issue that I found is that not all of the lines that meet the criteria above actually have the home_score and away_score values updated with the final score, so I needed a way to update the final scores based on the results. I came up with 3 functions to help me:

```
fun.OT_winners <- function(data_set, gameID) {
  #DESCRIPTION - finds the max home score and the max away score in an OT period. Sometimes necessary a
  #ARGUMENTS - data_set = a raw PBP file, gameID = a specified game_id

  home_score <- max(filter(data_set, game_id == as.integer(gameID))$home_score)
  away_score <- max(filter(data_set, game_id == as.integer(gameID))$away_score)

  return(c(home_score, away_score))
}
```

```
fun.shootout_winners <- function(data_set, gameID) {
  #DESCRIPTION - totals the number of goals by the home team and total by the away team to find the gam
  #ARGUMENTS - data_set = a raw PBP file, gameID = a specified game_id

  home_goals <- nrow(filter(data_set, game_id == as.integer(gameID) & event_type == "GOAL" & event_team
  away_goals <- nrow(filter(data_set, game_id == as.integer(gameID) & event_type == "GOAL" & event_team

  return(ifelse(home_goals > away_goals, "home", "away"))
}

fun.game_endings <- function(data_set) {
  #DESCRIPTION - Identifies the final ending of each game in a data set
  #ARGUMENTS - data_set = a raw PBP data frame, regular season only

  #Attempt to grab the final score of each game based on the period that it ends in
  game_endings <-
    data_set %>%
    filter((data_set$event_type == "PEND" & data_set$game_period == 3 & data_set$home_score != data_set$
            (data_set$event_type == "GOAL" & data_set$game_period == 4) |
            (data_set$event_type == "SOC" & data_set$game_period == 5 ))

  #Identifies if a game ended in OT or a shootout and calculates the final scores
  for (i in 1:nrow(game_endings)) {
    if (game_endings[i, "game_period"] == 5) {
      so_result <- fun.shootout_winners(data_set, game_endings[i, "game_id"])

      ifelse(so_result == "home",
             game_endings[i, "home_score"] <- game_endings[i, "home_score"] + 1,
             game_endings[i, "away_score"] <- game_endings[i, "away_score"] + 1)

    } else if (game_endings[i, "game_period"] == 4 & (game_endings[i, "home_score"] == game_endings[i,
      ot_result <- fun.OT_winners(data_set, game_endings[i, "game_id"])
      game_endings[i, "home_score"] <- ot_result[[1]]
      game_endings[i, "away_score"] <- ot_result[[2]]
    }
  }
  return(game_endings)
}
```

fun.game_endings filters my data set according to the criteria I laid out above, loops through each game individually, identifies how the game ended, and then manually calculates the score if needed using the fun.OT_winners or fun.shootout_winners functions as needed. The resulting data set looks like this:

```
results_20172018 <- fun.game_endings(raw_20172018)

head(results_20172018, n = 10)
```

```
## # A tibble: 10 x 57
##    game_id event_index season game_date  session game_period game_seconds
##      <int>       <int>  <int> <date>     <chr>         <int>        <int>
##  1  2.02e9         965 2.02e7 2017-10-04 R                 3         3600
##  2  2.02e9         958 2.02e7 2017-10-04 R                 4         3675
##  3  2.02e9         981 2.02e7 2017-10-04 R                 3         3600
##  4  2.02e9         993 2.02e7 2017-10-04 R                 3         3600
```

```
##  5  2.02e9        1001 2.02e7 2017-10-05 R                3         3600
##  6  2.02e9         947 2.02e7 2017-10-05 R                5         4800
##  7  2.02e9         928 2.02e7 2017-10-05 R                3         3600
##  8  2.02e9        1031 2.02e7 2017-10-05 R                5         4800
##  9  2.02e9         876 2.02e7 2017-10-05 R                3         3600
## 10  2.02e9        1012 2.02e7 2017-10-05 R                3         3600
## # ... with 50 more variables: event_type <chr>, event_description <chr>,
## #   event_detail <chr>, event_team <chr>, event_player_1 <chr>,
## #   event_player_2 <chr>, event_player_3 <chr>, event_length <int>,
## #   coords_x <int>, coords_y <int>, players_substituted <chr>,
## #   home_on_1 <chr>, home_on_2 <chr>, home_on_3 <chr>, home_on_4 <chr>,
## #   home_on_5 <chr>, home_on_6 <chr>, away_on_1 <chr>, away_on_2 <chr>,
## #   away_on_3 <chr>, away_on_4 <chr>, away_on_5 <chr>, away_on_6 <chr>,
## #   home_goalie <chr>, away_goalie <chr>, home_team <chr>,
## #   away_team <chr>, home_skaters <int>, away_skaters <int>,
## #   home_score <dbl>, away_score <dbl>, game_score_state <chr>,
## #   game_strength_state <chr>, highlight_code <chr>, event_distance <dbl>,
## #   event_angle <dbl>, event_rinkside <chr>, event_circle <int>,
## #   event_zone <chr>, home_zone <chr>, prob_goal <dbl>, prob_save <dbl>,
## #   adj_home_corsi <dbl>, adj_home_fenwick <dbl>, adj_home_shot <dbl>,
## #   adj_home_goal <dbl>, adj_away_corsi <dbl>, adj_away_fenwick <dbl>,
## #   adj_away_shot <dbl>, adj_away_goal <dbl>
```

