Shawn Michael

MSDS 692

Regis University

NHL Salaries Predictions 2017/2018 Stats

The National Hockey League salaries during the 2017/2018 season ranged from the league min of $650,000 - $13.8 million. The intent of the project is to model several Machine Learning techniques to predict player salaries. In addition, to the modeling, it is also important to clean, understand and discover meaning from the dataset.

NOTE: Due to the size of the legend, I have decided to include it as an imported document, into R.

Loading Libraries

```
# libraries required for exploring data
library(readr)
library(dplyr)
library(lattice)
library(gmodels)
library(ggplot2)
library(caret)
library(skimr)
library(GGally)
library(caret)
library(rpart)
library(rpart.plot) # loading rpart.plot to view the tree
library(neuralnet) # Calling neuralnet library
```

Legend File

```
# importing legend as a df
# mac book path
# read file and assign NA to missing values
#hockey_legend <- read.csv("~/OneDrive/Regis/MSDS 692/Final Project/Legend.csv")


hockey_legend <- read.csv("C:/Users/shawn/OneDrive/Regis/MSDS 692/Final Project/Legend.csv")
```

Functions

```
# mean absolute error
MAE <- function(actual, predicted) {
  mean(abs(actual - predicted))
}
```

```
# Function that returns Root Mean Squared Error
rmse <- function(predicted, actual) {
```

```
    sqrt(mean((predicted - actual)^2))
}
```

Reading Data

I have been working on both Mac and Windows through this project, therefore I would adjust my path accordingly.

```
# mac book path
# read file and assign NA to missing values
#data <- read.csv("~/OneDrive/Regis/MSDS 692/Final Project/NHL 2017-18.csv", na = c("", "NA"))

#pc path
# read file and assign NA to missing values
data <- read.csv("C:/Users/shawn/OneDrive/Regis/MSDS 692/Final Project/NHL 2017-18.csv",na = c("", "NA"))
```

```
# glimse allows to view the data quickly and in an organized manner
glimpse(data)
```

```
## Observations: 881
## Variables: 63
## $ ï..Born    <fct> 11/21/1989, 12/23/1987, 6/7/1990, 4/18/1982, 3/24/1...
## $ City       <fct> Dorchester, Helsinki, Winnipeg, Regina, Bolton, Reg...
## $ Cntry      <fct> USA, FIN, CAN, CAN, USA, CAN, CAN, USA, USA, CAN, U...
## $ Nat        <fct> USA, FIN, CAN, CAN, USA, CAN, CAN, USA, USA, CAN, U...
## $ Ht         <int> 77, 74, 75, 74, 75, 72, 72, 75, 75, 74, 74, 73, 72,...
## $ Wt         <int> 215, 212, 210, 215, 205, 195, 195, 217, 203, 205, 1...
## $ DftRd      <int> 2, 3, 3, 1, 1, NA, 3, 1, NA, 6, 5, 1, 1, 4, 4, 6, 4...
## $ Ovrl       <int> 60, 65, 69, 6, 13, NA, 82, 18, NA, 159, 127, 28, 29...
## $ Hand       <fct> R, L, R, L, L, L, L, L, L, L, R, R, L, L, R, L, L, ...
## $ Debut      <int> 2011, 2014, 2011, 2000, 2002, 2003, 2010, 2002, 201...
## $ Age        <int> 28, 30, 27, 35, 36, 38, 27, 37, 22, 26, 24, 36, 37,...
## $ Seasons    <int> 7, 4, 7, 17, 15, 14, 8, 15, 1, 4, 1, 17, 14, 5, 8, ...
## $ LastName   <fct> Hayes, Lehtera, Stone, Hartnell, Hainsey, Kunitz, H...
## $ FirstName  <fct> Jimmy, Jori, Michael, Scott, Ron, Chris, Adam, Broo...
## $ Pos        <fct> RW, C, D, LW, D, LW, C, D, D, LW, D, RW, D, D, RW, ...
## $ GP         <int> 33, 62, 82, 62, 80, 82, 81, 81, 1, 7, 24, 82, 79, 1...
## $ G          <int> 3, 3, 3, 13, 4, 13, 24, 0, 0, 0, 1, 16, 4, 0, 4, 40...
## $ A          <int> 6, 5, 7, 11, 19, 16, 26, 10, 1, 2, 1, 35, 23, 5, 4,...
## $ PTS        <int> 9, 8, 10, 24, 23, 29, 50, 10, 1, 2, 2, 51, 27, 5, 8...
## $ PTSvsGP    <dbl> 0.27, 0.13, 0.12, 0.39, 0.29, 0.35, 0.62, 0.12, 1.0...
## $ PAX        <dbl> -2.0, -20.4, -14.4, -11.7, 0.7, -24.8, 4.3, -5.7, 0...
## $ GC         <dbl> 3.4, 2.9, 3.5, 9.9, 7.8, 11.2, 19.9, 2.9, 0.3, 0.6,...
## $ PlusMinus  <int> -2, -8, -11, -3, 12, 8, 11, -9, 1, 3, 2, -9, -14, 4...
## $ PIM        <int> 6, 14, 28, 82, 20, 35, 20, 68, 0, 11, 0, 56, 36, 0,...
## $ Shifts     <int> 491, 952, 1759, 1099, 2215, 1383, 1844, 1797, 14, 8...
## $ TOI        <int> 23617, 39114, 82396, 44707, 104951, 58776, 86436, 9...
## $ TOIvsGP    <dbl> 11.93, 10.51, 16.75, 12.02, 21.86, 11.95, 17.79, 19...
## $ TOIpercent <dbl> 20.18, 17.88, 28.08, 20.52, 36.52, 20.19, 29.24, 32...
## $ IPPper     <fct> 47.40%, 47.10%, 24.40%, 70.60%, 29.90%, 74.40%, 57....
## $ iCF        <int> 64, 74, 251, 133, 190, 133, 264, 133, 1, 22, 63, 35...
## $ iSF        <int> 37, 45, 98, 82, 73, 81, 151, 54, 0, 11, 27, 198, 76...
## $ iSCF       <int> 42, 41, 57, 79, 53, 78, 183, 33, 0, 15, 12, 183, 49...
## $ iRush      <int> 0, 1, 1, 5, 5, 5, 12, 0, 0, 1, 2, 10, 5, 1, 0, 7, 7...
```

```
## $ sDist      <dbl> 19.2432, 26.1111, 55.4286, 32.5732, 65.3288, 28.530...
## $ Pass       <dbl> 71.5, 38.9, 113.3, 181.6, 148.8, 184.0, 245.2, 56.2...
## $ TOff       <dbl> 135.5, 112.9, 364.3, 314.6, 338.8, 317.0, 509.2, 18...
## $ OZS        <int> 134, 165, 381, 290, 429, 268, 495, 384, 4, 12, 112,...
## $ NZS        <int> 109, 178, 447, 242, 671, 398, 395, 500, 1, 20, 91, ...
## $ DZS        <int> 114, 181, 469, 220, 890, 309, 488, 694, 2, 12, 108,...
## $ iHF        <int> 50, 55, 106, 82, 91, 131, 61, 218, 1, 8, 20, 79, 82...
## $ iHA        <int> 39, 76, 94, 41, 66, 81, 78, 113, 0, 4, 23, 146, 65,...
## $ iHDf       <int> 11, -21, 11, 41, 25, 50, -17, 105, 1, 4, -3, -67, 1...
## $ iMiss      <int> 10, 10, 61, 25, 43, 24, 61, 21, 0, 5, 19, 76, 21, 6...
## $ iGVA       <int> 8, 13, 31, 25, 85, 29, 29, 44, 0, 2, 14, 85, 52, 7,...
## $ iTKA       <int> 15, 14, 15, 14, 18, 38, 40, 17, 0, 3, 5, 46, 14, 4,...
## $ iBLK       <int> 6, 21, 158, 25, 169, 30, 61, 168, 1, 1, 18, 42, 107...
## $ BLKpercent <fct> 2.30%, 4.60%, 12.80%, 5.00%, 9.50%, 4.30%, 5.50%, 1...
## $ Hat        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ OTG        <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ FirstGoal  <int> 1, 0, 0, 4, 1, 4, 7, 0, 0, 0, 0, 5, 1, 0, 1, 7, 1, ...
## $ GWG        <int> 2, 0, 0, 2, 2, 2, 8, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, ...
## $ ENG        <int> 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, ...
## $ PSG        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PSA        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ G.Bkhd     <int> 0, 0, 0, 0, 0, 2, 5, 0, 0, 0, 0, 2, 0, 0, 0, 8, 0, ...
## $ G.Dflct    <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, ...
## $ G.Slap     <int> 1, 0, 3, 1, 2, 1, 1, 0, 0, 0, 0, 2, 3, 0, 0, 2, 1, ...
## $ G.Snap     <int> 1, 0, 0, 1, 2, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, ...
## $ G.Tip      <int> 0, 0, 0, 5, 0, 4, 2, 0, 0, 0, 0, 2, 0, 0, 0, 9, 0, ...
## $ G.Wrap     <int> 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ...
## $ G.Wrst     <int> 1, 3, 0, 5, 0, 6, 8, 0, 0, 0, 1, 9, 1, 0, 2, 17, 2,...
## $ Status     <fct> UFA, UFA, UFA, UFA, UFA, UFA, UFA, UFA, RFA, UFA, R...
## $ Salary     <fct> "$700,000 ", "$4,700,000 ", "$3,500,000 ", "$1,000,...
```

Data Cleaning

```
# cleaning up first column name and setting it as Date
# opening on Windows I receive unique char
names(data)[1]<-"Born"

# add the first column as date with the formatting
data$Born <- as.Date(data$Born, "%m/%d/%Y")
```

Identifying the columns with NA

```
# search for document for columns with NA
col_NA <- colnames(data)[colSums(is.na(data)) > 0]

# print columns with NA
col_NA
```

```
## [1] "DftRd" "Ovrl"
```

There are many players in the NHL that were never drafted, so this missing data is expected, therefore I will leave it as is. The same applies to Overall (Drafted Overall)

Cleaning dataset to utilize the predictor

```
#view the Salary column
head(data$Salary)
```

```
## [1] $700,000    $4,700,000  $3,500,000  $1,000,000  $3,600,000  $2,000,000
## 158 Levels: $1,000,000  $1,050,000  $1,100,000  $1,125,000  ... $975,000
```

The data is not usable in the current state ($,), therefore we need to clean the data in order to use it in the model.

```
# create a new column and replace $ with blank "" in the data$Salary column
data$Salary1 = gsub("\\$", "", data$Salary)

# remove the , from the string and convert to numeric
data$Salary1 = as.numeric(gsub("\\,", "", data$Salary1))

#view transformed data
head(data$Salary1)
```

```
## [1]  700000 4700000 3500000 1000000 3600000 2000000
```

Data Exploration

```
# search and view the players that were not drafted. DrtRd "Draft Round"
hockey <- data %>%
  filter(is.na(DftRd))

#Display top 5 rows with DrfRd as NA
head(hockey, 5)
```

```
##         Born        City Cntry Nat Ht  Wt DftRd Ovrl Hand Debut Age
## 1 1979-09-26      Regina   CAN CAN 72 195    NA   NA    L  2003  38
## 2 1995-03-30       SANDY   USA USA 75 203    NA   NA    L  2017  22
## 3 1988-08-24 Maple Ridge   CAN CAN 69 187    NA   NA    L  2013  29
## 4 1991-04-12     Livonia   USA USA 69 186    NA   NA    L  2011  26
## 5 1990-07-02  Burnsville   USA USA 70 169    NA   NA    R  2011  27
##   Seasons LastName FirstName Pos GP  G  A PTS PTSvsGP   PAX   GC PlusMinus
## 1      14   Kunitz     Chris  LW 82 13 16  29    0.35 -24.8 11.2         8
## 2       1 Brickley    Daniel   D  1  0  1   1    1.00   0.0  0.3         1
## 3       5     Hunt      Brad   D 45  3 15  18    0.40   5.7  6.0        -5
## 4       7     Krug     Torey   D 76 14 45  59    0.78  17.5 20.2         0
## 5       6    Brown      J.T.  RW 47  2  5   7    0.15  -3.2  2.5        -5
##   PIM Shifts   TOI TOIvsGP TOIpercent  IPPper iCF iSF iSCF iRush   sDist
## 1  35   1383 58776   11.95      20.19  74.40% 133  81   78     5 28.5309
## 2   0     14   648   10.80      19.01 100.00%   1   0    0     0  0.0000
## 3   6    944 44947   16.65      27.86  45.00% 154  80   45     0 46.4875
## 4  36   1898 93030   20.40      33.77  54.60% 416 197  152     5 48.8782
## 5  24    616 25580    9.07      15.76  77.80%  78  40   41     6 30.9750
##    Pass  TOff OZS NZS DZS iHF iHA iHDf iMiss iGVA iTKA iBLK BLKpercent Hat
## 1 184.0 317.0 268 398 309 131  81   50    24   29   38   30      4.30%   0
## 2   0.0   1.0   4   1   2   1   0    1     0    0    0    1     20.00%   0
## 3 110.3 264.3 330 219 184  19  58  -39    30   11   22   36      7.50%   0
## 4 379.2 795.2 850 504 352  79 111  -32    77   71   35   73      7.20%   0
## 5  93.3 171.3 104 158 116  71  40   31    20   11   11   23      7.50%   0
```

```
##   OTG FirstGoal GWG ENG PSG PSA G.Bkhd G.Dflct G.Slap G.Snap G.Tip G.Wrap
## 1   0         4   2   0   0   0      2       0      1      0     4      0
## 2   0         0   0   0   0   0      0       0      0      0     0      0
## 3   0         0   0   0   0   0      0       0      2      0     0      0
## 4   0         3   4   0   0   0      0       1      8      1     0      0
## 5   0         1   1   0   0   0      0       0      0      2     0      0
##   G.Wrst Status      Salary Salary1
## 1      6    UFA $2,000,000  2000000
## 2      0    RFA   $925,000   925000
## 3      1    UFA   $650,000   650000
## 4      4    UFA $5,500,000  5500000
## 5      0    UFA $1,250,000  1250000
```

```
# glimse is used in tidyverse to view the data's structure
glimpse(data)
```

```
## Observations: 881
## Variables: 64
## $ Born       <date> 1989-11-21, 1987-12-23, 1990-06-07, 1982-04-18, 19...
## $ City       <fct> Dorchester, Helsinki, Winnipeg, Regina, Bolton, Reg...
## $ Cntry      <fct> USA, FIN, CAN, CAN, USA, CAN, CAN, USA, USA, CAN, U...
## $ Nat        <fct> USA, FIN, CAN, CAN, USA, CAN, CAN, USA, USA, CAN, U...
## $ Ht         <int> 77, 74, 75, 74, 75, 72, 72, 75, 75, 74, 74, 73, 72,...
## $ Wt         <int> 215, 212, 210, 215, 205, 195, 195, 217, 203, 205, 1...
## $ DftRd      <int> 2, 3, 3, 1, 1, NA, 3, 1, NA, 6, 5, 1, 1, 4, 4, 6, 4...
## $ Ovrl       <int> 60, 65, 69, 6, 13, NA, 82, 18, NA, 159, 127, 28, 29...
## $ Hand       <fct> R, L, R, L, L, L, L, L, L, L, R, R, L, L, R, L, L, ...
## $ Debut      <int> 2011, 2014, 2011, 2000, 2002, 2003, 2010, 2002, 201...
## $ Age        <int> 28, 30, 27, 35, 36, 38, 27, 37, 22, 26, 24, 36, 37,...
## $ Seasons    <int> 7, 4, 7, 17, 15, 14, 8, 15, 1, 4, 1, 17, 14, 5, 8, ...
## $ LastName   <fct> Hayes, Lehtera, Stone, Hartnell, Hainsey, Kunitz, H...
## $ FirstName  <fct> Jimmy, Jori, Michael, Scott, Ron, Chris, Adam, Broo...
## $ Pos        <fct> RW, C, D, LW, D, LW, C, D, D, LW, D, RW, D, D, RW, ...
## $ GP         <int> 33, 62, 82, 62, 80, 82, 81, 81, 1, 7, 24, 82, 79, 1...
## $ G          <int> 3, 3, 3, 13, 4, 13, 24, 0, 0, 0, 1, 16, 4, 0, 4, 40...
## $ A          <int> 6, 5, 7, 11, 19, 16, 26, 10, 1, 2, 1, 35, 23, 5, 4,...
## $ PTS        <int> 9, 8, 10, 24, 23, 29, 50, 10, 1, 2, 2, 51, 27, 5, 8...
## $ PTSvsGP    <dbl> 0.27, 0.13, 0.12, 0.39, 0.29, 0.35, 0.62, 0.12, 1.0...
## $ PAX        <dbl> -2.0, -20.4, -14.4, -11.7, 0.7, -24.8, 4.3, -5.7, 0...
## $ GC         <dbl> 3.4, 2.9, 3.5, 9.9, 7.8, 11.2, 19.9, 2.9, 0.3, 0.6,...
## $ PlusMinus  <int> -2, -8, -11, -3, 12, 8, 11, -9, 1, 3, 2, -9, -14, 4...
## $ PIM        <int> 6, 14, 28, 82, 20, 35, 20, 68, 0, 11, 0, 56, 36, 0,...
## $ Shifts     <int> 491, 952, 1759, 1099, 2215, 1383, 1844, 1797, 14, 8...
## $ TOI        <int> 23617, 39114, 82396, 44707, 104951, 58776, 86436, 9...
## $ TOIvsGP    <dbl> 11.93, 10.51, 16.75, 12.02, 21.86, 11.95, 17.79, 19...
## $ TOIpercent <dbl> 20.18, 17.88, 28.08, 20.52, 36.52, 20.19, 29.24, 32...
## $ IPPper     <fct> 47.40%, 47.10%, 24.40%, 70.60%, 29.90%, 74.40%, 57....
## $ iCF        <int> 64, 74, 251, 133, 190, 133, 264, 133, 1, 22, 63, 35...
## $ iSF        <int> 37, 45, 98, 82, 73, 81, 151, 54, 0, 11, 27, 198, 76...
## $ iSCF       <int> 42, 41, 57, 79, 53, 78, 183, 33, 0, 15, 12, 183, 49...
## $ iRush      <int> 0, 1, 1, 5, 5, 5, 12, 0, 0, 1, 2, 10, 5, 1, 0, 7, 7...
## $ sDist      <dbl> 19.2432, 26.1111, 55.4286, 32.5732, 65.3288, 28.530...
## $ Pass       <dbl> 71.5, 38.9, 113.3, 181.6, 148.8, 184.0, 245.2, 56.2...
## $ TOff       <dbl> 135.5, 112.9, 364.3, 314.6, 338.8, 317.0, 509.2, 18...
## $ OZS        <int> 134, 165, 381, 290, 429, 268, 495, 384, 4, 12, 112,...
```

```
## $ NZS        <int> 109, 178, 447, 242, 671, 398, 395, 500, 1, 20, 91, ...
## $ DZS        <int> 114, 181, 469, 220, 890, 309, 488, 694, 2, 12, 108,...
## $ iHF        <int> 50, 55, 106, 82, 91, 131, 61, 218, 1, 8, 20, 79, 82...
## $ iHA        <int> 39, 76, 94, 41, 66, 81, 78, 113, 0, 4, 23, 146, 65,...
## $ iHDf       <int> 11, -21, 11, 41, 25, 50, -17, 105, 1, 4, -3, -67, 1...
## $ iMiss      <int> 10, 10, 61, 25, 43, 24, 61, 21, 0, 5, 19, 76, 21, 6...
## $ iGVA       <int> 8, 13, 31, 25, 85, 29, 29, 44, 0, 2, 14, 85, 52, 7,...
## $ iTKA       <int> 15, 14, 15, 14, 18, 38, 40, 17, 0, 3, 5, 46, 14, 4,...
## $ iBLK       <int> 6, 21, 158, 25, 169, 30, 61, 168, 1, 1, 18, 42, 107...
## $ BLKpercent <fct> 2.30%, 4.60%, 12.80%, 5.00%, 9.50%, 4.30%, 5.50%, 1...
## $ Hat        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ OTG        <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ FirstGoal  <int> 1, 0, 0, 4, 1, 4, 7, 0, 0, 0, 0, 5, 1, 0, 1, 7, 1, ...
## $ GWG        <int> 2, 0, 0, 2, 2, 2, 8, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, ...
## $ ENG        <int> 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, ...
## $ PSG        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ PSA        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ G.Bkhd     <int> 0, 0, 0, 0, 0, 2, 5, 0, 0, 0, 0, 2, 0, 0, 0, 8, 0, ...
## $ G.Dflct    <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, ...
## $ G.Slap     <int> 1, 0, 3, 1, 2, 1, 1, 0, 0, 0, 0, 2, 3, 0, 0, 2, 1, ...
## $ G.Snap     <int> 1, 0, 0, 1, 2, 0, 6, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, ...
## $ G.Tip      <int> 0, 0, 0, 5, 0, 4, 2, 0, 0, 0, 0, 2, 0, 0, 0, 9, 0, ...
## $ G.Wrap     <int> 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ...
## $ G.Wrst     <int> 1, 3, 0, 5, 0, 6, 8, 0, 0, 0, 1, 9, 1, 0, 2, 17, 2,...
## $ Status     <fct> UFA, UFA, UFA, UFA, UFA, UFA, UFA, UFA, RFA, UFA, R...
## $ Salary     <fct> "$700,000 ", "$4,700,000 ", "$3,500,000 ", "$1,000,...
## $ Salary1    <dbl> 700000, 4700000, 3500000, 1000000, 3600000, 2000000...
```

Skim()

Using the skim function simply illustrates a number of important features about the dataset in a single view.

```
# using skimr to view and interpret the data quickly
skim(data)
```

```
## Skim summary statistics
##  n obs: 881
##  n variables: 64
##
## -- Variable type:Date ------------------------------------------------------------------------
##  variable missing complete   n        min        max     median n_unique
##      Born       0      881 881 1972-02-15 1999-09-13 1992-03-14      798
##
## -- Variable type:factor ----------------------------------------------------------------------
##    variable missing complete   n n_unique
##  BLKpercent       0      881 881      137
##        City       0      881 881      495
##       Cntry       0      881 881       21
##   FirstName       0      881 881      395
##        Hand       0      881 881        2
##       IPPper       0      881 881      346
##    LastName       0      881 881      818
##         Nat       0      881 881       18
##         Pos       0      881 881        8
##      Salary       0      881 881      158
```

```
##      Status       0      881 881        2
##                                    top_counts ordered
##                                   0.0: 35, 3    FALSE
##    Tor: 34, Edm: 21, Sto: 15, Win: 15          FALSE
##  CAN: 404, USA: 245, SWE: 85, CZE: 31          FALSE
##    Rya: 21, Mat: 18, Nic: 13, Ale: 12          FALSE
##                L: 551, R: 330, NA: 0           FALSE
##                                   0.0: 81, 50   FALSE
##       Smi: 5, Bro: 4, Joh: 3, Kar: 3           FALSE
##  CAN: 402, USA: 246, SWE: 87, FIN: 33          FALSE
##     D: 303, C: 232, LW: 171, RW: 137           FALSE
##  $65: 115, $92: 115, $83: 36, $5,: 29          FALSE
##            UFA: 500, RFA: 381, NA: 0           FALSE
##
## -- Variable type:integer ------------------------------------------------------------------------
##   variable missing complete   n      mean       sd   p0   p25   p50   p75
##          A       0      881 881    14.24     14.14    0     3    10    21
##        Age       0      881 881    26.08      4.31   18    23    25    29
##      Debut       0      881 881  2014.38     67.62 1990  2009  2013  2016
##      DftRd     126      755 881     2.74      1.98    1     1     2     4
##        DZS       0      881 881   282.64    213.11    0    88   275   420
##        ENG       0      881 881     0.4       0.87    0     0     0     0
##  FirstGoal       0      881 881     1.44      1.83    0     0     1     2
##          G       0      881 881     8.45      9.25    0     1     5    13
##     G.Bkhd       0      881 881     0.81      1.36    0     0     0     1
##    G.Dflct       0      881 881     0.28      0.66    0     0     0     0
##     G.Slap       0      881 881     0.93      1.59    0     0     0     1
##     G.Snap       0      881 881     1.22      1.84    0     0     1     2
##      G.Tip       0      881 881     0.8       1.39    0     0     0     1
##     G.Wrap       0      881 881     0.062     0.26    0     0     0     0
##     G.Wrst       0      881 881     4.35      5.23    0     0     2     6
##         GP       0      881 881    51.88     28.45    1    24    63    78
##        GWG       0      881 881     1.32      1.84    0     0     1     2
##        Hat       0      881 881     0.093     0.34    0     0     0     0
##         Ht       0      881 881    73         2.14   64    72    73    74
##       iBLK       0      881 881    42.47     40.95    0    12    32    56
##        iCF       0      881 881   169.53    133.64    0    49   151   256
##       iGVA       0      881 881    25.87     22.71    0     7    21    38
##        iHA       0      881 881    62.16     41.43    0    25    64    90
##       iHDf       0      881 881    -0.027    42.09 -131   -23    -2    16
##        iHF       0      881 881    62.22     53.44    0    19    49    93
##      iMiss       0      881 881    35.22     28.32    0    10    31    54
##      iRush       0      881 881     4.54      4.35    0     1     4     7
##       iSCF       0      881 881    84.62     75.08    0    21    65   131
##        iSF       0      881 881    92.16     73.43    0    26    81   141
##       iTKA       0      881 881    20.99     18.43    0     5    17    32
##        NZS       0      881 881   278.09    189.01    0    97   287   424
##        OTG       0      881 881     0.22      0.58    0     0     0     0
##       Ovrl     126      755 881    68.27     62.72    1    17    47   107
##        OZS       0      881 881   296.03    233.62    0    80   255   472
##        PIM       0      881 881    24.03     22.29    0     8    20    34
##  PlusMinus       0      881 881    -0.42     10.6   -42    -6     0     4
##        PSA       0      881 881     0.059     0.25    0     0     0     0
##        PSG       0      881 881     0.022     0.15    0     0     0     0
```

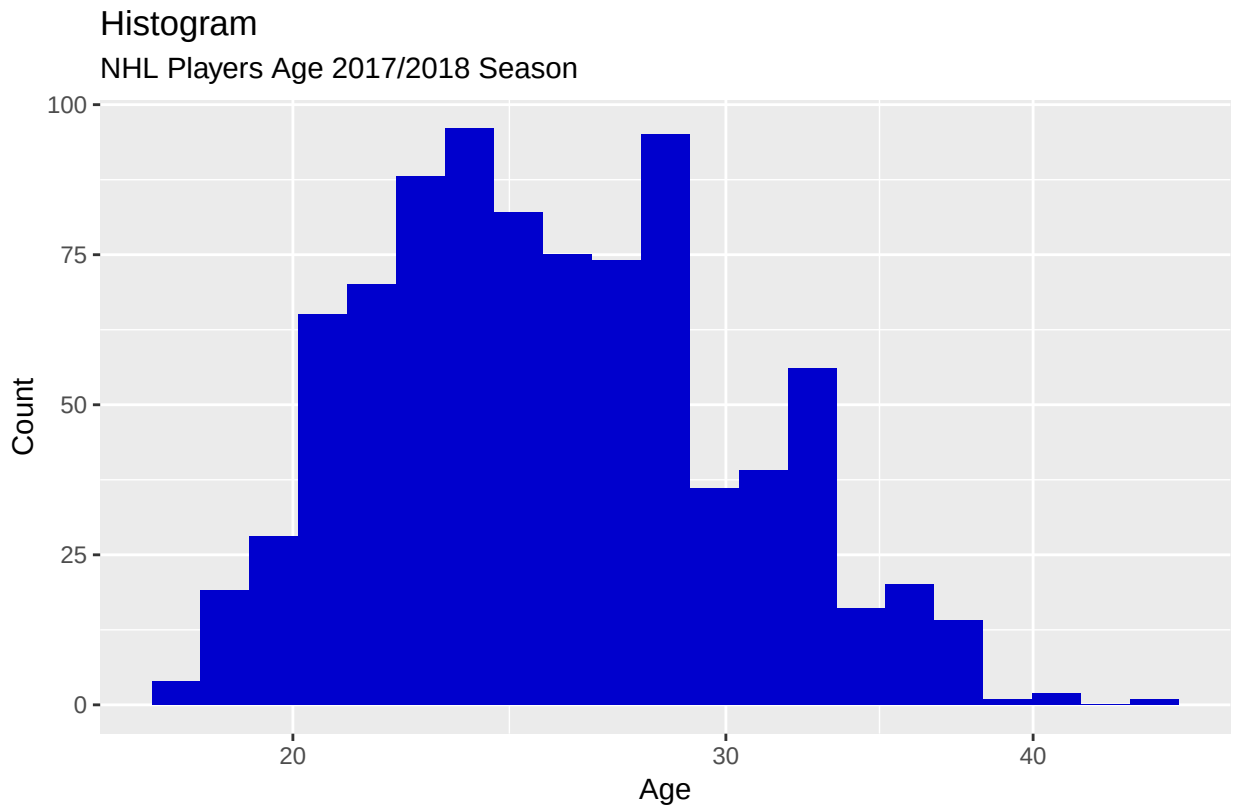
```
##        PTS       0      881 881    22.69     22       0     4    16    34
##    Seasons       0      881 881     5.73      4.17    1     2     5     8
##     Shifts       0      881 881  1113.27    700.65    5   427  1239  1702
##        TOI       0      881 881 51374.44  34022.02  205 18564 54407 79116
##         Wt       0      881 881   199.7      15.49  154   190   200   210
##    p100     hist
##      68 <U+2587><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##      45 <U+2583><U+2587><U+2587><U+2583><U+2582><U+2581><U+2581><U+2581>
##    4015 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       9 <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##     946 <U+2587><U+2585><U+2586><U+2583><U+2583><U+2582><U+2581><U+2581>
##       7 <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      11 <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      49 <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      10 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       6 <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      17 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      14 <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       9 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       3 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      28 <U+2587><U+2583><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      82 <U+2583><U+2582><U+2581><U+2581><U+2582><U+2582><U+2583><U+2587>
##      12 <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       3 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      81 <U+2581><U+2581><U+2582><U+2586><U+2587><U+2585><U+2581><U+2581>
##     223 <U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##     774 <U+2587><U+2586><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##     129 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     210 <U+2587><U+2585><U+2587><U+2586><U+2583><U+2581><U+2581><U+2581>
##     196 <U+2581><U+2582><U+2585><U+2587><U+2582><U+2581><U+2581><U+2581>
##     278 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     171 <U+2587><U+2585><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##      24 <U+2587><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##     352 <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     355 <U+2587><U+2585><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581>
##     111 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     751 <U+2587><U+2583><U+2585><U+2586><U+2586><U+2583><U+2582><U+2581>
##       5 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##     291 <U+2587><U+2583><U+2582><U+2582><U+2582><U+2581><U+2581><U+2581>
##     992 <U+2587><U+2585><U+2583><U+2583><U+2583><U+2582><U+2581><U+2581>
##     212 <U+2587><U+2583><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##      49 <U+2581><U+2581><U+2582><U+2587><U+2583><U+2581><U+2581><U+2581>
##       2 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##       2 <U+2587><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
##     108 <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##      24 <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##    2511 <U+2587><U+2583><U+2583><U+2583><U+2586><U+2587><U+2583><U+2582>
##  132031 <U+2587><U+2583><U+2583><U+2585><U+2586><U+2585><U+2582><U+2581>
##     260 <U+2581><U+2582><U+2586><U+2587><U+2586><U+2582><U+2581><U+2581>
##
## -- Variable type:numeric ------------------------------------------------------------------------
##    variable missing complete   n       mean         sd        p0       p25
##          GC       0      881 881       8.48       8.28      0         1.7
##        Pass       0      881 881     151.19     140.12      0        33.9
```

```
##         PAX       0      881 881       1.74      10.5     -47.8      -2.7
##     PTSvsGP       0      881 881       0.36       0.27      0         0.18
##     Salary1       0      881 881 2442916.58 2441001.07 650000    742500
##       sDist       0      881 881      36.41      13.44      0        27.25
##        TOff       0      881 881     320.73     261.88      0        84.9
##  TOIpercent       0      881 881      25.27       6.98      6.25     20.21
##     TOIvsGP       0      881 881      15.12       4.31      3.42     11.97
##        p50      p75       p100     hist
##       6       12.9    40.6     <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##     124.7    231.9   686.1     <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##       0        4.9    85       <U+2581><U+2581><U+2587><U+2583><U+2581><U+2581><U+2581><U+2581>
##       0.3      0.5     1.5     <U+2586><U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##  925000    4e+06       1.4e+07 <U+2587><U+2581><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      32.38    48.31   78.27    <U+2581><U+2581><U+2587><U+2587><U+2583><U+2585><U+2581><U+2581>
##     286.5    491.9  1435.3     <U+2587><U+2585><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581>
##      25.24    29.89   44.18    <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
##      15.08    18.02   26.84    <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
```

Skimr is a fantastic package that allows to you summarize data quickly and understand many elements from a single command.

Understanding what the data is saying

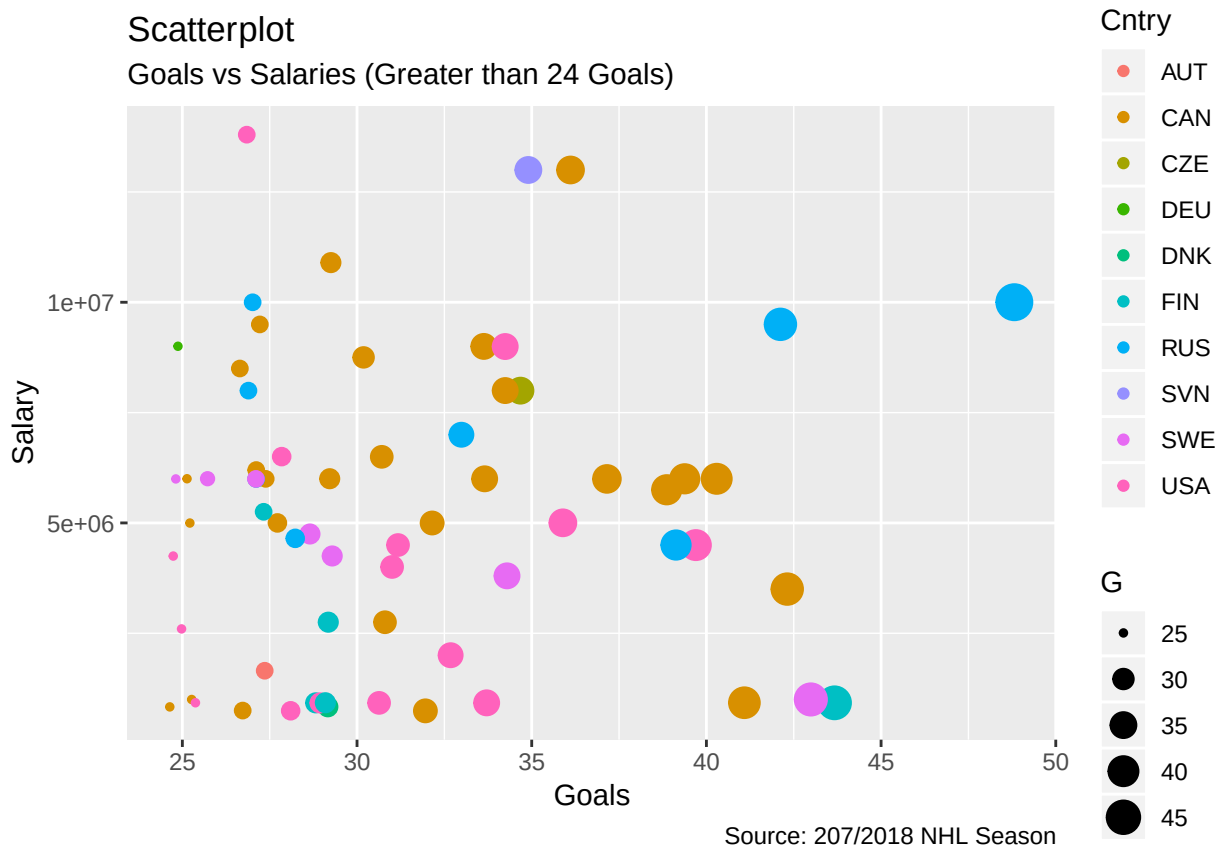
```
# Example of normaized histogram using log10
ggplot(data, aes(x = Age)) +
  geom_histogram(bins = 21, fill="blue3")+
  scale_x_log10()+
  labs(subtitle = "NHL Players Age 2017/2018 Season",
       caption="Source: 207/2018 NHL Season",
       y = "Count",
       x = "Age",
       title = "Histogram")
```



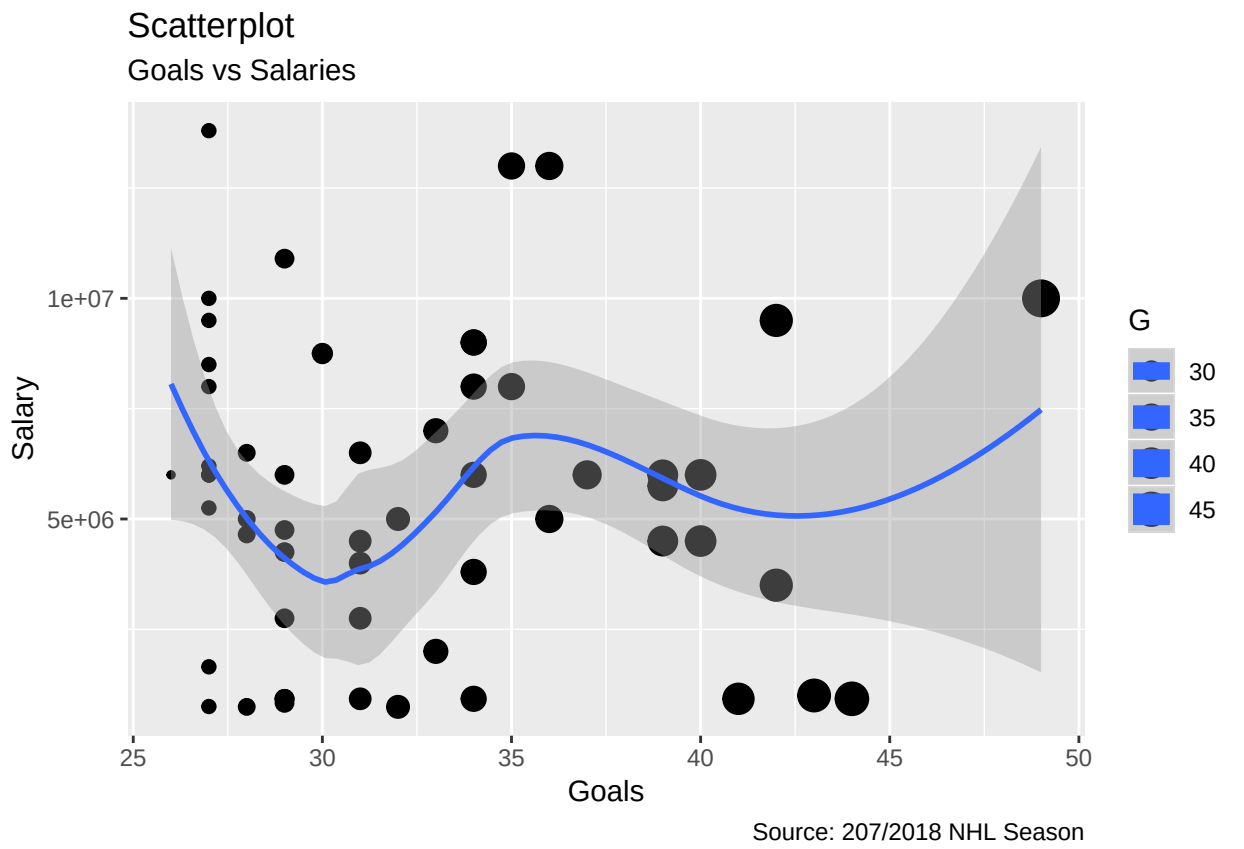
Hockey used to be a blend of experience and youth, however you can see by the distrubution above that the NHL is highly dependant on its youthful players. This contributes to the increased speed of the game.

```
# view players based on their Country
goals_scored <-
  data %>%
  filter(G > 24)%>%
  arrange(desc(G))
```

```
# plot salary from country with goals
ggplot(goals_scored, aes(x = G, y = Salary1, size = G, color = Cntry)) +
  geom_jitter()+
  labs(subtitle = "Goals vs Salaries (Greater than 24 Goals)",
       caption="Source: 207/2018 NHL Season",
       y = "Salary",
       x = "Goals",
       title = "Scatterplot")
```