This looks great. We should now have the games results for each game, complete with the right final scores. Time to move onto step 2

Step 2: Summarize Results by team

The output created above does a good job of discerning game results, but a poor job of giving us any tangible information about team results. In this step I hope to solve that problem. The first step is to grab just the columns that we need and add some columns to fill with data that we might need:

```
  #Select only columns needed and add columns for the number of points earned by each team, game totals
  game_results <-
    results_20172018 %>%
    arrange(game_date) %>%
    select(season, session, game_date, game_id, game_period, home_team, away_team, home_score, away_sco
    mutate(
      home_points = ifelse(home_score > away_score, 2, ifelse(game_period == 3, 0, 1)),
      away_points = ifelse(away_score > home_score, 2, ifelse(game_period == 3, 0, 1)),
      home_game_num = NA,
      home_point_total = NA,
      away_game_num = NA,
      away_point_total = NA)
```

The above code does the following: 1. Arranges game from earliest to lastest in the season based on date 2. Selects only the columns that we will need in the future and removes all others 3. Adds columns for points earned by the home team and points earned by the away team 4. Creates dummy columns for the number of games played by each team and for the total number of points through the current game

To populate the columns created in step 4 I'll use 2 functions:

```
fun.game_count <- function(data_set, gdate, team) {
  #DESCRIPTION - Grabs the cumulative game count for a given team
  #ARGUMENTS - data_set = a list of game results created by fun.game_endings, gdate = a given date, tea
```

```
  #Uses sum to build an array of games up to a given data that includes a set team
  game_count <-
    sum(data_set$game_date <= gdate &
          (data_set$home_team == team | data_set$away_team == team))

  return(game_count)
}

fun.point_total <- function(data_set, gdate, team) {
  #DESCRIPTION - Calculates a team's total points earned through a given date
  #ARGUMENTS - data_set = a list of game results created by fun.game_endings, gdate = a given date, tea

  point_count <-
    sum((data_set$game_date <= gdate) *
          (((data_set$home_team == team) * (data_set$home_points)) +
             ((data_set$away_team == team) * (data_set$away_points))))

  return(point_count)
}
```

Each function above is set to take a game date, filter on all games played on or before that date, and sums the points or game count for the given team. I've combined these functions with the code above that summarizes each game to create the below function:

```
fun.game_result_summary <- function(end_games) {
  #DESCRIPTION - take raw PbP data and summarize each game by score, games played, points earned, and c
  #ARGUMENTS - expects raw PbP data frame

  #Select only columns needed and add columns for the number of points earned by each team, game totals
  game_results <-
    end_games %>%
    arrange(game_date) %>%
    select(season, session, game_date, game_id, game_period, home_team, away_team, home_score, away_sco
    mutate(
      home_points = ifelse(home_score > away_score, 2, ifelse(game_period == 3, 0, 1)),
      away_points = ifelse(away_score > home_score, 2, ifelse(game_period == 3, 0, 1)),
      home_game_num = NA,
      home_point_total = NA,
      away_game_num = NA,
      away_point_total = NA)


  #Removes any dublicate lines that may show up, most frequently in games that have two different game
  game_results <- unique(game_results)

  #Loop through each line to apply game and point count functions
  for (i in 1:nrow(game_results)) {
    game_results[i, "home_game_num"] <-
      fun.game_count(data_set = game_results,
                     gdate = game_results[i, "game_date"],
                     team = as.character(game_results[i, "home_team"]))

    game_results[i, "home_point_total"] <-
      fun.point_total(data_set = game_results,
```

```
                    gdate = game_results[i, "game_date"],
                    team = as.character(game_results[i, "home_team"]))

    game_results[i, "away_game_num"] <-
      fun.game_count(data_set = game_results,
                    gdate = game_results[i, "game_date"],
                    team = as.character(game_results[i, "away_team"]))

    game_results[i, "away_point_total"] <-
      fun.point_total(data_set = game_results,
                    gdate = game_results[i, "game_date"],
                    team = as.character(game_results[i, "away_team"]))
  }

  return(game_results)
}
```