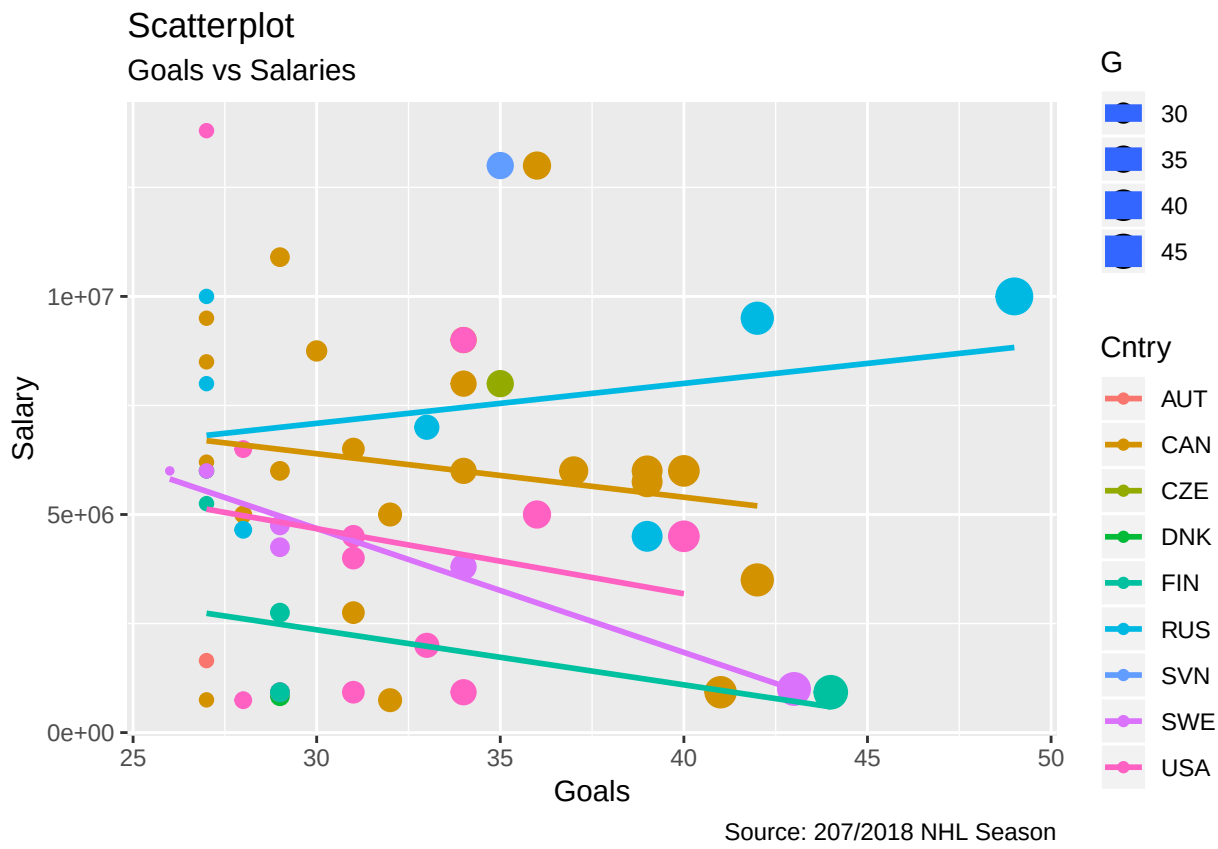



```r
# create a new variable that holds rows with goals > 25
salary_country <- data %>%
  filter(G > 25)

# plot salary from country with goals
ggplot(salary_country, aes(x = G, y = Salary1, size = G)) +
  geom_point()+
  geom_smooth(method = "loess", formula = "y ~ x")+
  labs(subtitle = "Goals vs Salaries",
       caption="Source: 207/2018 NHL Season",
       y = "Salary",
       x = "Goals",
       title = "Scatterplot")
```

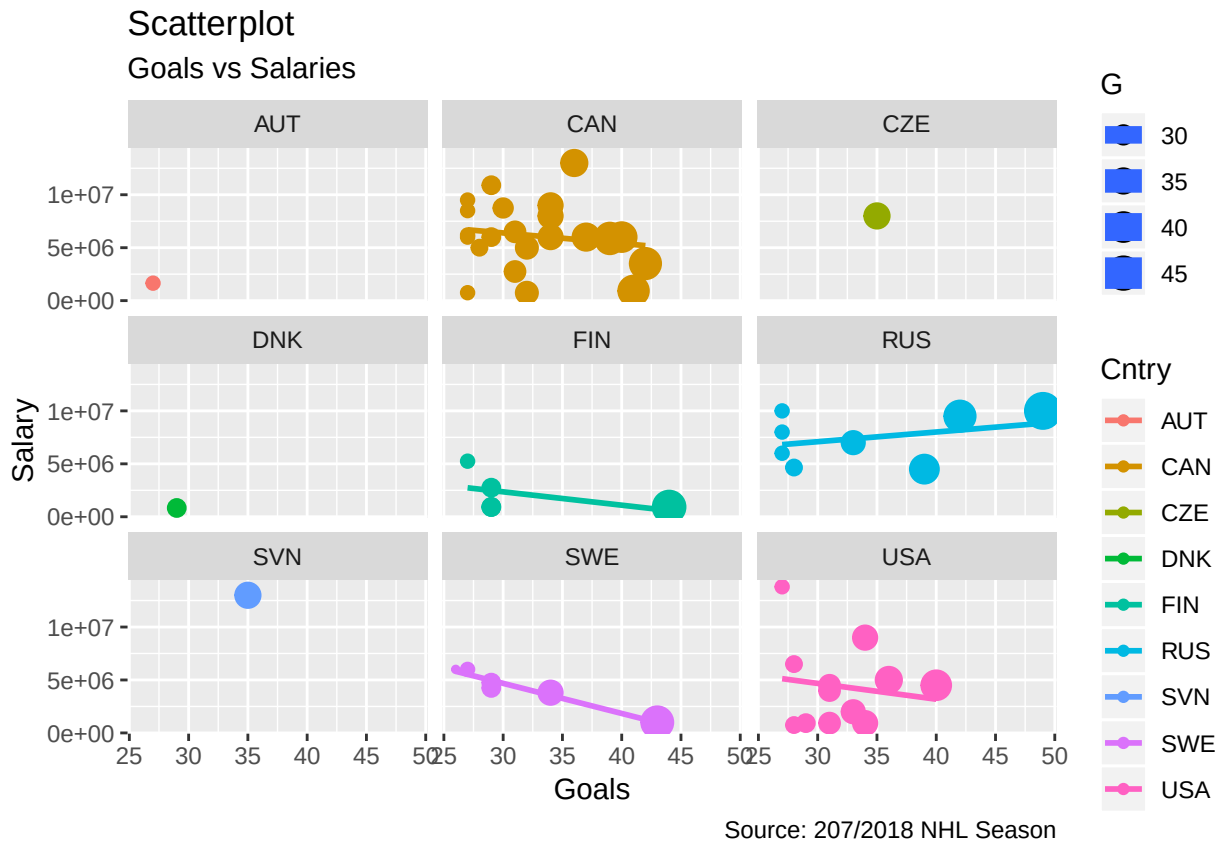


Use span to control the "wiggliness" of the default loess smoother. The span is the fraction of points used to fit each local regression: Small numbers make a wigglier curve, larger numbers make a smoother curve.

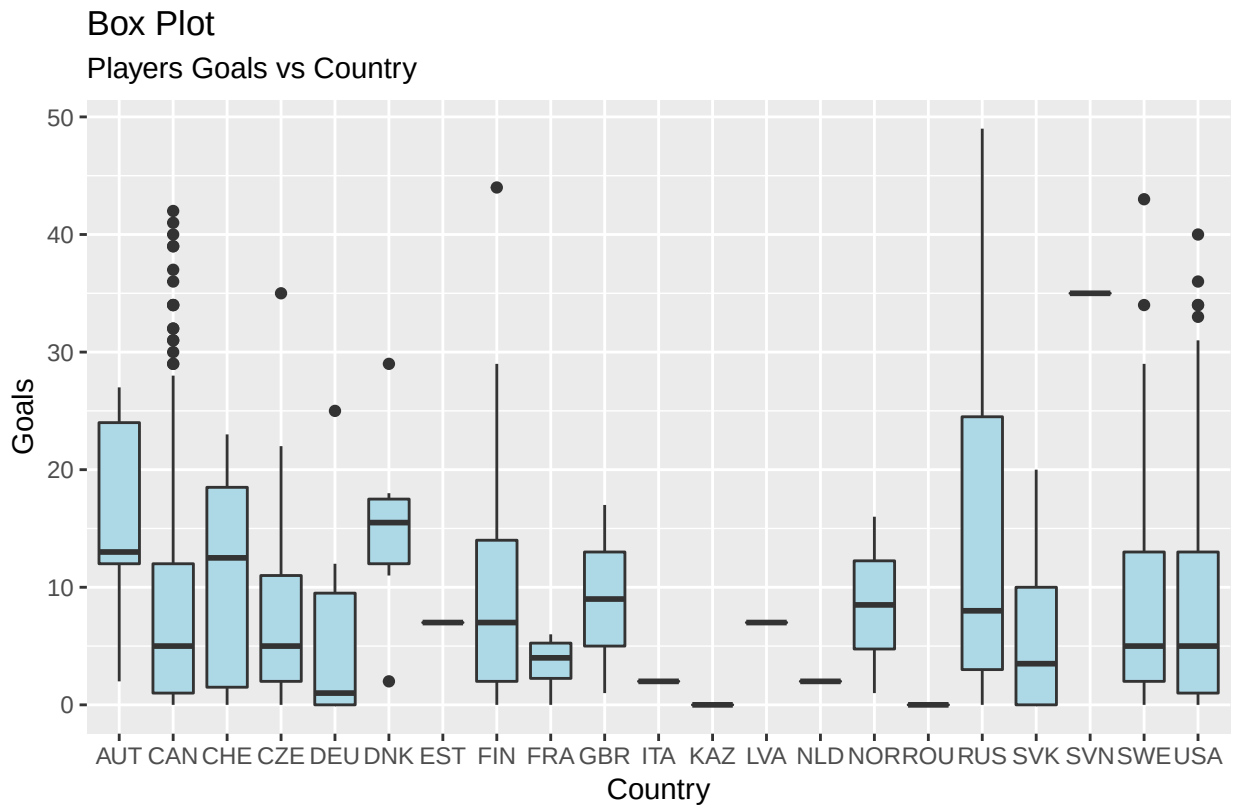
```
# plot salary from country with goals
ggplot(salary_country, aes(x = G, y = Salary1, size = G, color = Cntry)) +
  geom_point()+
  geom_smooth(se = FALSE, method = lm)+
  labs(subtitle = "Goals vs Salaries",
       caption="Source: 207/2018 NHL Season",
       y = "Salary",
       x = "Goals",
       title = "Scatterplot")
```




```
# plot salary from country with goals
ggplot(salary_country, aes(x = G, y = Salary1, size = G, color = Cntry)) +
  geom_point()+
  geom_smooth(se = FALSE, method = lm)+
  facet_wrap(~Cntry) +
  labs(subtitle = "Goals vs Salaries",
       caption="Source: 207/2018 NHL Season",
       y = "Salary",
       x = "Goals",
       title = "Scatterplot")
```




```
# create box plots to see the range of goals per country
ggplot(data, aes(x = Cntry, y = G))+
  geom_boxplot(fill="lightblue")+
      labs(subtitle = "Players Goals vs Country",
      caption="Source: 207/2018 NHL Season",
       y = "Goals",
       x = "Country",
       title = "Box Plot")
```



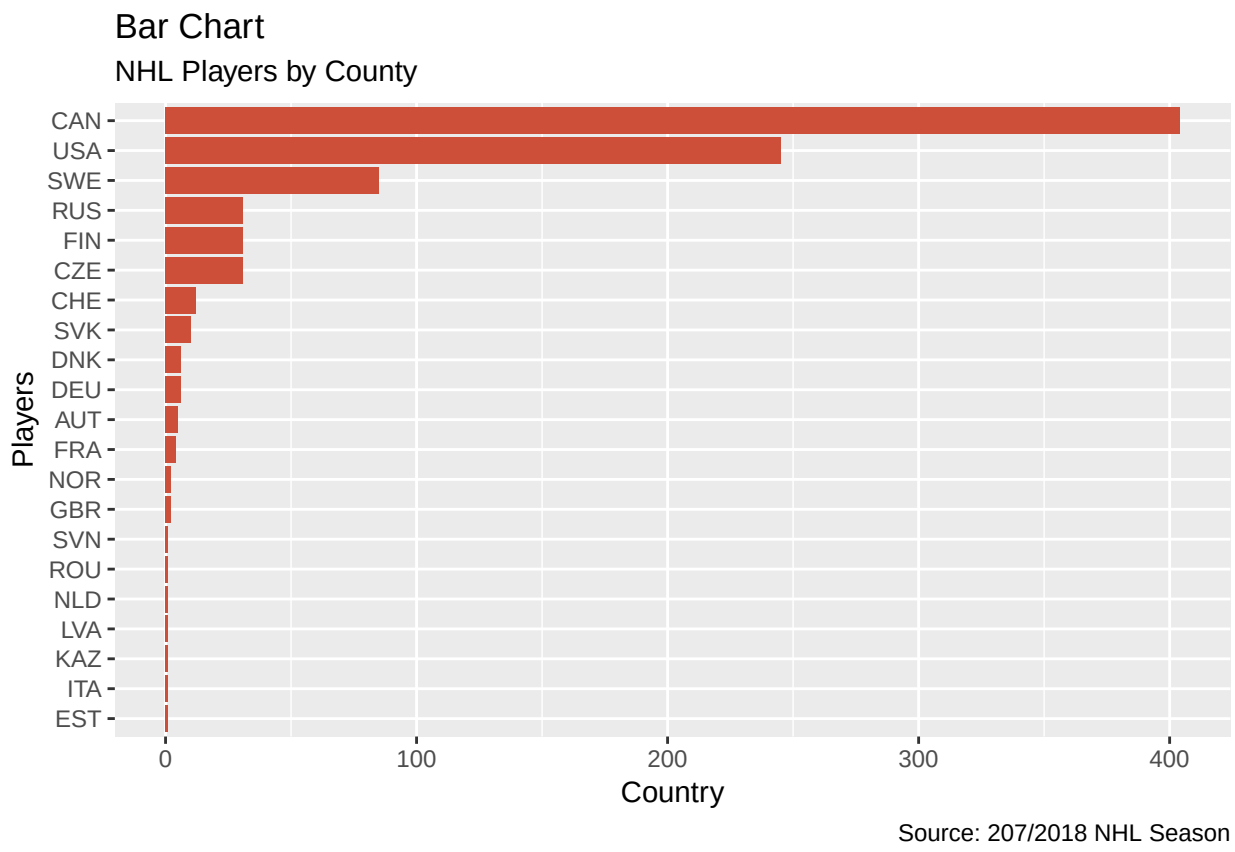
Source: 207/2018 NHL Season

```
# distribution of players and their country in the NHL
country_count <- as.data.frame(data %>%
  count(Cntry) %>%
  arrange(desc(n)))

# print results
country_count
```

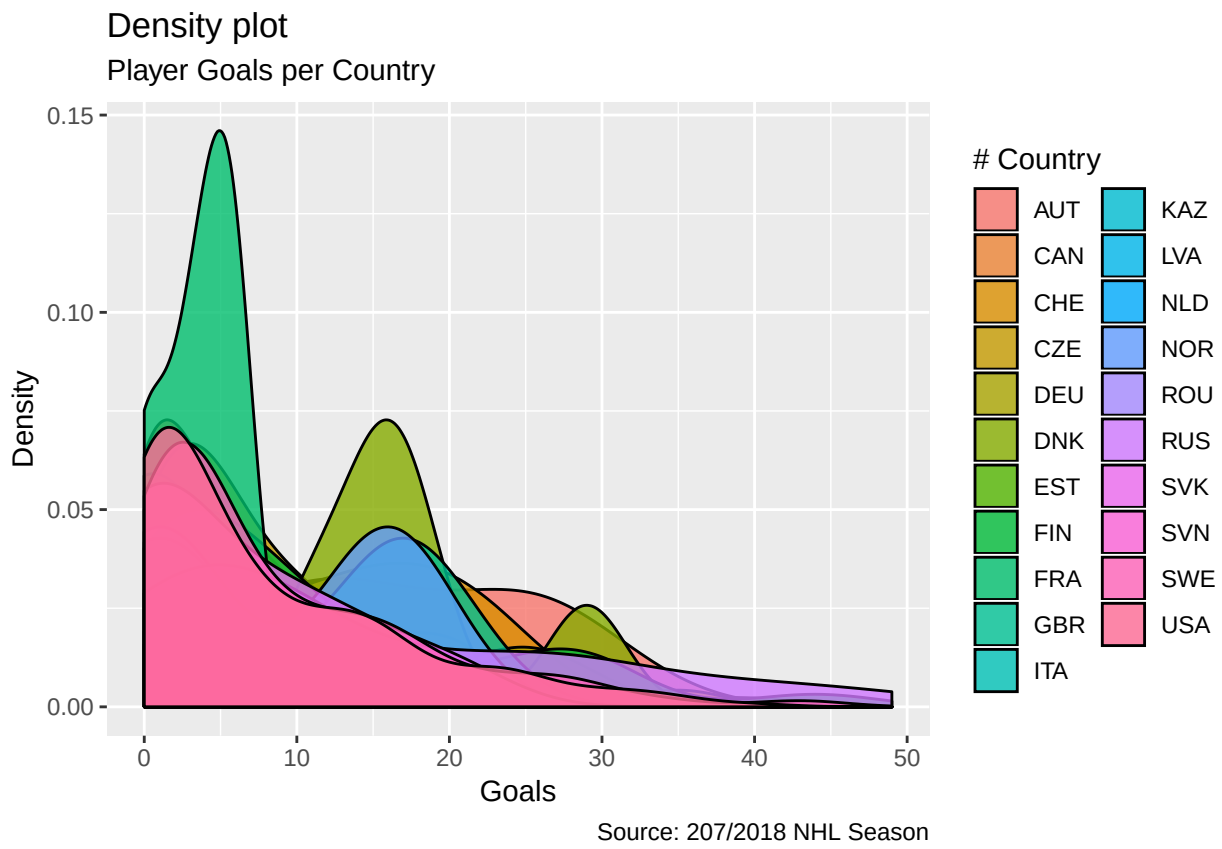
```
##    Cntry   n
## 1    CAN 404
## 2    USA 245
## 3    SWE  85
## 4    CZE  31
## 5    FIN  31
## 6    RUS  31
## 7    CHE  12
## 8    SVK  10
## 9    DEU   6
## 10   DNK   6
## 11   AUT   5
## 12   FRA   4
## 13   GBR   2
## 14   NOR   2
## 15   EST   1
## 16   ITA   1
## 17   KAZ   1
```

```
## 18   LVA   1
## 19   NLD   1
## 20   ROU   1
## 21   SVN   1
```

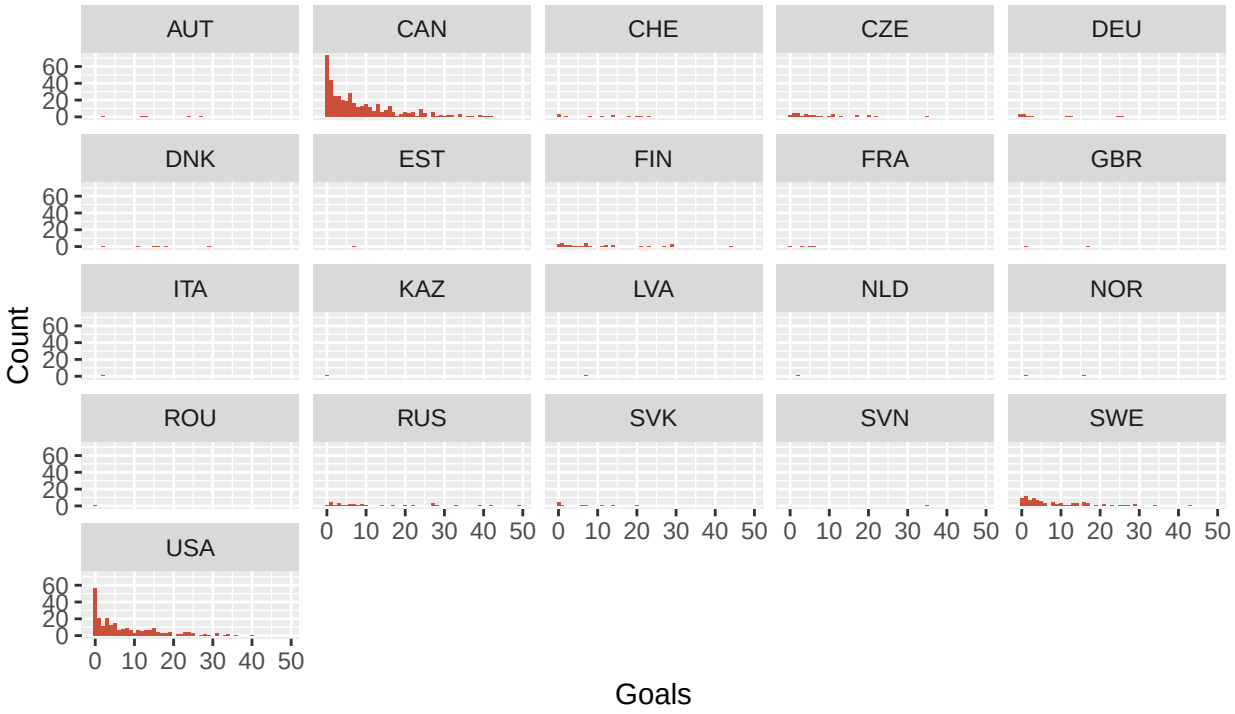
```
# plotting the results from country_count
ggplot(country_count, aes(x= reorder(Cntry, n), y = n)) +
  geom_bar(stat="identity", fill = "tomato3") +
  coord_flip()+
    labs(subtitle = "NHL Players by County",
      caption="Source: 207/2018 NHL Season",
       y = "Country",
       x = "Players",
       title = "Bar Chart")
```



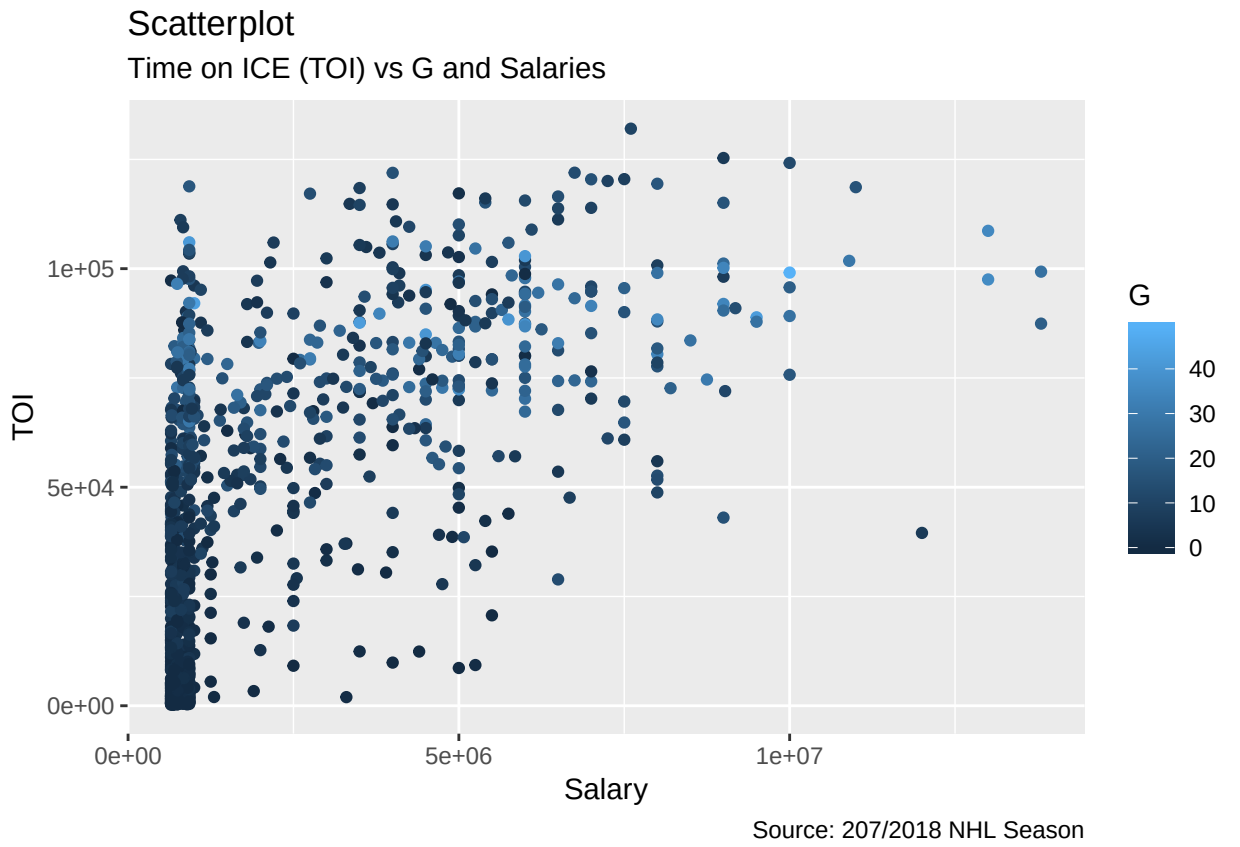

```
ggplot(data, aes(G))+
  geom_density(aes(fill=factor(Cntry)), alpha = .8)+
  labs(title="Density plot",
    subtitle="Player Goals per Country",
    caption="Source: 207/2018 NHL Season",
    x="Goals",
    y = "Density",
    fill="# Country")
```




```
# Goals per Country
ggplot(data, aes(G)) +
  geom_bar(fill="tomato3")+
  facet_wrap(~Cntry)+
    labs(subtitle = "Number of Goals per County",
       caption="Source: 207/2018 NHL Season",
       y = "Count",
       x = "Goals",
       title = "Facet Wrap")
```

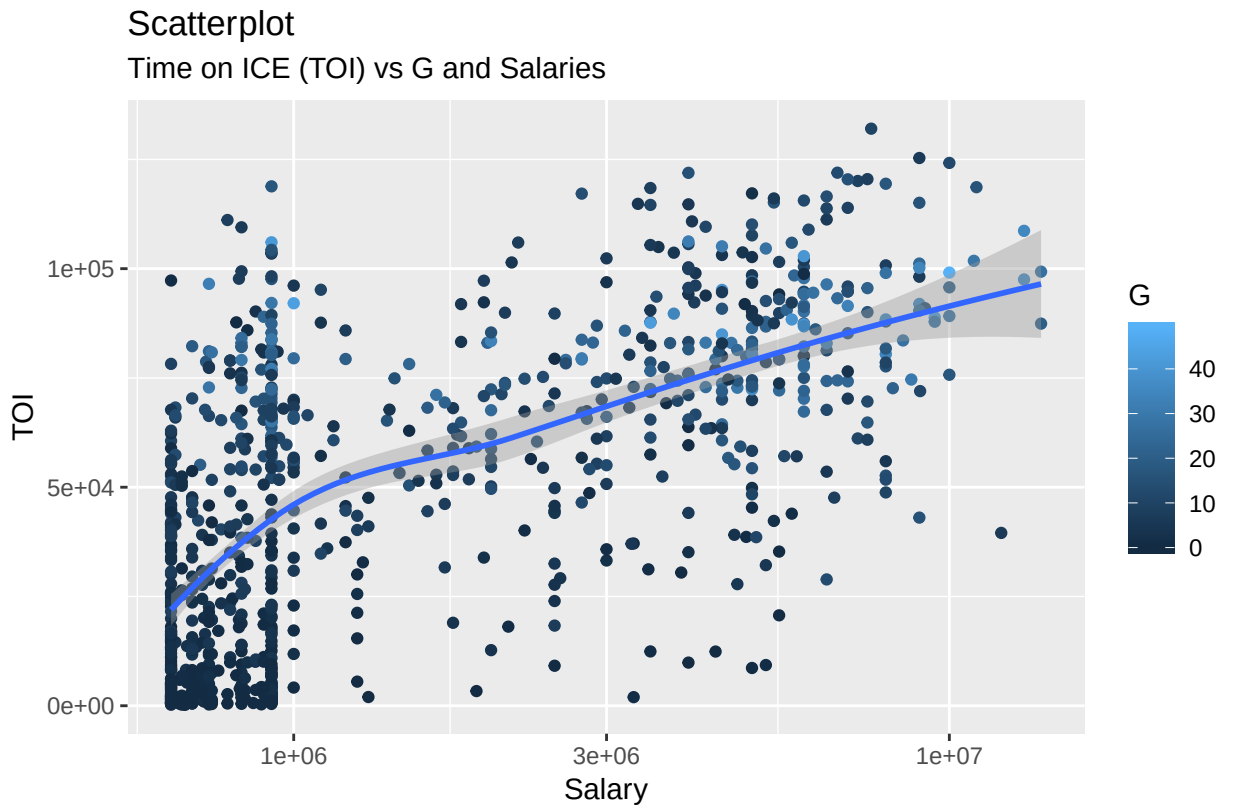



```
#time on ice, goals, salary
ggplot(data, aes(x = Salary1, y = TOI, color = G))+
  geom_point()+
    labs(subtitle = "Time on ICE (TOI) vs G and Salaries",
       caption="Source: 207/2018 NHL Season",
       y = "TOI",
       x = "Salary",
       title = "Scatterplot")
```

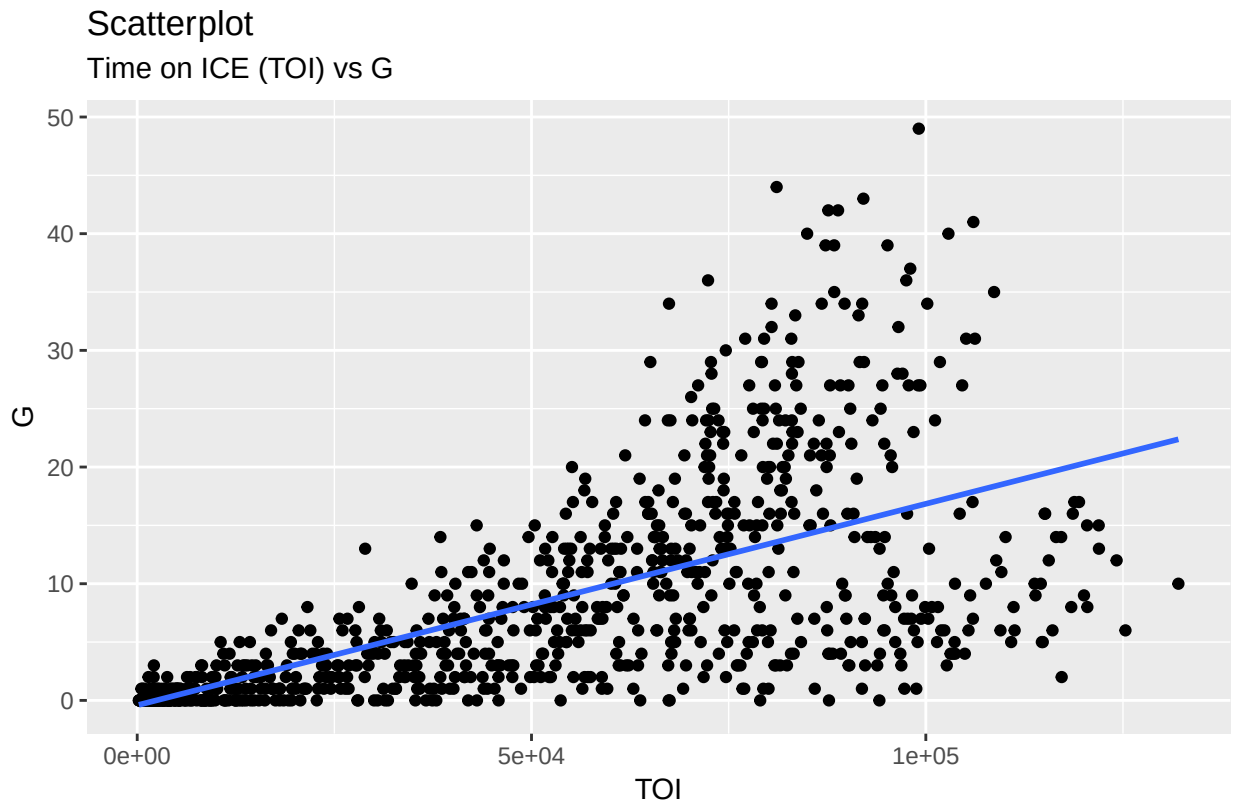


Time on ICE maybe directly related to salary, but maybe have a correlation when looking at all the skaters. This is because there are shutdown defensive players that clock significant hours and minumilaly contribute to offensive totals.

```
#time on ice, goals, salary
ggplot(data = data) +
  geom_point(mapping = aes(x= Salary1, y = TOI, color = G)) +
  geom_smooth(mapping = aes(x = Salary1, y = TOI)) +
  scale_x_log10()+
      labs(subtitle = "Time on ICE (TOI) vs G and Salaries",
       caption="Source: 207/2018 NHL Season",
       y = "TOI",
       x = "Salary",
       title = "Scatterplot")
```




```
# Time on ice vs G
ggplot(data, aes(x = TOI, y = G))+
  geom_point()+
  geom_smooth(method = "lm", se=F)+
    labs(subtitle = "Time on ICE (TOI) vs G",
       caption="Source: 207/2018 NHL Season",
       y = "G",
       x = "TOI",
       title = "Scatterplot")
```



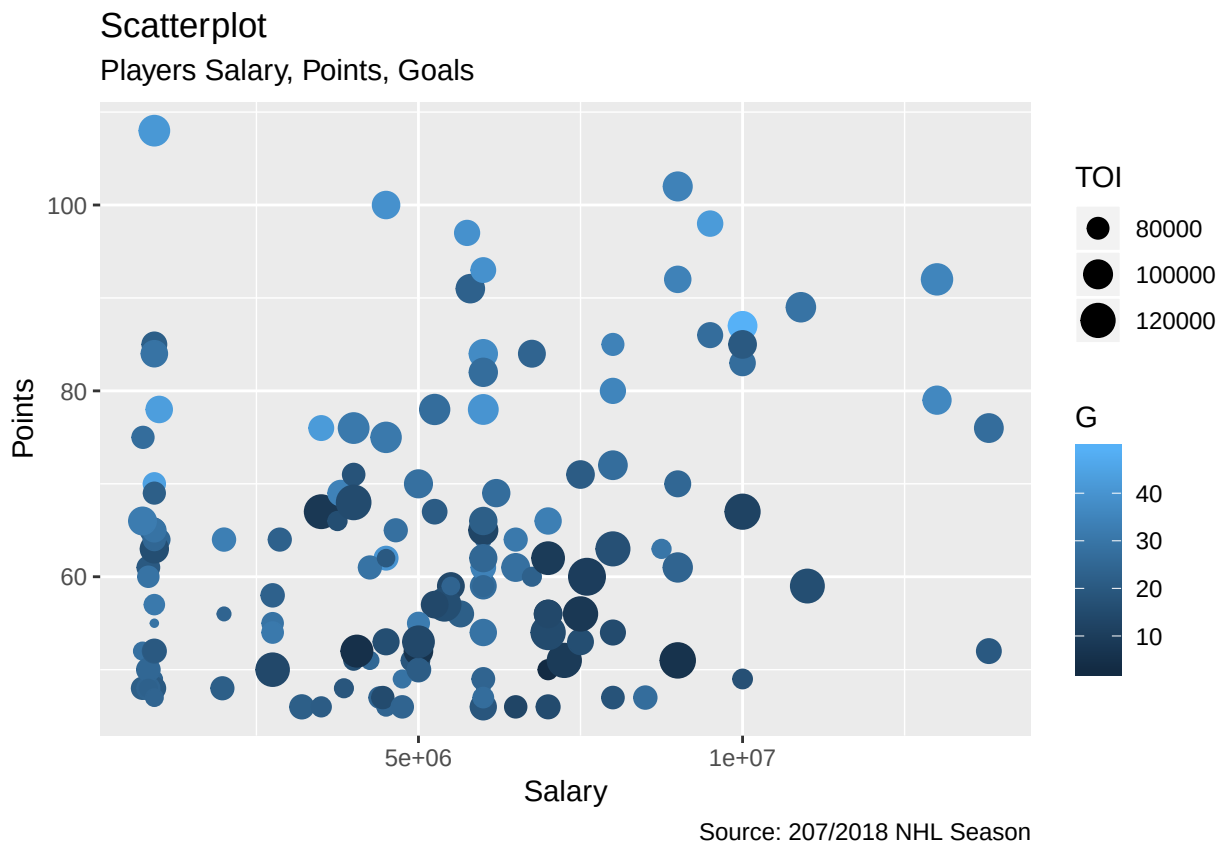

```
greater_45_PTS = data %>%
  filter(PTS > 45)
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
ggplot(greater_45_PTS, aes(x = Salary1, y = PTS, color = G, size = TOI))+
  geom_point()+

  labs(subtitle = "Players Salary, Points, Goals",
    caption="Source: 207/2018 NHL Season",
    y = "Points",
    x = "Salary",
    title = "Scatterplot")
```



Removing redundant variables

```
cleaned_hockey <- data %>%
  select(-Ht,
         -Wt,
         -Nat,
         -DftRd,
         -Ovrl,
         -Debut,
         -Seasons,
         -Born,
         -G.Bkhd,
         -G.Dflct,
         -G.Slap,
         -G.Tip,
         -G.Wrap,
         -G.Wrst,
         -G.Snap,
         -City,
         -FirstName,
         -LastName,
         -DZS,
         -iCF,
         -IPPper,
         -Cntry,
```

```
         -sDist,
         -PAX,
         -PIM,
         -BLKpercent,
         -FirstGoal,
         -GWG,
         -ENG,
         -PSA,
         -PSG,
         -iHDf,
         -PlusMinus,
         -OTG,
         -Hat,
         -iHF,
         -iHA,
         -iBLK)%>%
  mutate(Salary = Salary1)%>%
  select(-Salary1)

# Viewing cleaning data
glimpse(cleaned_hockey)
```

```
## Observations: 881
## Variables: 25
## $ Hand       <fct> R, L, R, L, L, L, L, L, L, L, R, R, L, L, R, L, L, ...
## $ Age        <int> 28, 30, 27, 35, 36, 38, 27, 37, 22, 26, 24, 36, 37,...
## $ Pos        <fct> RW, C, D, LW, D, LW, C, D, D, LW, D, RW, D, D, RW, ...
## $ GP         <int> 33, 62, 82, 62, 80, 82, 81, 81, 1, 7, 24, 82, 79, 1...
## $ G          <int> 3, 3, 3, 13, 4, 13, 24, 0, 0, 0, 1, 16, 4, 0, 4, 40...
## $ A          <int> 6, 5, 7, 11, 19, 16, 26, 10, 1, 2, 1, 35, 23, 5, 4,...
## $ PTS        <int> 9, 8, 10, 24, 23, 29, 50, 10, 1, 2, 2, 51, 27, 5, 8...
## $ PTSvsGP    <dbl> 0.27, 0.13, 0.12, 0.39, 0.29, 0.35, 0.62, 0.12, 1.0...
## $ GC         <dbl> 3.4, 2.9, 3.5, 9.9, 7.8, 11.2, 19.9, 2.9, 0.3, 0.6,...
## $ Shifts     <int> 491, 952, 1759, 1099, 2215, 1383, 1844, 1797, 14, 8...
## $ TOI        <int> 23617, 39114, 82396, 44707, 104951, 58776, 86436, 9...
## $ TOIvsGP    <dbl> 11.93, 10.51, 16.75, 12.02, 21.86, 11.95, 17.79, 19...
## $ TOIpercent <dbl> 20.18, 17.88, 28.08, 20.52, 36.52, 20.19, 29.24, 32...
## $ iSF        <int> 37, 45, 98, 82, 73, 81, 151, 54, 0, 11, 27, 198, 76...
## $ iSCF       <int> 42, 41, 57, 79, 53, 78, 183, 33, 0, 15, 12, 183, 49...
## $ iRush      <int> 0, 1, 1, 5, 5, 5, 12, 0, 0, 1, 2, 10, 5, 1, 0, 7, 7...
## $ Pass       <dbl> 71.5, 38.9, 113.3, 181.6, 148.8, 184.0, 245.2, 56.2...
## $ TOff       <dbl> 135.5, 112.9, 364.3, 314.6, 338.8, 317.0, 509.2, 18...
## $ OZS        <int> 134, 165, 381, 290, 429, 268, 495, 384, 4, 12, 112,...
## $ NZS        <int> 109, 178, 447, 242, 671, 398, 395, 500, 1, 20, 91, ...
## $ iMiss      <int> 10, 10, 61, 25, 43, 24, 61, 21, 0, 5, 19, 76, 21, 6...
## $ iGVA       <int> 8, 13, 31, 25, 85, 29, 29, 44, 0, 2, 14, 85, 52, 7,...
## $ iTKA       <int> 15, 14, 15, 14, 18, 38, 40, 17, 0, 3, 5, 46, 14, 4,...
## $ Status     <fct> UFA, UFA, UFA, UFA, UFA, UFA, UFA, UFA, RFA, UFA, R...
## $ Salary     <dbl> 700000, 4700000, 3500000, 1000000, 3600000, 2000000...
```

```
# transforming salaries into units of a million
cleaned_hockey <- cleaned_hockey %>%
  mutate(Salary = Salary/1000000)
```