We can then go ahead and summarize each of these games with our new data points

```
gamesummary_20172018 <- fun.game_result_summary(results_20172018)

head(gamesummary_20172018, n = 10)
```

```
## # A tibble: 10 x 15
##    season session game_date  game_id game_period home_team away_team
##     <int> <chr>   <date>       <int>       <int> <chr>     <chr>
##  1 2.02e7 R       2017-10-04  2.02e9           3 WPG       TOR
##  2 2.02e7 R       2017-10-04  2.02e9           4 PIT       STL
##  3 2.02e7 R       2017-10-04  2.02e9           3 EDM       CGY
##  4 2.02e7 R       2017-10-04  2.02e9           3 S.J       PHI
##  5 2.02e7 R       2017-10-05  2.02e9           3 BOS       NSH
##  6 2.02e7 R       2017-10-05  2.02e9           5 BUF       MTL
##  7 2.02e7 R       2017-10-05  2.02e9           3 NYR       COL
##  8 2.02e7 R       2017-10-05  2.02e9           5 OTT       WSH
##  9 2.02e7 R       2017-10-05  2.02e9           3 DET       MIN
## 10 2.02e7 R       2017-10-05  2.02e9           3 CHI       PIT
## # ... with 8 more variables: home_score <dbl>, away_score <dbl>,
## #   home_points <dbl>, away_points <dbl>, home_game_num <int>,
## #   home_point_total <dbl>, away_game_num <int>, away_point_total <dbl>
```

This looks great, however since this analysis is ultimately about individual team performance I think this data may be better suited being summarized with only one team per line. In it's current form there will be a lot of identifying which team is the home team in a given scenario and which is the away team. I came up with the following function to help me clean this up even further:

```
fun.results_by_team <- function(game_results) {
  #DESCRIPTION - Takes a summary file generated by fun.game_result_summary and summarizes for each team
  #ARGUMENTS - game_results expects a dataframe output by fun.game_result_summary

  home_list <-
    rename(game_results,
           team = home_team,
           opp = away_team,
           team_score = home_score,
           opp_score = away_score,
```

```
            team_points = home_points,
            opp_points = away_points,
            team_game = home_game_num,
            opp_game = away_game_num,
            team_point_total = home_point_total,
            opp_point_total = away_point_total) %>%
    mutate(side = "home",
            result = ifelse(team_score > opp_score, "W", ifelse(game_period == 3, "L", "OTL")))

  away_list <-
    rename(game_results,
            team = away_team,
            opp = home_team,
            team_score = away_score,
            opp_score = home_score,
            team_points = away_points,
            opp_points = home_points,
            team_game = away_game_num,
            opp_game = home_game_num,
            team_point_total = away_point_total,
            opp_point_total = home_point_total) %>%
    mutate(side = "away",
            result = ifelse(team_score > opp_score, "W", ifelse(game_period == 3, "L", "OTL")))

  team_list <- rbind(home_list, away_list)

  return(team_list)
}
```

The above function creates 2 different data sets. One for all home teams, with the away team listed in the "opp" column, and then all of the relevant columns renamed to reflect that change. The second data set is the same methodology for the away teams. Finally, the two sets are joined together to yield the following:

```
teamsummary_20172018 <- fun.results_by_team(gamesummary_20172018)

head(teamsummary_20172018, n = 10)
```

```
## # A tibble: 10 x 17
##    season session game_date  game_id game_period team  opp   team_score
##     <int> <chr>   <date>       <int>       <int> <chr> <chr>      <dbl>
##  1 2.02e7 R       2017-10-04  2.02e9           3 WPG   TOR            2
##  2 2.02e7 R       2017-10-04  2.02e9           4 PIT   STL            4
##  3 2.02e7 R       2017-10-04  2.02e9           3 EDM   CGY            3
##  4 2.02e7 R       2017-10-04  2.02e9           3 S.J   PHI            3
##  5 2.02e7 R       2017-10-05  2.02e9           3 BOS   NSH            4
##  6 2.02e7 R       2017-10-05  2.02e9           5 BUF   MTL            2
##  7 2.02e7 R       2017-10-05  2.02e9           3 NYR   COL            2
##  8 2.02e7 R       2017-10-05  2.02e9           5 OTT   WSH            4
##  9 2.02e7 R       2017-10-05  2.02e9           3 DET   MIN            4
## 10 2.02e7 R       2017-10-05  2.02e9           3 CHI   PIT           10
## # ... with 9 more variables: opp_score <dbl>, team_points <dbl>,
## #   opp_points <dbl>, team_game <int>, team_point_total <dbl>,
## #   opp_game <int>, opp_point_total <dbl>, side <chr>, result <chr>
```