Grouping Salaries

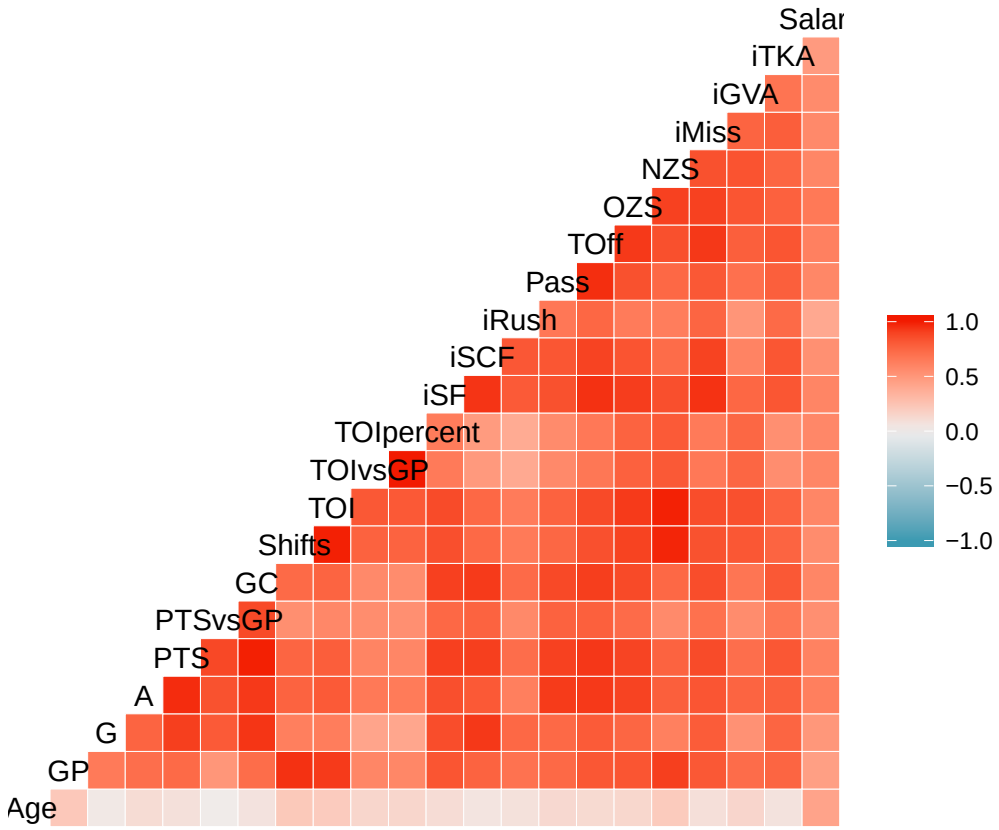
```
# creating salary ranges for cleaner visualization
cleaned_hockey$grp_Salary[cleaned_hockey$Salary < .99] <- .75
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 1, 1.99) ] <- 1.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 2, 2.99) ] <- 2.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 3, 3.99) ] <- 3.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 4, 4.99) ] <- 4.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 5, 5.99) ] <- 5.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 6, 6.99) ] <- 6.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 7, 7.99) ] <- 7.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 8, 8.99) ] <- 8.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 9, 9.99) ] <- 9.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 10, 10.99) ] <- 10.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 11, 11.99) ] <- 11.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 12, 12.99) ] <- 12.5
cleaned_hockey$grp_Salary[between(cleaned_hockey$Salary, 13, 13.99) ] <- 13.5

# viewing the upated dataset
skim(cleaned_hockey)
```

```
## Skim summary statistics
##  n obs: 881
##  n variables: 26
##
## -- Variable type:factor ---------------------------------------------------------------------
##  variable missing complete   n n_unique                      top_counts
##      Hand       0      881 881        2           L: 551, R: 330, NA: 0
##       Pos       0      881 881        8 D: 303, C: 232, LW: 171, RW: 137
##    Status       0      881 881        2       UFA: 500, RFA: 381, NA: 0
##  ordered
##    FALSE
##    FALSE
##    FALSE
##
## -- Variable type:integer --------------------------------------------------------------------
##  variable missing complete   n     mean       sd  p0   p25   p50   p75
##         A       0      881 881    14.24    14.14   0     3    10    21
##       Age       0      881 881    26.08     4.31  18    23    25    29
##         G       0      881 881     8.45     9.25   0     1     5    13
##        GP       0      881 881    51.88    28.45   1    24    63    78
##      iGVA       0      881 881    25.87    22.71   0     7    21    38
##     iMiss       0      881 881    35.22    28.32   0    10    31    54
##     iRush       0      881 881     4.54     4.35   0     1     4     7
##      iSCF       0      881 881    84.62    75.08   0    21    65   131
##       iSF       0      881 881    92.16    73.43   0    26    81   141
##      iTKA       0      881 881    20.99    18.43   0     5    17    32
##       NZS       0      881 881   278.09   189.01   0    97   287   424
##       OZS       0      881 881   296.03   233.62   0    80   255   472
##       PTS       0      881 881    22.69    22      0     4    16    34
##    Shifts       0      881 881  1113.27   700.65   5   427  1239  1702
##       TOI       0      881 881 51374.44 34022.02 205 18564 54407 79116
##    p100     hist
##      68 <U+2587><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
```

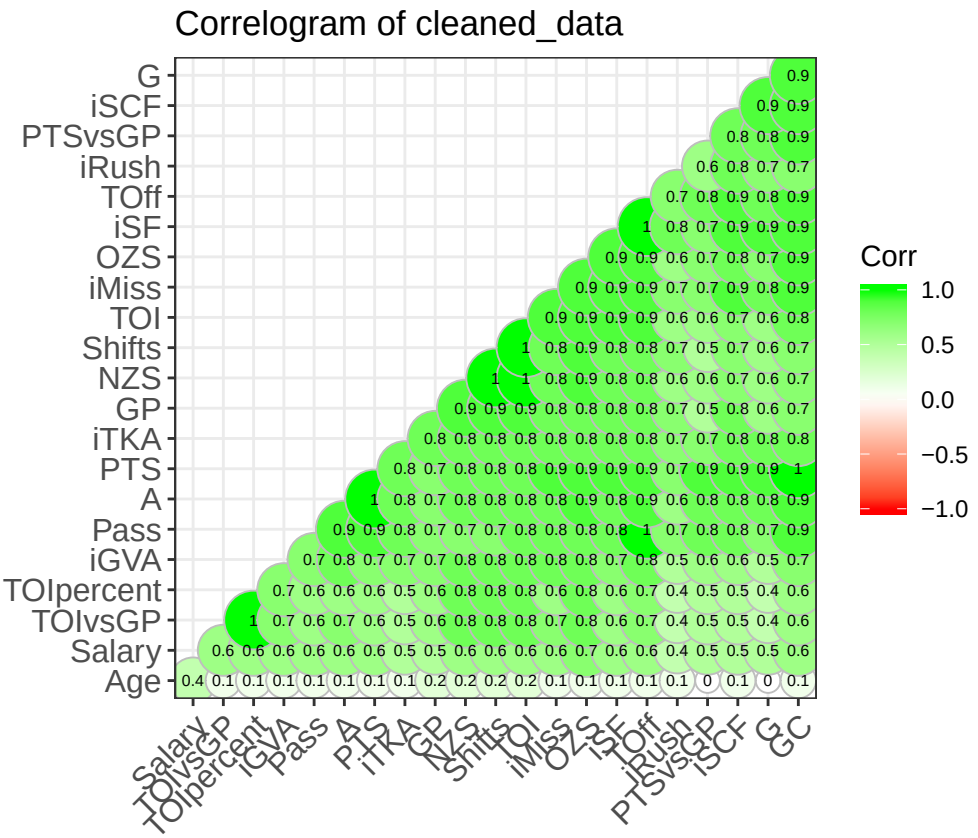
```
##      45 <U+2583><U+2587><U+2587><U+2583><U+2582><U+2581><U+2581><U+2581>
##      49 <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      82 <U+2583><U+2582><U+2581><U+2581><U+2582><U+2582><U+2583><U+2587>
##     129 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     171 <U+2587><U+2585><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##      24 <U+2587><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##     352 <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     355 <U+2587><U+2585><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581>
##     111 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     751 <U+2587><U+2583><U+2585><U+2586><U+2586><U+2583><U+2582><U+2581>
##     992 <U+2587><U+2585><U+2583><U+2583><U+2583><U+2582><U+2581><U+2581>
##     108 <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##    2511 <U+2587><U+2583><U+2583><U+2583><U+2586><U+2587><U+2583><U+2582>
##  132031 <U+2587><U+2583><U+2583><U+2585><U+2586><U+2585><U+2582><U+2581>
##
## -- Variable type:numeric -------------------------------------------------------------------------
##    variable missing complete   n   mean     sd   p0   p25    p50    p75
##          GC       0      881 881   8.48   8.28 0     1.7    6     12.9
##  grp_Salary       0      881 881   2.51   2.55 0.75  0.75   0.75   4.5
##        Pass       0      881 881 151.19 140.12 0    33.9  124.7  231.9
##     PTSvsGP       0      881 881   0.36   0.27 0     0.18   0.3    0.5
##      Salary       0      881 881   2.44   2.44 0.65  0.74   0.92   4
##        TOff       0      881 881 320.73 261.88 0    84.9  286.5  491.9
##  TOIpercent       0      881 881  25.27   6.98 6.25 20.21  25.24  29.89
##     TOIvsGP       0      881 881  15.12   4.31 3.42 11.97  15.08  18.02
##     p100     hist
##    40.6  <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##    13.5  <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##   686.1  <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     1.5  <U+2586><U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##    13.8  <U+2587><U+2581><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##  1435.3  <U+2587><U+2585><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581>
##    44.18 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
##    26.84 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
```

Correlation

```
# removing 3 variables for correlation
corr_cleaned_hockey <- cleaned_hockey %>%
  select(-Hand, -Status, -Pos, -grp_Salary)

# library for creating a corr plot

#
ggcorr(corr_cleaned_hockey)
```




```
# loading library for plotting a correlation matrix
library(ggcorrplot)

# creating correlation matrix - Another view
corr <- round(cor(corr_cleaned_hockey),1)

# plotting a Correlogram
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 2,
           method = "circle",
           colors = c("red", "white", "green"),
           title = "Correlogram of cleaned_data",
           ggtheme = theme_bw())
```



Random Forest

```
# loading random forest package
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(caret)

# setting seed so I am abel to reproduce tests
set.seed(1025)

# creating df for random forest
rf_hockey <- cleaned_hockey
```

```
#removing grouped salaries for the first run
rf_hockey <- rf_hockey%>%
  select(-grp_Salary)
```

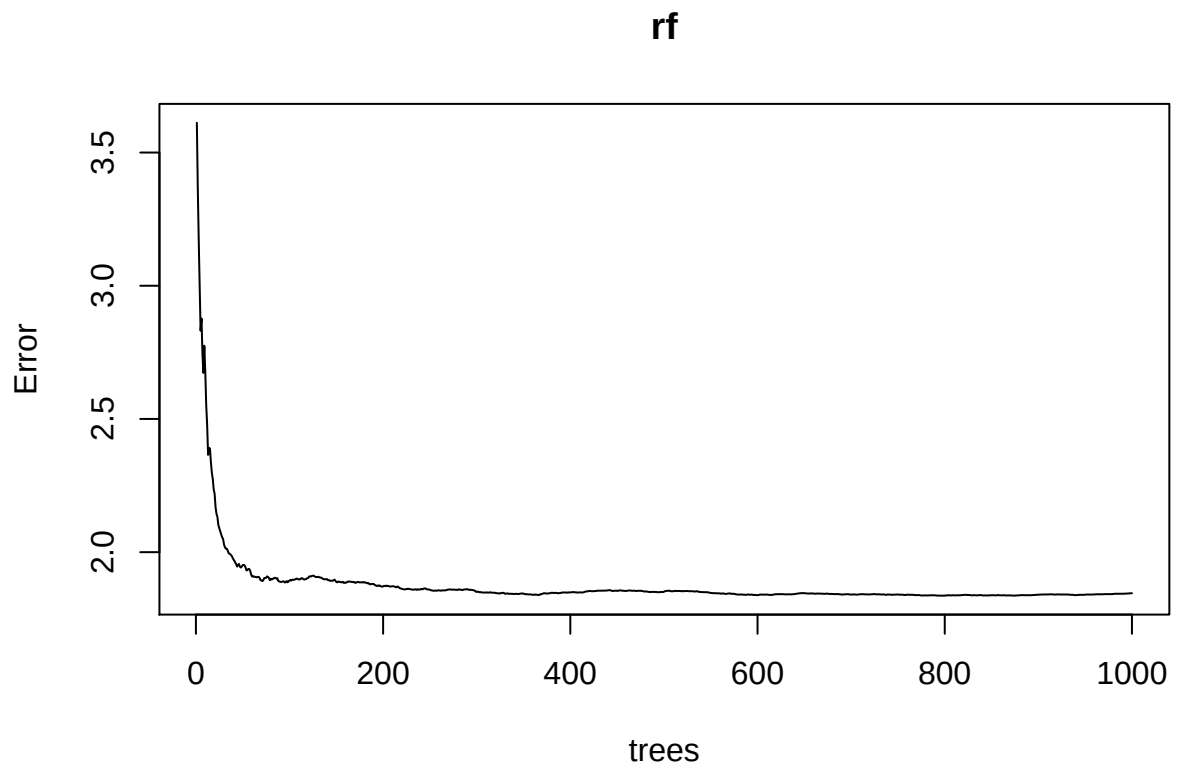
Creating Random Forest Model

```
# creating random forest with 1000 trees
rf <- randomForest(Salary~., data = rf_hockey, importance=TRUE, ntree=1000)
rf
```

```
##
## Call:
##  randomForest(formula = Salary ~ ., data = rf_hockey, importance = TRUE,      ntree = 1000)
##                Type of random forest: regression
##                      Number of trees: 1000
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 1.845816
##                     % Var explained: 68.99
```

Based on the performance about we have about ~69% success accurately predicting NHL Salaries, based off of 2017/2018 player stats.

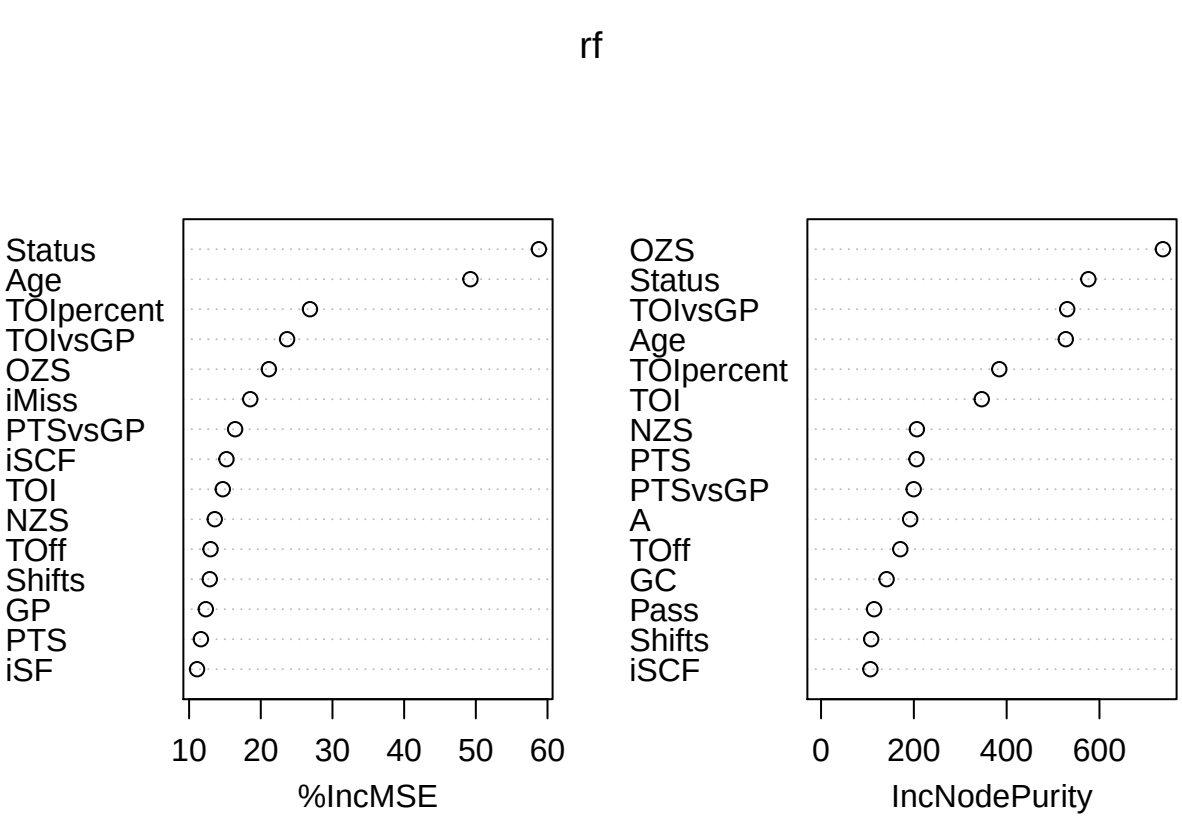
```
plot(rf)
```



```r
#View the most important variables
df_importance <- sort(importance(rf)[,1], decreasing = TRUE)
df_importance
```

```
##     Status        Age TOIpercent    TOIvsGP        OZS      iMiss
## 58.8027544 49.2478428 26.8638057 23.6710498 21.1298004 18.5316101
##    PTSvsGP       iSCF        TOI        NZS       TOff     Shifts
## 16.4225563 15.2122045 14.6982542 13.5782289 12.9920660 12.8829516
##         GP        PTS        iSF         GC          A          G
## 12.3278360 11.6448234 11.1090612 10.9451654 10.7344477 10.0606148
##        Pos       iTKA       iGVA       Pass      iRush       Hand
##  9.0134500  8.9329833  8.9326238  8.8492618  5.2155375 -0.6574002
```

```r
varImpPlot(rf, n.var = 15)
```



OZS is when a player starts in the offensive zone for a faceoff. You want to give your team the best chance to score by placing more most offensive players on the ice.