Step 3: Repeat for seasons with available PBP data and combine into one file

As I found the hard way, the methodology outlined above works just fine but unfortunately my computer is unable to manage this for all of the seasons from 2007 to present. The easiest solution was to create a loop to load in each season's PBP file individually, summarize the game results, save the results, and then bind all of those files together

Now I just need to combine the files using an existing function that I wrote for reading in all .csv files in a directory and binding them

Anticipating that they may come in handy later, I've also written in some code that will designate the Conference and Division alignments as well

```
df.alignment_20002011 <- data.frame(row.names = c("PHI", "NYR", "NYI", "PIT", "N.J",
                                                  "BUF", "BOS", "MTL", "OTT", "TOR",
                                                  "WSH", "T.B", "FLA", "CAR", "ATL",
                                                  "DET", "STL", "CHI", "NSH", "CBJ",
                                                  "VAN", "EDM", "CGY", "COL", "MIN",
                                                  "DAL", "ARI", "ANA", "S.J", "L.A"),

                                    conference = c("East", "East", "East", "East", "East",
                                                   "East", "East", "East", "East", "East",
                                                   "East", "East", "East", "East", "East",
                                                   "West", "West", "West", "West", "West",
                                                   "West", "West", "West", "West", "West",
                                                   "West", "West", "West", "West", "West"),

                                    division = c("Atlantic", "Atlantic", "Atlantic", "Atlantic", "Atlan
                                                 "Northeast", "Northeast", "Northeast", "Northeast", "N
                                                 "Southeast", "Southeast", "Southeast", "Southeast", "S
                                                 "Central", "Central", "Central", "Central", "Central",
                                                 "Northwest", "Northwest", "Northwest", "Northwest", "N
                                                 "Pacific", "Pacific", "Pacific", "Pacific", "Pacific"))

df.alignment_20112013 <- data.frame(row.names = c("PHI", "NYR", "NYI", "PIT", "N.J",
                                                  "BUF", "BOS", "MTL", "OTT", "TOR",
                                                  "WSH", "T.B", "FLA", "CAR", "WPG",
                                                  "DET", "STL", "CHI", "NSH", "CBJ",
                                                  "VAN", "EDM", "CGY", "COL", "MIN",
                                                  "DAL", "ARI", "ANA", "S.J", "L.A"),

                                    conference = c("East", "East", "East", "East", "East",
                                                   "East", "East", "East", "East", "East",
                                                   "East", "East", "East", "East", "East",
                                                   "West", "West", "West", "West", "West",
                                                   "West", "West", "West", "West", "West",
                                                   "West", "West", "West", "West", "West"),

                                    division = c("Atlantic", "Atlantic", "Atlantic", "Atlantic", "Atlan
                                                 "Northeast", "Northeast", "Northeast", "Northeast", "N
                                                 "Southeast", "Southeast", "Southeast", "Southeast", "S
                                                 "Central", "Central", "Central", "Central", "Central",
                                                 "Northwest", "Northwest", "Northwest", "Northwest", "N
                                                 "Pacific", "Pacific", "Pacific", "Pacific", "Pacific"))
```

```
df.alignment_present <- data.frame(row.names = c("BUF", "BOS", "MTL", "OTT", "TOR", "T.B", "FLA", "DET"
                                                 "WSH", "CAR", "PHI", "NYR", "NYI", "PIT", "N.J", "CBJ"
                                                 "VAN", "EDM", "CGY", "ARI", "ANA", "S.J", "L.A", "VGK"
                                                 "STL", "CHI", "COL", "MIN", "DAL", "WPG", "NSH"),

                                   conference = c("East", "East", "East", "East", "East", "East", "Eas
                                                  "East", "East", "East", "East", "East", "East", "Eas
                                                  "West", "West", "West", "West", "West", "West", "Wes
                                                  "West", "West", "West", "West", "West", "West", "Wes

                                   division = c("Atlantic", "Atlantic", "Atlantic", "Atlantic", "Atlan
                                                "Metro", "Metro", "Metro", "Metro", "Metro", "Metro",
                                                "Pacific", "Pacific", "Pacific", "Pacific", "Pacific",
                                                "Central", "Central", "Central", "Central", "Central",


all_seasons <- read_csv("~/Data_Sets/Hockey/Enhanced PbP/test_summaries/combined_seasons.csv", col_names

all_seasons$conference <- NA
all_seasons$division <- NA

for (i in 1:nrow(all_seasons)) {
  team = as.character(all_seasons[i, "team"])

  if (all_seasons[i, "season"] <= 20102011) {
    all_seasons[i, "conference"] = as.character(df.alignment_20002011[team, "conference"])
    all_seasons[i, "division"] = as.character(df.alignment_20002011[team, "division"])
  } else if (all_seasons[i, "season"] <= 20122013) {
    all_seasons[i, "conference"] = as.character(df.alignment_20112013[team, "conference"])
    all_seasons[i, "division"] = as.character(df.alignment_20112013[team, "division"])
  } else {
    all_seasons[i, "conference"] = as.character(df.alignment_present[team, "conference"])
    all_seasons[i, "division"] = as.character(df.alignment_present[team, "division"])
  }
}
```