%explained variance is a measure of how well out-of-bag predictions explain the target variance of the training set. Unexplained variance would be to due true random behaviour or lack of fit. %explained variance is retrieved by randomForest:::print.randomForest as last element in rf.fit$rsq and multiplied with 100.

Second run with grouped salaries

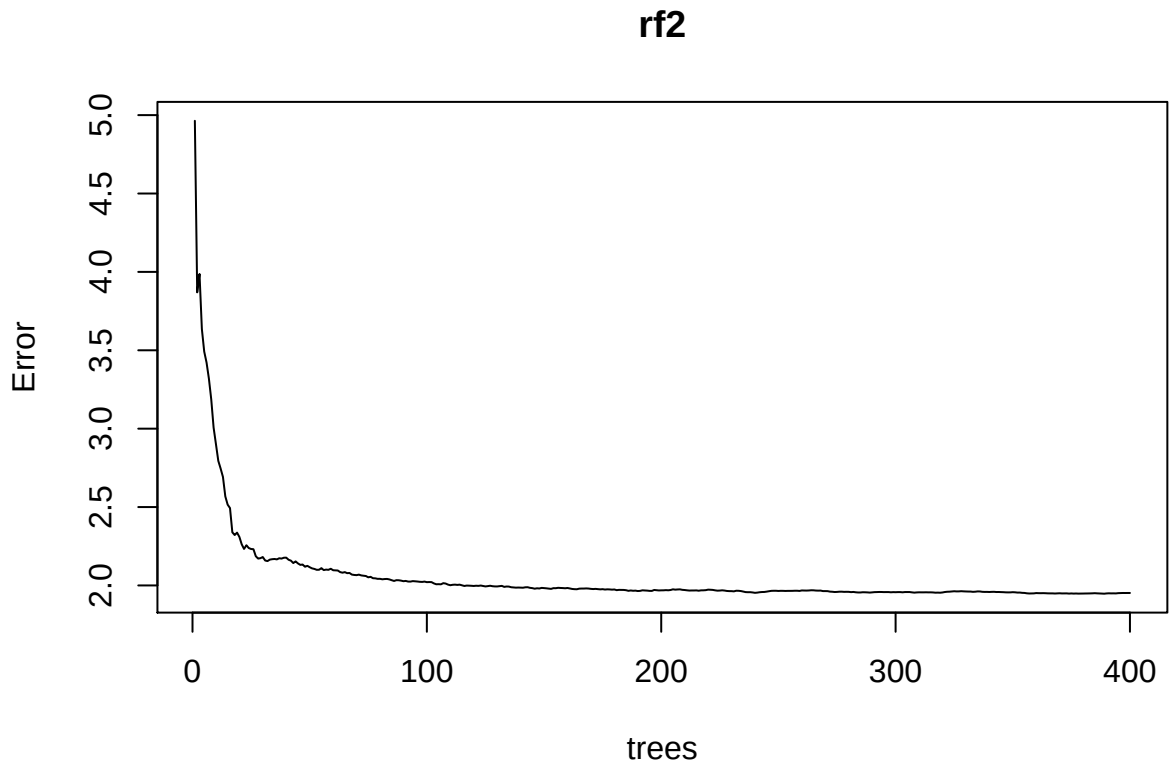
```r
#removing grouped salaries for the first run
rf_hockey2 <- cleaned_hockey%>%
  select(-Salary)
```

```
# creating 2nd model
rf2 <- randomForest(grp_Salary ~., data = rf_hockey2, importance = TRUE, ntree= 400)
rf2
```

```
##
## Call:
##  randomForest(formula = grp_Salary ~ ., data = rf_hockey2, importance = TRUE,      ntree = 400)
##                Type of random forest: regression
##                      Number of trees: 400
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 1.951812
##                     % Var explained: 69.95
```

I was able to increase the performance slightly using salary groups, but I was able able to reduce the amount of processessing the model has to complete.

```
plot(rf2)
```



Decision Tree

```
# creating a decision tree
dt_hockey <- cleaned_hockey

# viewing the data
```

```
glimpse(dt_hockey)
```

```
## Observations: 881
## Variables: 26
## $ Hand       <fct> R, L, R, L, L, L, L, L, L, L, R, R, L, L, R, L, L, ...
## $ Age        <int> 28, 30, 27, 35, 36, 38, 27, 37, 22, 26, 24, 36, 37,...
## $ Pos        <fct> RW, C, D, LW, D, LW, C, D, D, LW, D, RW, D, D, RW, ...
## $ GP         <int> 33, 62, 82, 62, 80, 82, 81, 81, 1, 7, 24, 82, 79, 1...
## $ G          <int> 3, 3, 3, 13, 4, 13, 24, 0, 0, 0, 1, 16, 4, 0, 4, 40...
## $ A          <int> 6, 5, 7, 11, 19, 16, 26, 10, 1, 2, 1, 35, 23, 5, 4,...
## $ PTS        <int> 9, 8, 10, 24, 23, 29, 50, 10, 1, 2, 2, 51, 27, 5, 8...
## $ PTSvsGP    <dbl> 0.27, 0.13, 0.12, 0.39, 0.29, 0.35, 0.62, 0.12, 1.0...
## $ GC         <dbl> 3.4, 2.9, 3.5, 9.9, 7.8, 11.2, 19.9, 2.9, 0.3, 0.6,...
## $ Shifts     <int> 491, 952, 1759, 1099, 2215, 1383, 1844, 1797, 14, 8...
## $ TOI        <int> 23617, 39114, 82396, 44707, 104951, 58776, 86436, 9...
## $ TOIvsGP    <dbl> 11.93, 10.51, 16.75, 12.02, 21.86, 11.95, 17.79, 19...
## $ TOIpercent <dbl> 20.18, 17.88, 28.08, 20.52, 36.52, 20.19, 29.24, 32...
## $ iSF        <int> 37, 45, 98, 82, 73, 81, 151, 54, 0, 11, 27, 198, 76...
## $ iSCF       <int> 42, 41, 57, 79, 53, 78, 183, 33, 0, 15, 12, 183, 49...
## $ iRush      <int> 0, 1, 1, 5, 5, 5, 12, 0, 0, 1, 2, 10, 5, 1, 0, 7, 7...
## $ Pass       <dbl> 71.5, 38.9, 113.3, 181.6, 148.8, 184.0, 245.2, 56.2...
## $ TOff       <dbl> 135.5, 112.9, 364.3, 314.6, 338.8, 317.0, 509.2, 18...
## $ OZS        <int> 134, 165, 381, 290, 429, 268, 495, 384, 4, 12, 112,...
## $ NZS        <int> 109, 178, 447, 242, 671, 398, 395, 500, 1, 20, 91, ...
## $ iMiss      <int> 10, 10, 61, 25, 43, 24, 61, 21, 0, 5, 19, 76, 21, 6...
## $ iGVA       <int> 8, 13, 31, 25, 85, 29, 29, 44, 0, 2, 14, 85, 52, 7,...
## $ iTKA       <int> 15, 14, 15, 14, 18, 38, 40, 17, 0, 3, 5, 46, 14, 4,...
## $ Status     <fct> UFA, UFA, UFA, UFA, UFA, UFA, UFA, UFA, RFA, UFA, R...
## $ Salary     <dbl> 0.700, 4.700, 3.500, 1.000, 3.600, 2.000, 5.000, 5....
## $ grp_Salary <dbl> 0.75, 4.50, 3.50, 1.50, 3.50, 2.50, 5.50, 5.50, 0.7...
```

```
#removing grouped salaries for the first run
dt_hockey <- dt_hockey%>%
  select(-grp_Salary)
```

```
skim(dt_hockey)
```

```
## Skim summary statistics
##  n obs: 881
##  n variables: 25
##
## -- Variable type:factor ---------------------------------------------------------------------------
##  variable missing complete   n n_unique                       top_counts
##      Hand       0      881 881        2            L: 551, R: 330, NA: 0
##       Pos       0      881 881        8 D: 303, C: 232, LW: 171, RW: 137
##    Status       0      881 881        2        UFA: 500, RFA: 381, NA: 0
##  ordered
##    FALSE
##    FALSE
##    FALSE
##
## -- Variable type:integer --------------------------------------------------------------------------
##  variable missing complete   n     mean       sd  p0   p25   p50   p75
##         A       0      881 881    14.24    14.14   0     3    10    21
##       Age       0      881 881    26.08     4.31  18    23    25    29
```

```
##         G       0      881 881     8.45     9.25   0     1     5    13
##        GP       0      881 881    51.88    28.45   1    24    63    78
##      iGVA       0      881 881    25.87    22.71   0     7    21    38
##     iMiss       0      881 881    35.22    28.32   0    10    31    54
##     iRush       0      881 881     4.54     4.35   0     1     4     7
##      iSCF       0      881 881    84.62    75.08   0    21    65   131
##       iSF       0      881 881    92.16    73.43   0    26    81   141
##      iTKA       0      881 881    20.99    18.43   0     5    17    32
##       NZS       0      881 881   278.09   189.01   0    97   287   424
##       OZS       0      881 881   296.03   233.62   0    80   255   472
##       PTS       0      881 881    22.69    22      0     4    16    34
##    Shifts       0      881 881  1113.27   700.65   5   427  1239  1702
##       TOI       0      881 881 51374.44 34022.02 205 18564 54407 79116
##    p100     hist
##      68 <U+2587><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##      45 <U+2583><U+2587><U+2587><U+2583><U+2582><U+2581><U+2581><U+2581>
##      49 <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      82 <U+2583><U+2582><U+2581><U+2581><U+2582><U+2582><U+2583><U+2587>
##     129 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     171 <U+2587><U+2585><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##      24 <U+2587><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##     352 <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     355 <U+2587><U+2585><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581>
##     111 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     751 <U+2587><U+2583><U+2585><U+2586><U+2586><U+2583><U+2582><U+2581>
##     992 <U+2587><U+2585><U+2583><U+2583><U+2583><U+2582><U+2581><U+2581>
##     108 <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##    2511 <U+2587><U+2583><U+2583><U+2583><U+2586><U+2587><U+2583><U+2582>
##  132031 <U+2587><U+2583><U+2583><U+2585><U+2586><U+2585><U+2582><U+2581>
##
## -- Variable type:numeric ------------------------------------------------------------------------
##    variable missing complete   n   mean     sd   p0   p25    p50    p75
##          GC       0      881 881   8.48   8.28 0     1.7    6     12.9
##        Pass       0      881 881 151.19 140.12 0    33.9  124.7  231.9
##     PTSvsGP       0      881 881   0.36   0.27 0     0.18   0.3    0.5
##      Salary       0      881 881   2.44   2.44 0.65  0.74   0.92   4
##        TOff       0      881 881 320.73 261.88 0    84.9  286.5  491.9
##  TOIpercent       0      881 881  25.27   6.98 6.25 20.21  25.24  29.89
##     TOIvsGP       0      881 881  15.12   4.31 3.42 11.97  15.08  18.02
##     p100     hist
##    40.6  <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##   686.1  <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     1.5  <U+2586><U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##    13.8  <U+2587><U+2581><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##  1435.3  <U+2587><U+2585><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581>
##    44.18 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
##    26.84 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
```

Average hockey player in the NHL in the 2017/2018 season is 2.4MM

```
# last quick check for missing data
apply(dt_hockey, 2, function(x) any(is.na(x)))
```

```
##       Hand        Age        Pos         GP          G          A
##      FALSE      FALSE      FALSE      FALSE      FALSE      FALSE
```

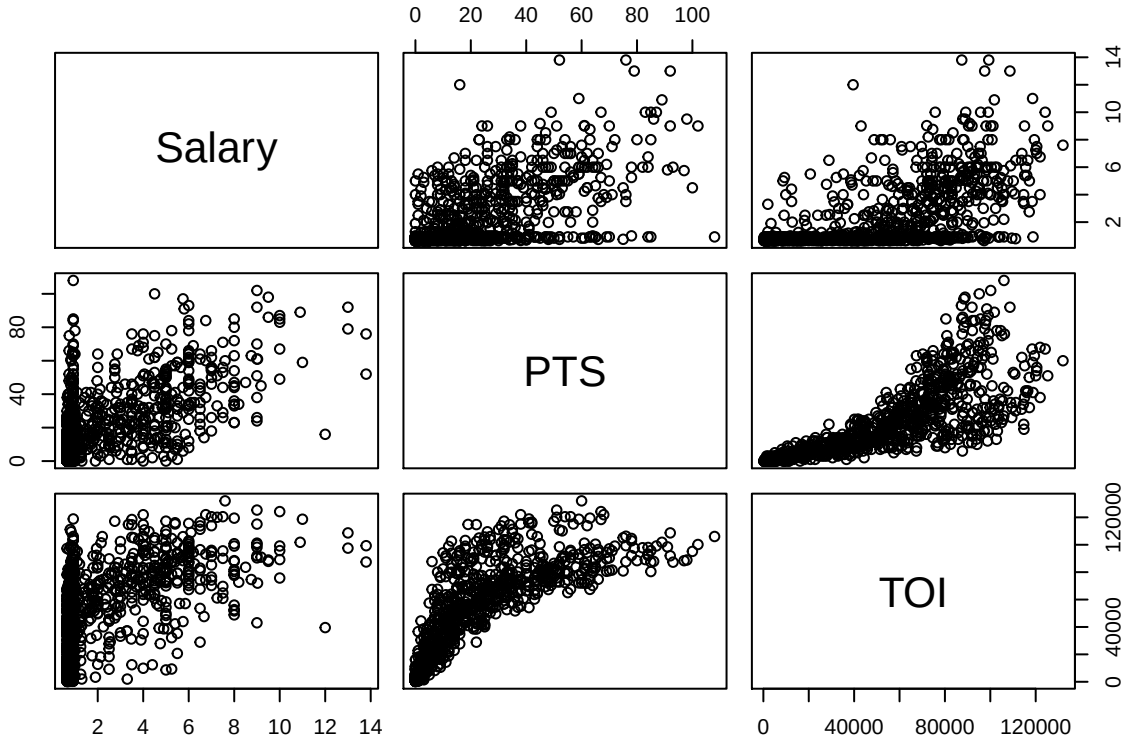
```
##         PTS    PTSvsGP          GC      Shifts         TOI    TOIvsGP
##       FALSE      FALSE       FALSE       FALSE       FALSE      FALSE
## TOIpercent         iSF        iSCF       iRush        Pass       TOff
##       FALSE      FALSE       FALSE       FALSE       FALSE      FALSE
##         OZS        NZS       iMiss        iGVA        iTKA     Status
##       FALSE      FALSE       FALSE       FALSE       FALSE      FALSE
##      Salary
##       FALSE
```

Correlation Matrix

```
# checking to see the corelation between common categories
cor(dt_hockey[c("Salary", "G", "A", "PTS", "Shifts", "TOI")])
```

```
##            Salary         G         A       PTS    Shifts       TOI
## Salary 1.0000000 0.4966459 0.6234975 0.6097064 0.5536948 0.5862746
## G      0.4966459 1.0000000 0.7579382 0.9077969 0.6240139 0.6349533
## A      0.6234975 0.7579382 1.0000000 0.9616462 0.7596010 0.7991290
## PTS    0.6097064 0.9077969 0.9616462 1.0000000 0.7507698 0.7807839
## Shifts 0.5536948 0.6240139 0.7596010 0.7507698 1.0000000 0.9889227
## TOI    0.5862746 0.6349533 0.7991290 0.7807839 0.9889227 1.0000000
```

```
pairs(dt_hockey[c("Salary", "PTS", "TOI")])
```

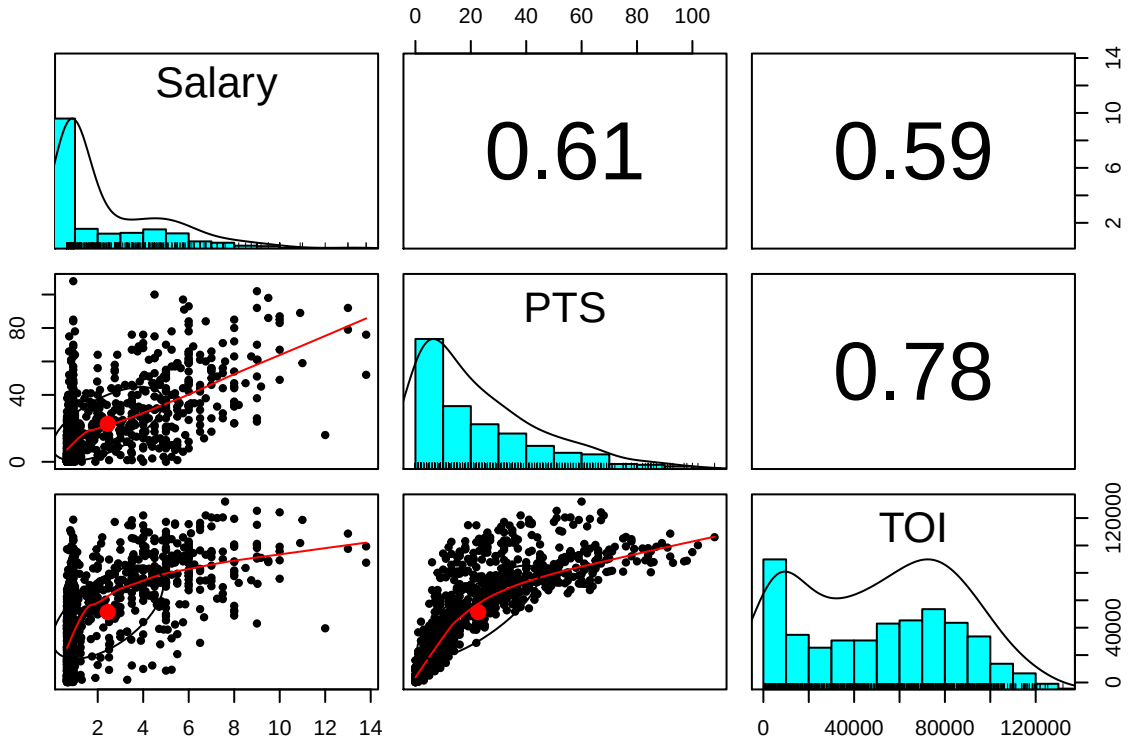



```
library(psych)
```

```
##
```

```
## Attaching package: 'psych'

## The following object is masked from 'package:randomForest':
##
##     outlier

## The following objects are masked from 'package:scales':
##
##     alpha, rescale

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
# pairs pannels pg. 192 for description
pairs.panels(dt_hockey[c("Salary", "PTS", "TOI")])
```