Step 4: Identify Playoff teams and Non-Playoff Teams

This step should be pretty easy using the resulting dataframe above:

```
postseason_teams <-
  all_seasons %>%
  filter(session == "P") %>%
  select(season, conference, division, team) %>%
  unique()

postseason_teams
```

```
## # A tibble: 176 x 4
##      season conference division  team
##       <int> <chr>      <chr>     <chr>
##  1 20072008 East       Atlantic  PIT
##  2 20072008 East       Atlantic  N.J
```

```
##  3 20072008 West       Pacific   S.J
##  4 20072008 West       Northwest MIN
##  5 20072008 East       Northeast MTL
##  6 20072008 West       Central   DET
##  7 20072008 West       Pacific   ANA
##  8 20072008 East       Southeast WSH
##  9 20072008 East       Northeast BOS
## 10 20072008 East       Atlantic  NYR
## # ... with 166 more rows
```

Step 5: Group Dataset by game and identify min/min point values for postseason teams and non postseason teams

```
#Grab only regular season
reg_season <- filter(all_seasons, session == "R")

#Cycle through each row and assign a "Y" or "N" based on if the team made the playoffs or not
for (i in 1:nrow(reg_season)) {
  team_name <- as.character(reg_season[i, "team"])
  season_ID <- as.integer(reg_season[i, "season"])

  if(team_name %in% filter(postseason_teams, season == season_ID)$team) {
    reg_season[i, "playoff_team"] <- "Y"
  } else {
    reg_season[i, "playoff_team"] <- "N"
  }
}

#Filter summaries into playoff and non-playoff teams
playoff_teams_summary <- filter(reg_season, playoff_team == "Y")
nonplayoff_teams_summary <- filter(reg_season, playoff_team == "N")

#Grab min/max points for each game of the season for playoff teams
playoff_table <-
  playoff_teams_summary %>%
  group_by(team_game) %>%
  summarise(min_playoff = min(team_point_total),
            max_playoff = max(team_point_total))

#Grab min/max points for each game of the season for nonplayoff teams
nonplayoff_table <-
  nonplayoff_teams_summary %>%
  group_by(team_game) %>%
  summarise(min_nonplayoff = min(team_point_total),
            max_nonplayoff = max(team_point_total))

#Join playoff and non-playoff lists
final_list <- join(playoff_table, nonplayoff_table, by = "team_game")

#Cycle through each row and calculate the lock line and point of no return
for (i in 1:82) {
```

```
  #Lock Line
  if(final_list[i, "max_nonplayoff"] + 1 <= final_list[i, "team_game"] * 2) {
    final_list[i, "lock_line"] <- final_list[i, "max_nonplayoff"] + 1
  } else {
    final_list[i, "lock_line"] <- NA
  }

  #Point of No Return
  if(final_list[i, "min_playoff"] - 1 <= final_list[i, "team_game"] * 2 & final_list[i, "min_playoff"] -
    final_list[i, "no_return"] <- (final_list[i, "min_playoff"] - 1)
  } else {
    final_list[i, "no_return"] <- NA
  }
}

#Grab final results
lock_line_results <-
  final_list %>%
  select(team_game, lock_line, no_return)

lock_line_results
```

```
##    team_game lock_line no_return
## 1          1        NA        NA
## 2          2        NA        NA
## 3          3        NA        NA
## 4          4        NA        NA
## 5          5        NA         1
## 6          6        NA         1
## 7          7        NA         1
## 8          8        NA         1
## 9          9        NA         3
## 10        10        19         3
## 11        11        19         4
## 12        12        21         5
## 13        13        23         7
## 14        14        24         7
## 15        15        26         8
## 16        16        28        10
## 17        17        29        12
## 18        18        29        12
## 19        19        31        12
## 20        20        31        12
## 21        21        33        12
## 22        22        35        14
## 23        23        37        16
## 24        24        39        16
## 25        25        40        17
## 26        26        42        17
## 27        27        42        19
## 28        28        42        19
## 29        29        42        21
## 30        30        44        23
## 31        31        45        25
```

```
## 32         32         45         25
## 33         33         46         27
## 34         34         47         28
## 35         35         47         28
## 36         36         47         29
## 37         37         49         30
## 38         38         49         30
## 39         39         51         32
## 40         40         53         34
## 41         41         54         34
## 42         42         56         34
## 43         43         56         36
## 44         44         58         37
## 45         45         60         39
## 46         46         62         41
## 47         47         64         41
## 48         48         64         43
## 49         49         64         44
## 50         50         66         46
## 51         51         66         47
## 52         52         66         49
## 53         53         66         49
## 54         54         68         50
## 55         55         69         52
## 56         56         71         54
## 57         57         72         54
## 58         58         73         56
## 59         59         73         57
## 60         60         73         59
## 61         61         75         61
## 62         62         75         62
## 63         63         77         63
## 64         64         78         65
## 65         65         80         65
## 66         66         81         67
## 67         67         82         67
## 68         68         84         69
## 69         69         85         69
## 70         70         87         71
## 71         71         87         72
## 72         72         87         74
## 73         73         87         76
## 74         74         89         78
## 75         75         90         79
## 76         76         92         81
## 77         77         93         81
## 78         78         94         81
## 79         79         96         83
## 80         80         96         85
## 81         81         96         85
## 82         82         97         86
```