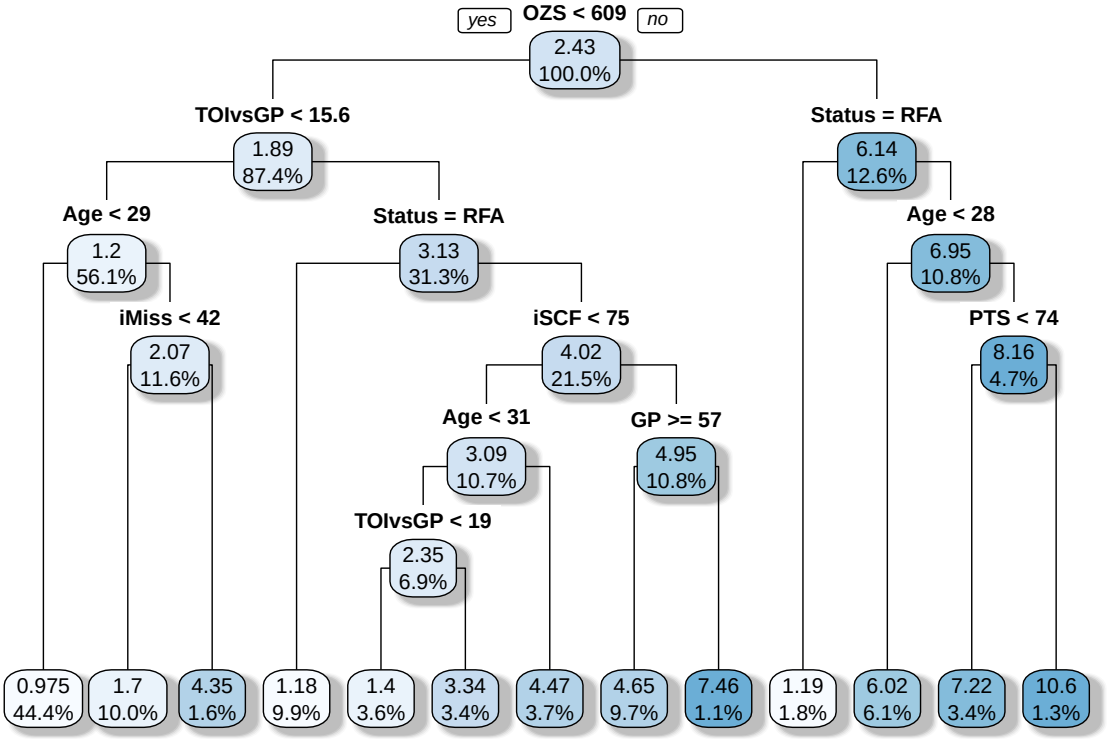
Model Training

```r
#Using seed to ensure my tests are consistant using the different models
set.seed(1025)

#creating training data with a 70/30 split
indx = createDataPartition(dt_hockey$Salary, p = 0.7, list=FALSE)

#Train and test data created
hockeyTrain <- dt_hockey[indx, ]
hockeyTest <- dt_hockey[-indx, ]
```

```r
#Creating the decision tree with our training data
# grp_Salary is the depentant variable,
hockey.rpart <- rpart(Salary ~., data = hockeyTrain, method = "anova")

# viewing basic information about he tree, pg 210
hockey.rpart
```

```
## n= 619
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 619 3736.300000  2.428165
##    2) OZS< 608.5 541 1823.323000  1.893686
##      4) TOIvsGP< 15.615 347  361.725100  1.202898
##        8) Age< 28.5 275   96.163810  0.975257 *
##        9) Age>=28.5 72  196.881100  2.072361
##         18) iMiss< 41.5 62  101.329700  1.704597 *
##         19) iMiss>=41.5 10   35.175560  4.352500 *
##      5) TOIvsGP>=15.615 194  999.840700  3.129270
##       10) Status=RFA 61   34.055340  1.179954 *
##       11) Status=UFA 133  627.686000  4.023317
##         22) iSCF< 74.5 66  304.063600  3.086263
##           44) Age< 30.5 43  122.065800  2.347519
##             88) TOIvsGP< 19.02 22   25.866250  1.396629 *
##             89) TOIvsGP>=19.02 21   55.467850  3.343690 *
##           45) Age>=30.5 23  114.658000  4.467391 *
##         23) iSCF>=74.5 67  208.582100  4.946385
##           46) GP>=56.5 60  146.313100  4.653463 *
##           47) GP< 56.5 7   12.993390  7.457143 *
##    3) OZS>=608.5 78  686.513100  6.135256
##      6) Status=RFA 11    4.103951  1.193182 *
##      7) Status=UFA 67  369.634900  6.946642
##       14) Age< 27.5 38  114.082700  6.021711 *
##       15) Age>=27.5 29  180.445300  8.158621
##         30) PTS< 74 21   64.227380  7.219048 *
##         31) PTS>=74 8   49.015000 10.625000 *
```

```r
# create regression tree
dt <- rpart(Salary ~., data = hockeyTrain, method = "anova")

# plot tree
rpart.plot(dt, type = 1, digits = 3, shadow.col = "gray" )
```



iMiss = shots that missed the net. iSCF = Scoring chances by this player

```
varImp(dt)
```

```
##               Overall
## A           0.2834796
## Age         1.6660505
## G           0.2766998
## GC          0.8709974
## GP          0.2362405
## iMiss       0.3066616
## iSCF        0.6029430
## iSF         0.3085721
## NZS         1.2186043
## OZS         0.9221375
## Pass        0.1359221
## Pos         0.1379150
## PTS         0.5558088
## PTSvsGP     0.6496790
## Shifts      0.4190901
## Status      1.0266535
## TOff        0.6734957
## TOI         0.7941551
## TOIpercent  1.0150362
## TOIvsGP     1.0739730
## Hand        0.0000000
## iRush       0.0000000
```

```
## iGVA       0.0000000
## iTKA       0.0000000
```

Evaluating Model Performance

```
# running test data through the model
p.rpart <- predict(dt, hockeyTest)

# printing results
printcp(dt)
```

```
##
## Regression tree:
## rpart(formula = Salary ~ ., data = hockeyTrain, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age     GP      iMiss   iSCF    OZS     PTS     Status  TOIvsGP
##
## Root node error: 3736.3/619 = 6.036
##
## n= 619
##
##          CP nsplit rel error  xerror     xstd
## 1  0.328256      0   1.00000 1.00401 0.083529
## 2  0.123587      1   0.67174 0.79339 0.065937
## 3  0.090490      2   0.54816 0.64171 0.056768
## 4  0.083712      3   0.45767 0.58758 0.053798
## 5  0.030790      4   0.37395 0.47979 0.046458
## 6  0.020102      5   0.34316 0.45758 0.045350
## 7  0.018382      6   0.32306 0.43540 0.044530
## 8  0.018023      7   0.30468 0.43516 0.045405
## 9  0.017987      8   0.28666 0.43458 0.045383
## 10 0.016159      9   0.26867 0.43508 0.045465
## 11 0.013188     10   0.25251 0.41537 0.043358
## 12 0.010902     11   0.23932 0.41433 0.043203
## 13 0.010000     12   0.22842 0.40727 0.040146
```

```
# lowest xerror, then at xerror and xstd for target error
0.37778 + 0.041408
```

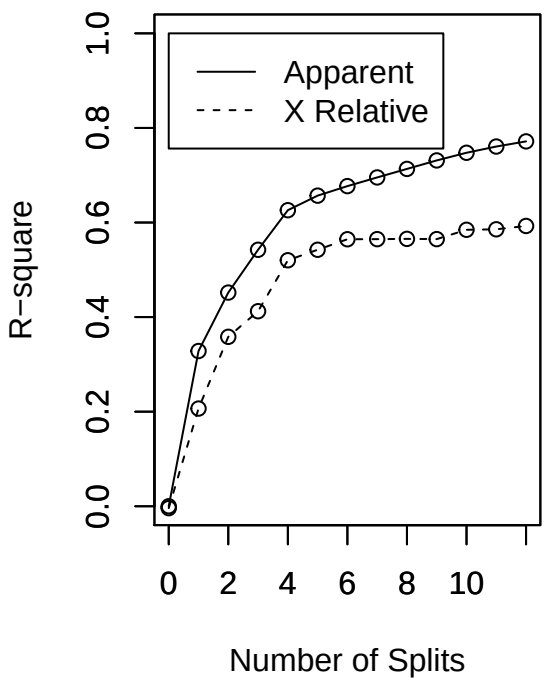
```
## [1] 0.419188
```

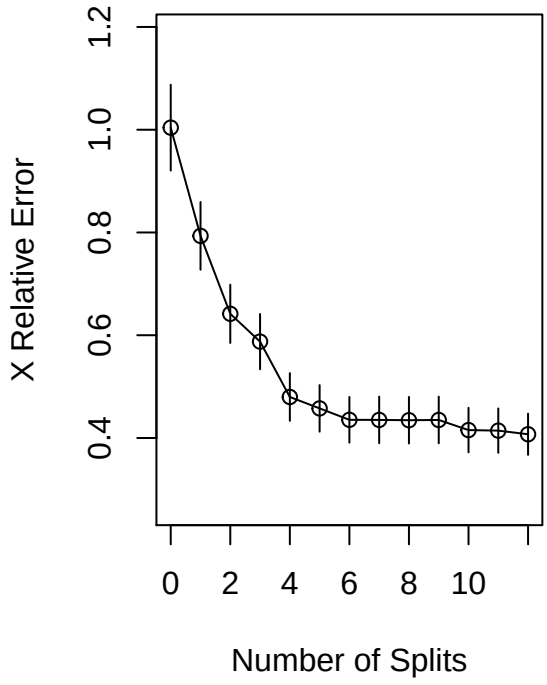
Target Error is 0.419567

```
# creating two plots
par(mfrow=c(1,2))

# ploting dt
rsq.rpart(dt)
```

```
##
## Regression tree:
## rpart(formula = Salary ~ ., data = hockeyTrain, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age     GP      iMiss   iSCF    OZS     PTS     Status  TOIvsGP
```

```
##
## Root node error: 3736.3/619 = 6.036
##
## n= 619
##
##          CP nsplit rel error  xerror     xstd
## 1  0.328256      0   1.00000 1.00401 0.083529
## 2  0.123587      1   0.67174 0.79339 0.065937
## 3  0.090490      2   0.54816 0.64171 0.056768
## 4  0.083712      3   0.45767 0.58758 0.053798
## 5  0.030790      4   0.37395 0.47979 0.046458
## 6  0.020102      5   0.34316 0.45758 0.045350
## 7  0.018382      6   0.32306 0.43540 0.044530
## 8  0.018023      7   0.30468 0.43516 0.045405
## 9  0.017987      8   0.28666 0.43458 0.045383
## 10 0.016159      9   0.26867 0.43508 0.045465
## 11 0.013188     10   0.25251 0.41537 0.043358
## 12 0.010902     11   0.23932 0.41433 0.043203
## 13 0.010000     12   0.22842 0.40727 0.040146
```





The first chart shows how R-Squared is improves as the increase, since R-square gets better as it nears one. Therefore our model is improving with each split.

The second chart illustrates our decreasing errot with each split. The fact that our error continues to near zero and does not treand upward indicates that the tree is trimed.

```
# Evalutate the trees performance
prediction <-
```

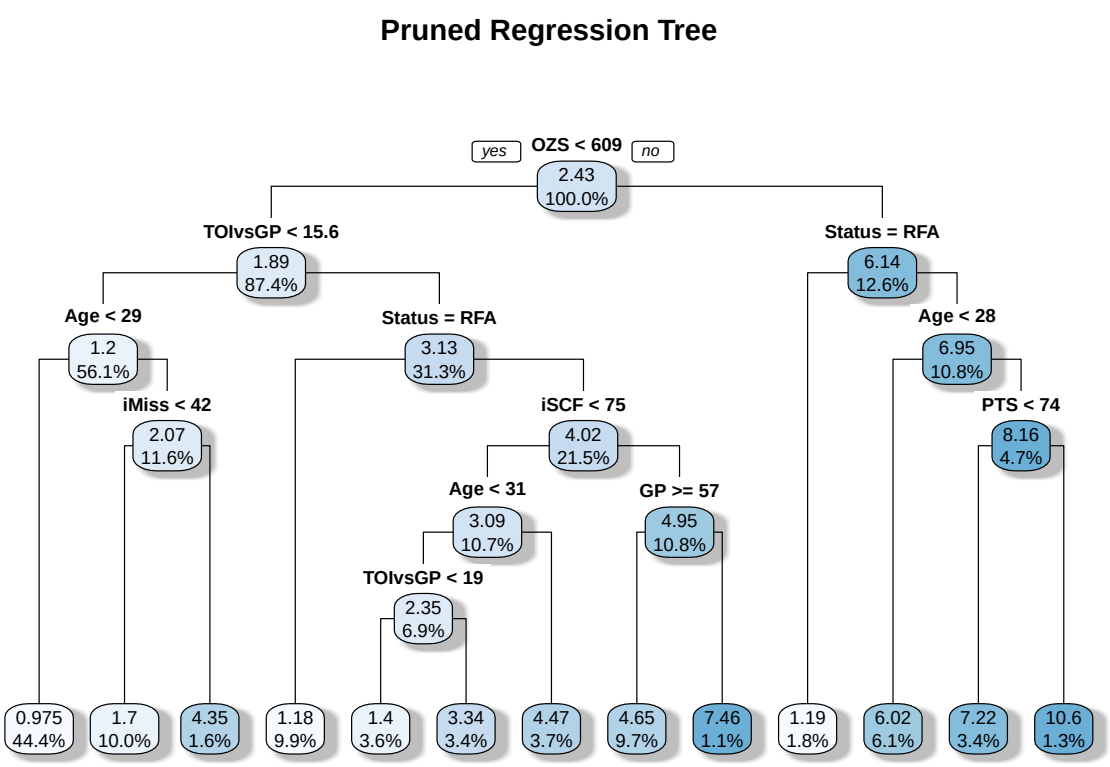
```
  predict(dt, hockeyTest, method = "anova")
```

Prune the tree

```
#prune tree on the optimal CP
optimalCP <- dt$cptable[which.min(dt$cptable[ ,"xerror"]), "CP"]

#Print optimal 0.0179865
print(optimalCP)
```

```
## [1] 0.01
```

```
#prune tree
pruneTree <- prune(dt, cp = optimalCP)

#rpart.plot(tree.fit, type = 1, digits = 3)
#Plot results
rpart.plot(pruneTree, main = "Pruned Regression Tree",
           tweak = 1, gap=0, type = 1, digits = 3, shadow.col = "gray")
```




```
# calling function for calculating MAE
MAE(prediction, hockeyTest$Salary)
```

```
## [1] 1.063867
```

```r
# calling function for calculating RMSE
RMSE(prediction, hockeyTest$Salary)
```

```
## [1] 1.631958
```

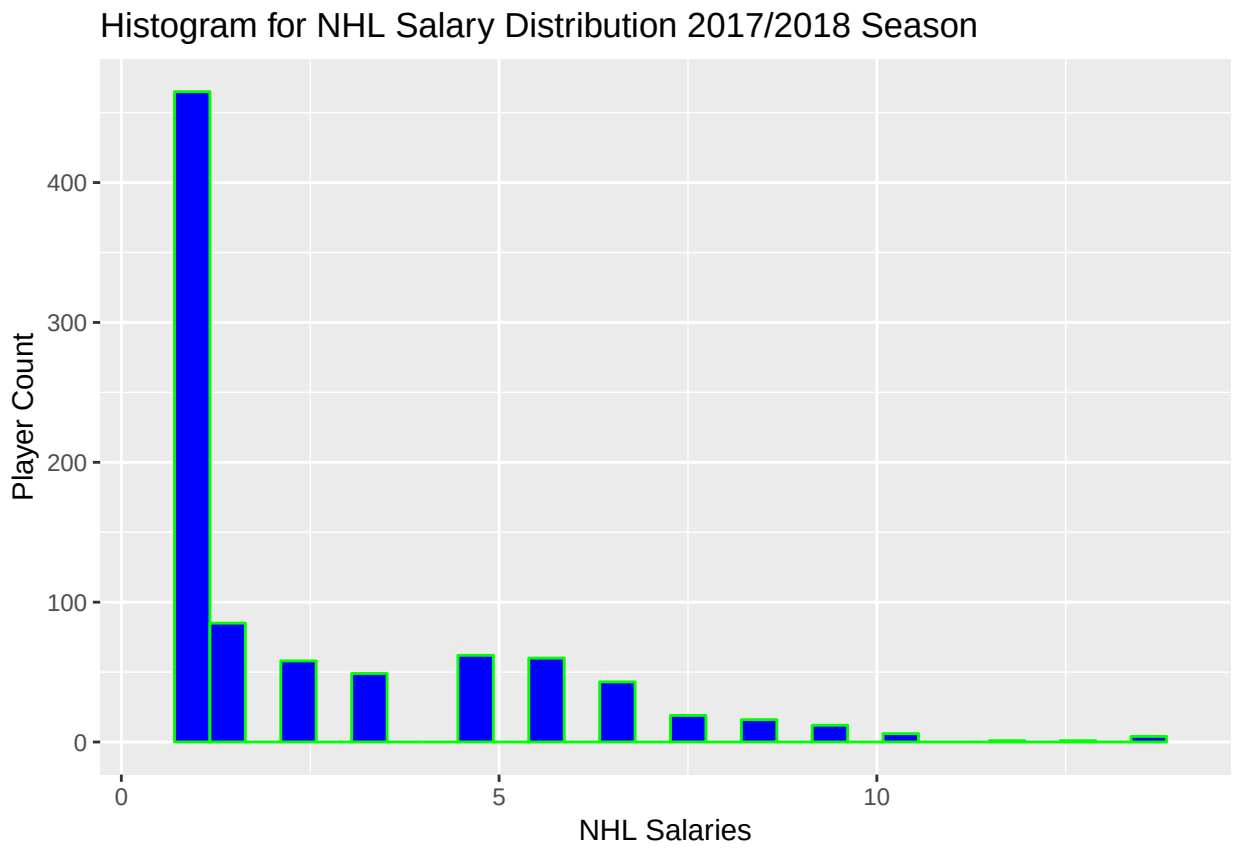
Decision Tree run #2 with grouped salaries

```r
# remove duplicate salaries columns
dt_hockey2<-cleaned_hockey%>%
  select(-Salary)

# to be used in another model
nn_cleaned_hockey <- dt_hockey2
```

```r
glimpse(dt_hockey2)
```

```
## Observations: 881
## Variables: 25
## $ Hand       <fct> R, L, R, L, L, L, L, L, L, L, R, R, L, L, R, L, L, ...
## $ Age        <int> 28, 30, 27, 35, 36, 38, 27, 37, 22, 26, 24, 36, 37,...
## $ Pos        <fct> RW, C, D, LW, D, LW, C, D, D, LW, D, RW, D, D, RW, ...
## $ GP         <int> 33, 62, 82, 62, 80, 82, 81, 81, 1, 7, 24, 82, 79, 1...
## $ G          <int> 3, 3, 3, 13, 4, 13, 24, 0, 0, 0, 1, 16, 4, 0, 4, 40...
## $ A          <int> 6, 5, 7, 11, 19, 16, 26, 10, 1, 2, 1, 35, 23, 5, 4,...
## $ PTS        <int> 9, 8, 10, 24, 23, 29, 50, 10, 1, 2, 2, 51, 27, 5, 8...
## $ PTSvsGP    <dbl> 0.27, 0.13, 0.12, 0.39, 0.29, 0.35, 0.62, 0.12, 1.0...
## $ GC         <dbl> 3.4, 2.9, 3.5, 9.9, 7.8, 11.2, 19.9, 2.9, 0.3, 0.6,...
## $ Shifts     <int> 491, 952, 1759, 1099, 2215, 1383, 1844, 1797, 14, 8...
## $ TOI        <int> 23617, 39114, 82396, 44707, 104951, 58776, 86436, 9...
## $ TOIvsGP    <dbl> 11.93, 10.51, 16.75, 12.02, 21.86, 11.95, 17.79, 19...
## $ TOIpercent <dbl> 20.18, 17.88, 28.08, 20.52, 36.52, 20.19, 29.24, 32...
## $ iSF        <int> 37, 45, 98, 82, 73, 81, 151, 54, 0, 11, 27, 198, 76...
## $ iSCF       <int> 42, 41, 57, 79, 53, 78, 183, 33, 0, 15, 12, 183, 49...
## $ iRush      <int> 0, 1, 1, 5, 5, 5, 12, 0, 0, 1, 2, 10, 5, 1, 0, 7, 7...
## $ Pass       <dbl> 71.5, 38.9, 113.3, 181.6, 148.8, 184.0, 245.2, 56.2...
## $ TOff       <dbl> 135.5, 112.9, 364.3, 314.6, 338.8, 317.0, 509.2, 18...
## $ OZS        <int> 134, 165, 381, 290, 429, 268, 495, 384, 4, 12, 112,...
## $ NZS        <int> 109, 178, 447, 242, 671, 398, 395, 500, 1, 20, 91, ...
## $ iMiss      <int> 10, 10, 61, 25, 43, 24, 61, 21, 0, 5, 19, 76, 21, 6...
## $ iGVA       <int> 8, 13, 31, 25, 85, 29, 29, 44, 0, 2, 14, 85, 52, 7,...
## $ iTKA       <int> 15, 14, 15, 14, 18, 38, 40, 17, 0, 3, 5, 46, 14, 4,...
## $ Status     <fct> UFA, UFA, UFA, UFA, UFA, UFA, UFA, UFA, RFA, UFA, R...
## $ grp_Salary <dbl> 0.75, 4.50, 3.50, 1.50, 3.50, 2.50, 5.50, 5.50, 0.7...
```

```r
ggplot(dt_hockey2, aes(x = grp_Salary))+
  geom_histogram(col= "green",
                 fill = "blue")+
  labs(title = "Histogram for NHL Salary Distribution 2017/2018 Season")+
  labs(x = "NHL Salaries", y = "Player Count")+
  xlim(c(0.4,14))
```