Bonus: Visualization and Analysis

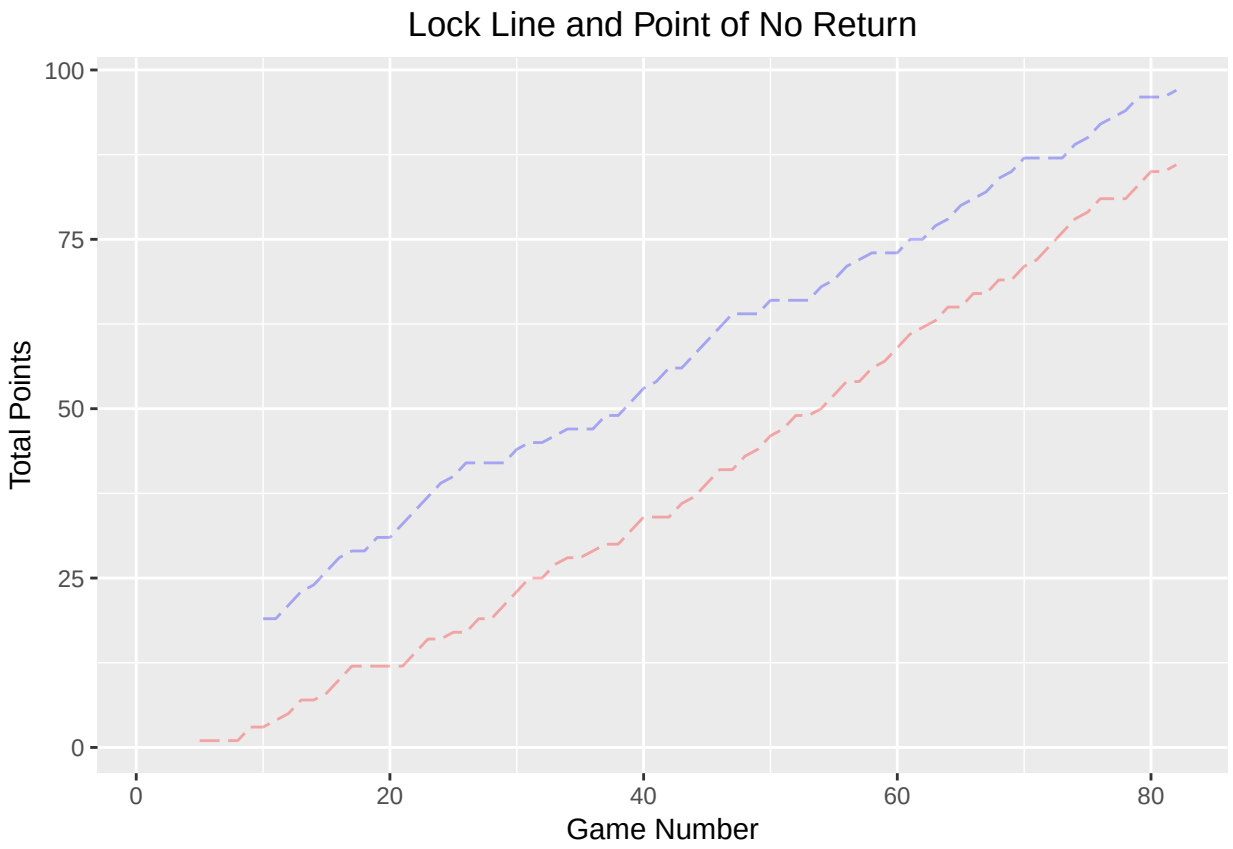
Let's start by taking a look at the lock line results

```
require(ggplot2)
```

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

```
viz.lockline_ponr <-
  ggplot() +
  ggtitle("Lock Line and Point of No Return") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "none") +
  labs(x = "Game Number", y = "Total Points", color = "Team") +
  geom_line(data = lock_line_results, na.rm = TRUE, mapping = aes(x = team_game, y = lock_line), color =
  geom_line(data = lock_line_results, na.rm = TRUE, mapping = aes(x = team_game, y = no_return), color =

viz.lockline_ponr
```



The first thing that stands out is that neither line starts at zero games. The point of no return doesn't start until game 5, whereas the lock line doesn't start until game 10. Considering that the point of no return is the point where no team has made the playoffs, it would stand to reason that no line exists because at least one team in our set has started out the season 0-4 and still made the playoffs.

```
#Identify team that made playoffs after starting 0-4
reg_season %>%
  filter(playoff_team == "Y" & team_game == 4 & team_point_total == 0) %>%
  select(team, season, team_game, team_point_total)
```

```
## # A tibble: 2 x 4
```

```
##   team    season team_game team_point_total
##   <chr>    <int>     <int>            <int>
## 1 ANA   20082009         4                0
## 2 ANA   20082009         4                0
```

Conversely, a lock line that doesn't start until game 10 must mean that there was at least one team that missed the playoffs after starting the season 9-0

```
#Identify team that made playoffs after starting 9-0
reg_season %>%
  filter(playoff_team == "N" & team_game == 9 & team_point_total == 18) %>%
  select(team, season, team_game, team_point_total)
```

```
## # A tibble: 2 x 4
##   team    season team_game team_point_total
##   <chr>    <int>     <int>            <int>
## 1 MTL   20152016         9               18
## 2 MTL   20152016         9               18
```

It's quite interesting that based on historical results, a team that starts out 0-4 may still have a chance of making the playoffs, but a team that goes on a 9-0 run to start the season still isn't considered "safe". Let's take a look at how the Ducks turned their season around in 2008-2009 and where it all went wrong for The Habs in 2015-2016

```
team_results <-
  reg_season %>%
  filter((team == "ANA" & season == 20082009) | (team == "MTL" & season == 20152016))

line_labels <-
  team_results %>%
  group_by(team) %>%
  summarise(points = max(team_point_total),
            games = max(team_game)) %>%
            mutate(label = paste(team, points, sep = ", "))

viz.team_results <-
  viz.lockline_ponr +
  geom_line(data = team_results, mapping = aes(x = team_game, y = team_point_total, color = team)) +
  geom_text(data = line_labels, mapping = aes(x = games, y = points, label = label, color = team), nudg
```