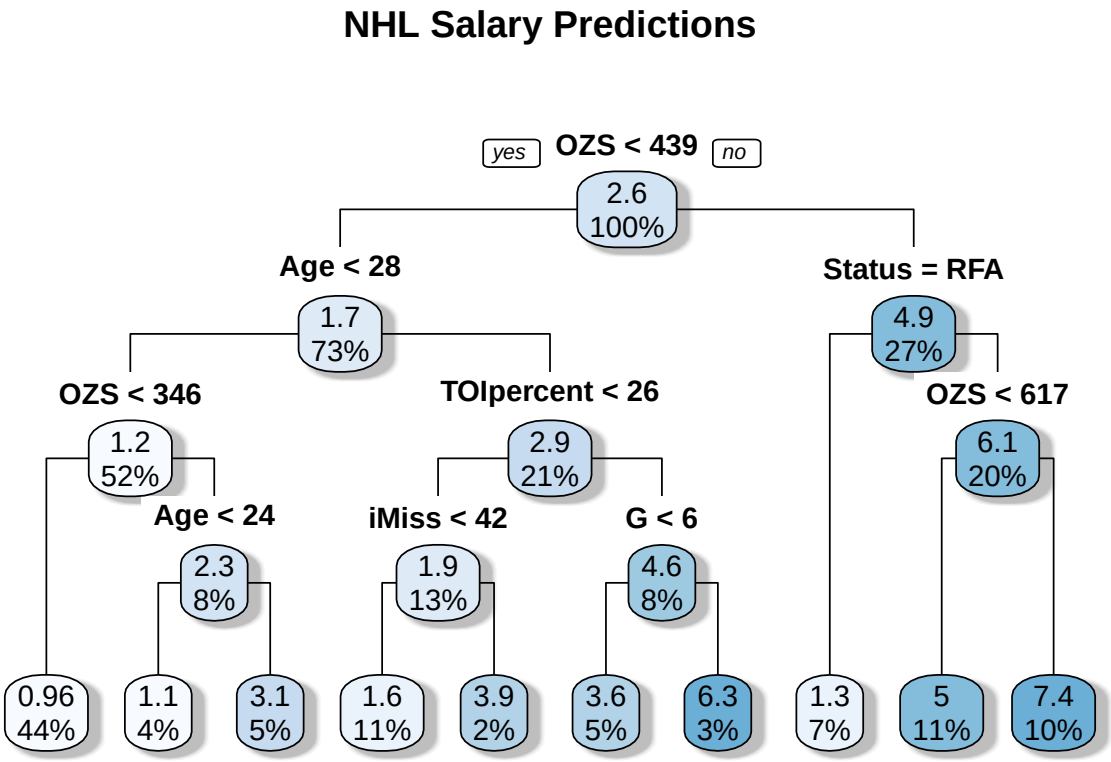
Histogram showing the distribution of salaries in the NHL 2017/2018

```
#Using seed to ensure my tests are consistant using the different models
set.seed(1025)

#creating training data with a 70/30 split
indx = createDataPartition(dt_hockey2$grp_Salary, p = 0.7, list=FALSE)

#Train and test data created
hockeyTrain2 <- dt_hockey2[indx, ]
hockeyTest2 <- dt_hockey2[-indx, ]

#Creating the decision tree with our training data
# Salary is the depentant variable,
hockey.rpart2 <- rpart(grp_Salary ~., data = hockeyTrain2, method = "anova")

# viewing basic information about he tree, pg 210
hockey.rpart2
```

```
## n= 618
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 618 4019.30600 2.5513750
##    2) OZS< 438.5 450 1252.70600 1.6738890
##      4) Age< 27.5 320  313.90920 1.1679690
```

```
##         8) OZS< 345.5 269  109.67520 0.9591078 *
##         9) OZS>=345.5 51  130.60540 2.2696080
##          18) Age< 23.5 22   12.09091 1.1363640 *
##          19) Age>=23.5 29   68.82759 3.1293100 *
##       5) Age>=27.5 130  655.27690 2.9192310
##        10) TOIpercent< 25.625 81  170.27010 1.9135800
##          20) iMiss< 41.5 71  104.85390 1.6338030 *
##          21) iMiss>=41.5 10   20.40000 3.9000000 *
##        11) TOIpercent>=25.625 49  267.67350 4.5816330
##          22) G< 5.5 31   75.87097 3.5645160 *
##          23) G>=5.5 18  104.50000 6.3333330 *
##     3) OZS>=438.5 168 1492.00400 4.9017860
##       6) Status=RFA 43   50.21512 1.3313950 *
##       7) Status=UFA 125  705.07500 6.1300000
##        14) OZS< 616.5 66  198.31160 5.0037880 *
##        15) OZS>=616.5 59  329.40890 7.3898310 *
```

```r
# training the second model
dt2 <- rpart(grp_Salary ~., data = hockeyTrain2, method = "anova")

# ploting second model
rpart.plot(dt2, tweak = 1, gap=0, type = 1, shadow.col = "gray",
           main = "NHL Salary Predictions")
```



Evaluating Second Model Performance

```
# test model with new data
p.rpart <- predict(dt2, hockeyTest2)

# print model results
printcp(dt2)
```

```
##
## Regression tree:
## rpart(formula = grp_Salary ~ ., data = hockeyTrain2, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age        G          iMiss      OZS        Status     TOIpercent
##
## Root node error: 4019.3/618 = 6.5037
##
## n= 618
##
##          CP nsplit rel error  xerror     xstd
## 1  0.317118      0   1.00000 1.00269 0.076169
## 2  0.183294      1   0.68288 0.74864 0.062628
## 3  0.070539      2   0.49959 0.57021 0.054781
## 4  0.054072      3   0.42905 0.52039 0.050102
## 5  0.044126      4   0.37498 0.44144 0.043788
## 6  0.021721      5   0.33085 0.37842 0.037032
## 7  0.018319      6   0.30913 0.37710 0.037191
## 8  0.012362      7   0.29081 0.34468 0.033129
## 9  0.011200      8   0.27845 0.35926 0.034835
## 10 0.010000      9   0.26725 0.38108 0.039414
```

```
# calling function for calculating MAE
MAE(p.rpart, hockeyTest2$grp_Salary)
```

```
## [1] 0.9933494
```

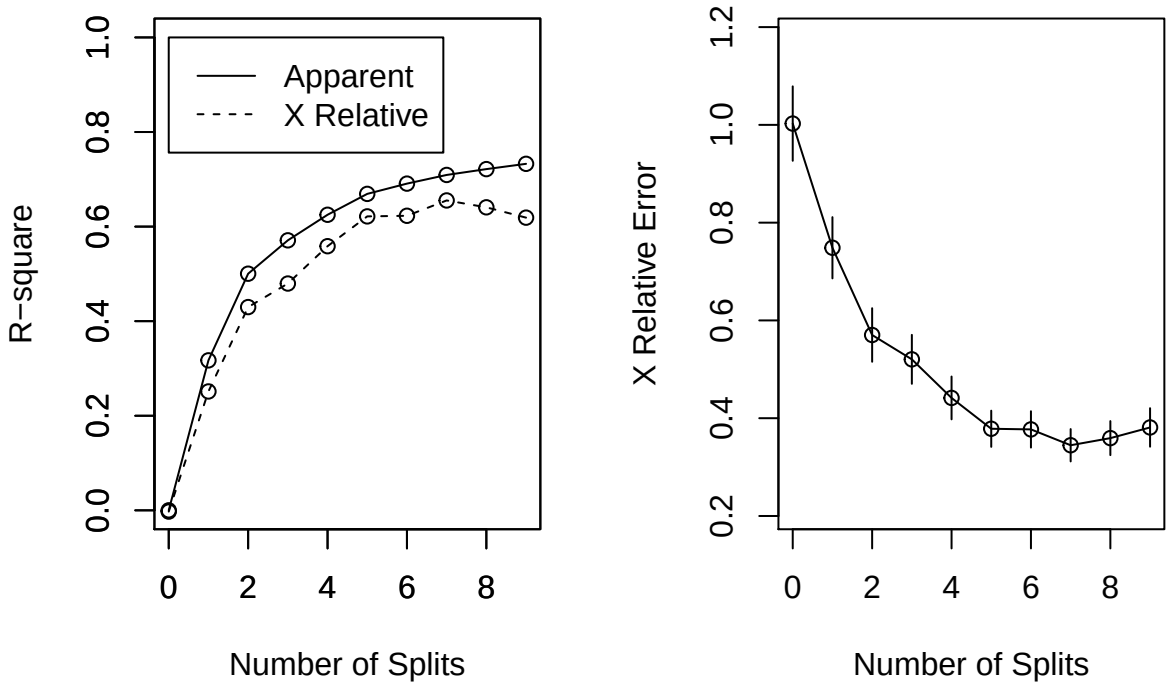
```
# calling function for calculating RMSE
RMSE(p.rpart, hockeyTest2$grp_Salary)
```

```
## [1] 1.523315
```

```
# create plot for R-square and X Rel error/splits
par(mfrow=c(1,2))
rsq.rpart(dt2)
```

```
##
## Regression tree:
## rpart(formula = grp_Salary ~ ., data = hockeyTrain2, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age        G          iMiss      OZS        Status     TOIpercent
##
## Root node error: 4019.3/618 = 6.5037
##
## n= 618
##
##          CP nsplit rel error  xerror     xstd
```

```
## 1  0.317118      0   1.00000 1.00269 0.076169
## 2  0.183294      1   0.68288 0.74864 0.062628
## 3  0.070539      2   0.49959 0.57021 0.054781
## 4  0.054072      3   0.42905 0.52039 0.050102
## 5  0.044126      4   0.37498 0.44144 0.043788
## 6  0.021721      5   0.33085 0.37842 0.037032
## 7  0.018319      6   0.30913 0.37710 0.037191
## 8  0.012362      7   0.29081 0.34468 0.033129
## 9  0.011200      8   0.27845 0.35926 0.034835
## 10 0.010000      9   0.26725 0.38108 0.039414
```



The MAE and RMSE is lower in the second run, as well as the number of required splits.

Neural Networks

```
# copying data in NN place holder
nn_hockey <- nn_cleaned_hockey

# view data to determine if it needs to be normalized
skim(nn_hockey)
```

```
## Skim summary statistics
##  n obs: 881
##  n variables: 25
##
## -- Variable type:factor ---------------------------------------------------------------------------
```

```
##  variable missing complete   n n_unique                      top_counts
##      Hand       0      881 881        2           L: 551, R: 330, NA: 0
##       Pos       0      881 881        8 D: 303, C: 232, LW: 171, RW: 137
##    Status       0      881 881        2        UFA: 500, RFA: 381, NA: 0
##  ordered
##    FALSE
##    FALSE
##    FALSE
##
## -- Variable type:integer ---------------------------------------------------------------------
##  variable missing complete   n     mean       sd  p0   p25   p50   p75
##         A       0      881 881    14.24    14.14   0     3    10    21
##       Age       0      881 881    26.08     4.31  18    23    25    29
##         G       0      881 881     8.45     9.25   0     1     5    13
##        GP       0      881 881    51.88    28.45   1    24    63    78
##      iGVA       0      881 881    25.87    22.71   0     7    21    38
##     iMiss       0      881 881    35.22    28.32   0    10    31    54
##     iRush       0      881 881     4.54     4.35   0     1     4     7
##      iSCF       0      881 881    84.62    75.08   0    21    65   131
##       iSF       0      881 881    92.16    73.43   0    26    81   141
##      iTKA       0      881 881    20.99    18.43   0     5    17    32
##       NZS       0      881 881   278.09   189.01   0    97   287   424
##       OZS       0      881 881   296.03   233.62   0    80   255   472
##       PTS       0      881 881    22.69    22      0     4    16    34
##    Shifts       0      881 881  1113.27   700.65   5   427  1239  1702
##       TOI       0      881 881 51374.44 34022.02 205 18564 54407 79116
##    p100     hist
##      68 <U+2587><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##      45 <U+2583><U+2587><U+2587><U+2583><U+2582><U+2581><U+2581><U+2581>
##      49 <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##      82 <U+2583><U+2582><U+2581><U+2581><U+2582><U+2582><U+2583><U+2587>
##     129 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     171 <U+2587><U+2585><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##      24 <U+2587><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##     352 <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     355 <U+2587><U+2585><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581>
##     111 <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##     751 <U+2587><U+2583><U+2585><U+2586><U+2586><U+2583><U+2582><U+2581>
##     992 <U+2587><U+2585><U+2583><U+2583><U+2583><U+2582><U+2581><U+2581>
##     108 <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##    2511 <U+2587><U+2583><U+2583><U+2583><U+2586><U+2587><U+2583><U+2582>
##  132031 <U+2587><U+2583><U+2583><U+2585><U+2586><U+2585><U+2582><U+2581>
##
## -- Variable type:numeric ---------------------------------------------------------------------
##    variable missing complete   n   mean     sd   p0   p25    p50    p75
##          GC       0      881 881   8.48   8.28 0     1.7    6     12.9
##  grp_Salary       0      881 881   2.51   2.55 0.75  0.75   0.75   4.5
##        Pass       0      881 881 151.19 140.12 0    33.9  124.7  231.9
##     PTSvsGP       0      881 881   0.36   0.27 0     0.18   0.3    0.5
##        TOff       0      881 881 320.73 261.88 0    84.9  286.5  491.9
##  TOIpercent       0      881 881  25.27   6.98 6.25 20.21  25.24  29.89
##     TOIvsGP       0      881 881  15.12   4.31 3.42 11.97  15.08  18.02
##     p100     hist
##    40.6  <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
```

```
##    13.5  <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##   686.1  <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##     1.5  <U+2586><U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##  1435.3  <U+2587><U+2585><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581>
##    44.18 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
##    26.84 <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
```

Clearly looking at the dataset, you can see the requirement to normalize the data, based on features like TOI and shifts. Since Neural Networks perform best when data is scaled close to zero, I will normalize the dataset.

```
# since NN require numerical values, I will remove other values
nn_hockey <- nn_hockey %>%
  select(-Hand, -Pos)

# changing position and
nn_hockey$Status <- as.numeric(nn_hockey$Status)
str(nn_hockey$Status)
```

```
##  num [1:881] 2 2 2 2 2 2 2 2 1 2 ...
```

Normalizing Data

```
# function that will normalize vaules in our dataset
normalize <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}
```

```
# calling the normalization function
normalized_hockey <- as.data.frame(lapply(nn_hockey, normalize))

# viewing the normalized data
skim(normalized_hockey)
```

```
## Skim summary statistics
##  n obs: 881
##  n variables: 23
##
## -- Variable type:numeric ------------------------------------------------------------------------
##    variable missing complete   n mean   sd p0   p25  p50  p75 p100
##           A       0      881 881 0.21 0.21  0 0.044 0.15 0.31    1
##         Age       0      881 881 0.3  0.16  0 0.19  0.26 0.41    1
##           G       0      881 881 0.17 0.19  0 0.02  0.1  0.27    1
##          GC       0      881 881 0.21 0.2   0 0.042 0.15 0.32    1
##          GP       0      881 881 0.63 0.35  0 0.28  0.77 0.95    1
##  grp_Salary       0      881 881 0.14 0.2   0 0     0    0.29    1
##        iGVA       0      881 881 0.2  0.18  0 0.054 0.16 0.29    1
##       iMiss       0      881 881 0.21 0.17  0 0.058 0.18 0.32    1
##       iRush       0      881 881 0.19 0.18  0 0.042 0.17 0.29    1
##        iSCF       0      881 881 0.24 0.21  0 0.06  0.18 0.37    1
##         iSF       0      881 881 0.26 0.21  0 0.073 0.23 0.4     1
##        iTKA       0      881 881 0.19 0.17  0 0.045 0.15 0.29    1
##         NZS       0      881 881 0.37 0.25  0 0.13  0.38 0.56    1
##         OZS       0      881 881 0.3  0.24  0 0.081 0.26 0.48    1
##        Pass       0      881 881 0.22 0.2   0 0.049 0.18 0.34    1
##         PTS       0      881 881 0.21 0.2   0 0.037 0.15 0.31    1
##     PTSvsGP       0      881 881 0.24 0.18  0 0.12  0.2  0.33    1
```

```
##      Shifts       0      881 881 0.44 0.28  0 0.17  0.49 0.68    1
##      Status       0      881 881 0.57 0.5   0 0     1    1       1
##        TOff       0      881 881 0.22 0.18  0 0.059 0.2  0.34    1
##         TOI       0      881 881 0.39 0.26  0 0.14  0.41 0.6     1
##  TOIpercent       0      881 881 0.5  0.18  0 0.37  0.5  0.62    1
##     TOIvsGP       0      881 881 0.5  0.18  0 0.37  0.5  0.62    1
##      hist
##  <U+2587><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2583><U+2587><U+2587><U+2583><U+2582><U+2581><U+2581><U+2581>
##  <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2583><U+2582><U+2581><U+2581><U+2582><U+2582><U+2583><U+2587>
##  <U+2587><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2585><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##  <U+2587><U+2585><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581>
##  <U+2587><U+2585><U+2583><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2585><U+2586><U+2586><U+2583><U+2582><U+2581>
##  <U+2587><U+2585><U+2583><U+2583><U+2583><U+2582><U+2581><U+2581>
##  <U+2587><U+2585><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2582><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2586><U+2587><U+2585><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2583><U+2583><U+2586><U+2587><U+2583><U+2582>
##  <U+2586><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581><U+2587>
##  <U+2587><U+2585><U+2585><U+2582><U+2582><U+2581><U+2581><U+2581>
##  <U+2587><U+2583><U+2583><U+2585><U+2586><U+2585><U+2582><U+2581>
##  <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
##  <U+2581><U+2582><U+2586><U+2587><U+2587><U+2585><U+2582><U+2581>
```

Building NN Model

```r
#Setting seed to be able to reproduce the results
set.seed(1025)

#Creating 70% Training parition
indx <- createDataPartition(normalized_hockey$grp_Salary, p= 0.7, list=FALSE)

#Train and test data created
nTrain <- normalized_hockey[indx, ]
nTest <- normalized_hockey[-indx, ]

# creating model
nn_model <- neuralnet(grp_Salary ~ A + Age + G +  GC + GP + iGVA +iMiss +iRush + iSCF +iTKA + NZS +OZS +
               Status + TOff + TOI + TOIpercent + TOIvsGP, data = nTrain, stepmax = 1000000, hidden = 1

# plotting NN model
plot(nn_model)
```

Predicting with Neutral Networks

```
# predict using NN
predict_testNN <- compute(nn_model, nTest[,c(1:22)])
```

Evaluating model performance

```
predicted_salary <- predict_testNN$net.result

# corelating model success
cor(predicted_salary, nTest$grp_Salary)
```

```
##           [,1]
## [1,] 0.8502163
```

85.6% of the instances NN was able to accurately predict the appropirate salary group.

```
# calling function to calculate MAE
MAE(predict_testNN$net.result, nTest$grp_Salary)
```

```
## [1] 0.0669543
```

```
# calling function to calculate RMSE
RMSE(predict_testNN$net.result, nTest$grp_Salary)
```

```
## [1] 0.1056555
```

Model improvement

```
# creating second model
nn_model2 <- neuralnet(grp_Salary ~ A + Age + G +  GC + GP + iGVA +iMiss +iRush + iSCF +iTKA + NZS +OZS
               Status + TOff + TOI + TOIpercent + TOIvsGP, data = nTrain, stepmax = 1000000, hidden = 2

# plotting NN model
plot(nn_model2)
```

Predicting with Neutral Networks

```
# predict using NN
predict_testNN2 <- compute(nn_model2, nTest[,c(1:22)])

# exporting results to variable
predicted_salary2 <- predict_testNN2$net.result

# corelating model success
cor(predicted_salary2, nTest$grp_Salary)
```

```
##           [,1]
## [1,] 0.8521131
```

I tried increasing the number of hidden relationships up to 5, however in most cases my accuracy dropped while increasing the processing time of the model.

```
# Calculate MAE
MAE(predict_testNN2$net.result, nTest$grp_Salary)
```

```
## [1] 0.06912613
```

```r
# calculate RMSE
RMSE(predict_testNN2$net.result, nTest$grp_Salary)
```

```
## [1] 0.1054525
```