To be honest the Ducks run to right the ship after starting 0-4 is pretty much what you would expect. While no one wants to start out by dropping 8 possible points, with a few strong win streaks it's possible to get back in the hunt. That appears to be what happened in this case. Let's take a looke at games 5 through 25 of the season

```
head(filter(team_results, team == "ANA" & team_game >= 5 & team_game <= 25), n = 21)
```

```
## # A tibble: 21 x 20
##    season session game_date  game_id game_period team  opp   team_score
##     <int> <chr>   <date>       <int>       <int> <chr> <chr>      <int>
##  1 2.01e7 R       2008-10-17  2.01e9           3 ANA   S.J            4
##  2 2.01e7 R       2008-10-19  2.01e9           3 ANA   CAR            1
##  3 2.01e7 R       2008-10-29  2.01e9           4 ANA   DET            5
##  4 2.01e7 R       2008-10-31  2.01e9           5 ANA   VAN            6
##  5 2.01e7 R       2008-11-02  2.01e9           3 ANA   CGY            3
##  6 2.01e7 R       2008-11-05  2.01e9           3 ANA   STL            5
##  7 2.01e7 R       2008-11-07  2.01e9           3 ANA   DAL            2
```

```
##  8 2.01e7 R       2008-11-09  2.01e9           3 ANA   FLA            1
##  9 2.01e7 R       2008-11-14  2.01e9           4 ANA   NSH            3
## 10 2.01e7 R       2008-11-16  2.01e9           3 ANA   L.A            2
## # ... with 11 more rows, and 12 more variables: opp_score <int>,
## #   team_points <int>, opp_points <int>, team_game <int>,
## #   team_point_total <int>, opp_game <int>, opp_point_total <int>,
## #   side <chr>, result <chr>, conference <chr>, division <chr>,
## #   playoff_team <chr>
```

The Ducks recovered from a rough start by going 14-4-3 in their next 21 games for a 73.8% points percentage. They also ended the season on a 7-2-1 run to end the season with a total of 91 points which was good enough for second in the Pacific.

The story isn't nearly as fun for fans of the 2015-16 Habs. After a blistering 9-0 start, Montreal was able to keep up a pretty strong pace through 26 games with a 19-4-2 record for a 80.8% points percentage. Unfortunately a number of very significant injuries (the most significant of which was an MCL sprain to Carey Price that sidelined him for much of the season) were just too much to overcome. Over the next 26 games the team went 5-1-20 and the season was in real jeopardy.

```
head(filter(team_results, team == "MTL" & team_game >= 27 & team_game <= 52), n = 26)
```

```
## # A tibble: 26 x 20
##    season session game_date  game_id game_period team  opp   team_score
##     <int> <chr>   <date>       <int>       <int> <chr> <chr>      <int>
##  1 2.02e7 R       2015-12-03  2.02e9           3 MTL   WSH            2
##  2 2.02e7 R       2015-12-09  2.02e9           3 MTL   BOS            1
##  3 2.02e7 R       2015-12-12  2.02e9           3 MTL   OTT            3
##  4 2.02e7 R       2015-12-15  2.02e9           3 MTL   S.J            1
##  5 2.02e7 R       2015-12-17  2.02e9           3 MTL   L.A            0
##  6 2.02e7 R       2016-01-06  2.02e9           3 MTL   N.J            2
##  7 2.02e7 R       2016-01-09  2.02e9           3 MTL   PIT            1
##  8 2.02e7 R       2016-01-14  2.02e9           3 MTL   CHI            1
##  9 2.02e7 R       2016-01-19  2.02e9           3 MTL   BOS            1
## 10 2.02e7 R       2016-01-26  2.02e9           3 MTL   CBJ            2
## # ... with 16 more rows, and 12 more variables: opp_score <int>,
## #   team_points <int>, opp_points <int>, team_game <int>,
## #   team_point_total <int>, opp_game <int>, opp_point_total <int>,
## #   side <chr>, result <chr>, conference <chr>, division <chr>,
## #   playoff_team <chr>
```

The team first crossed the point of no return after their 66th game on March 3rd, 2016 after a 4-2 road loss to Winnipeg. The team would continue to hover around the point of no return for a number of games before ultimately ending the season with a disappointing 82 points, good for 6th in the Atlantic.

Next Steps

This has been a great project and is pretty fun to plug in different teams to see how they trended, but as far as using the lock line and point of no return as a model for who could make the playoffs and who is likely out, it leaves something to be desired. There are a number of items that may be able to fix that.

1. *Expand number of teams included in line:* Instead of just identifying the historically best and worst point totals, there is some room to adjust the lock line and point of no return to identify the top 10% of teams, set number of teams, etc. to better identify who is most likely in or out
2. *Focus on points out of a playoff spot:* We know that point totals do not occur in a vacuum. Getting into the playoffs also depends on the performance of the teams around you. In order to adjust for that,

it would be interesting to re-run this analysis to identify how many points out of the playoffs a team is at any given point of the season

3. *Player Injuries:* Similar to Carey Price's injury above, it would be interesting to layer in some significant injuries to see whether or not those injuries were significant enough to change a team's overall playoff outlook