```
#install.packages("ggeffects")

library(ggeffects)

library(moments)


# Store the data in a new variable allTeamsData, this includes data for all the teams for season 2015

allTeamsData <- "Load the .rmd file into R environment and then store it in an object"
```

1. Organise and clean the data

We need to fetch data into a subset based on student Id 2035261. So for team Id 6 and 1 we will fetch the data into a subset as follows:

```
# subset() allTeamsData based on teamID.x = 6(LAA) and 1(TEX)

myTeamsData <- subset(allTeamsData, teamID.x == "LAA" | teamID.x == "TEX")
```

It can be seen that we have loaded a subset of allTeamsData into myTeamsData based on teamID.x(LAA and TEX)

1.2 Data quality analysis

Looking at data and its quality visually and numerically/descriptively is an important part of Data Quality Analysis.

We will have a birds eye view at the data to check if there is any bad data or any issues with our data, we will check the structure of the data and also apply some validations and checks based on the provided metadata. Looking at top and bottom values can also be beneficial for our analysis.

We can summarize our data frame and look at min, max, mean, median and quantile values and check if they are in-line with ranges and rules provided in our metadata.

```
# head() can be used to look at the top 6(default=6, can be changed) rows of a data frame.

head(myTeamsData)
```

```
##     playerID teamID.x   G  R   H  AB RBI weight height   salary  birthDate
## 16 alvarjo02      LAA  64  0   0   0   0    180     71   509500 1989-05-06
## 23 andruel01      TEX 160 69 154 596  62    200     72 15000000 1988-08-26
## 37 aybarer01      LAA 156 74 161 597  44    195     70  8500000 1984-01-14
## 47  bassan01      TEX  33  0   0   0   0    200     74   725000 1987-11-01
```

```
## 58 beltrad01      TEX 143 83 163 567  83    220     71 16000000 1979-04-
07

## 97 buterdr01      LAA  10  3   4  21   0    205     73   987500 1983-08-
09

##    career.length bats      age hit.ind

## 16      1.563313    L 25.65640       0

## 23      5.738535    R 26.34908       1

## 37      8.629706    B 30.96509       1

## 47      3.553730    R 27.16769       0

## 58     16.522930    R 35.73717       1

## 97      4.731006    R 31.39767       1
```

```
# tail() is also used to have a look at the bottom 6(default=6, can be chan
ged) rows of a data frame.

tail(myTeamsData)
```

```
##      playerID teamID.x  G  R  H  AB RBI weight height       salary  birt
hDate

## 943 gonzeto02      LAA 48  0  0   0   0    218     75 1000000.0000 1982-
07-12

## 959 lewisco01      TEX 33  0  0   2   0    240     76 4000000.0000 1979-
08-02

## 960 mosslaf01      TEX 94 36 73 337  50    210     73     650.0075 1983-
09-16

## 961 murphpr04      TEX 84 22 61 206  27    210     75 6000000.0000 1981-
10-18

## 962 perezjj03      TEX 70 30 42 184 -21    220     71  508600.0000 1988-
12-23

## 963 rabbibb01      TEX 82 22 52 173  29    585     72 2500000.0000 1981-
04-17

##     career.length bats      age hit.ind

## 943     9.2813142    R 32.47365       1

## 959    12.7529090    R 35.41684       0

## 960     7.4058864    L 31.29363       1

## 961     8.3312799    L 23.20465       1

## 962     0.4791239    R 26.02327       1

## 963    10.3025325    R 33.70842       1
```

```
# View() allows us to scroll through an entire data frame. As we have less
than 100 rows its a good practice to see the data frame.

View(myTeamsData)

# str() provides information about the data types, column names, dimensions
and some values of the data frame.

str(myTeamsData)
```

```
## 'data.frame':    80 obs. of  15 variables:
```

```
##  $ playerID     : chr  "alvarjo02" "andruel01" "aybarer01" "bassan01" ..
.
##  $ teamID.x     : Factor w/ 149 levels "ALT","ANA","ARI",..: 71 131 71 1
31 131 71 71 131 131 131 ...
##  $ G            : int  64 160 156 33 143 10 159 78 149 18 ...
##  $ R            : int  0 69 74 0 83 3 78 33 94 0 ...
##  $ H            : int  0 154 161 0 163 4 161 54 153 0 ...
##  $ AB           : int  0 596 597 0 567 21 630 233 555 0 ...
##  $ RBI          : int  0 62 44 0 83 0 83 34 82 0 ...
##  $ weight       : int  180 200 195 200 220 205 215 210 210 180 ...
##  $ height       : int  71 72 70 74 71 73 70 73 71 75 ...
##  $ salary       : num  509500 15000000 8500000 725000 16000000 ...
##  $ birthDate    : Date, format: "1989-05-06" "1988-08-26" ...
##  $ career.length: num  1.56 5.74 8.63 3.55 16.52 ...
##  $ bats         : Factor w/ 3 levels "B","L","R": 2 3 1 3 3 3 2 3 2 2 ..
.
##  $ age          : num  25.7 26.3 31 27.2 35.7 ...
##  $ hit.ind      : num  0 1 1 0 1 1 1 1 1 0 ...
```

```
# One way to look at column names is names() function which returns all the
column names only and its easy for the reader
names(myTeamsData)
```

```
##  [1] "playerID"      "teamID.x"      "G"             "R"
##  [5] "H"             "AB"            "RBI"           "weight"
##  [9] "height"        "salary"        "birthDate"     "career.length"
## [13] "bats"          "age"           "hit.ind"
```

```
# summary() is a very informative function which provides us with the overa
ll description/summary of the data set
summary(myTeamsData)
```

```
##    playerID           teamID.x        G                R
##  Length:80          TEX    :42   Min.   :  2.00   Min.   :  0.00
##  Class :character   LAA    :38   1st Qu.: 24.75   1st Qu.:  0.00
##  Mode  :character   ALT    : 0   Median : 36.00   Median :  2.50
##                     ANA    : 0   Mean   : 56.41   Mean   : 16.81
##                     ARI    : 0   3rd Qu.: 79.00   3rd Qu.: 22.00
##                     ATL    : 0   Max.   :160.00   Max.   :104.00
##                     (Other): 0
##        H               AB             RBI             weight
##  Min.   :  0.0   Min.   :  0.0   Min.   :-21.00   Min.   :180.0
##  1st Qu.:  0.0   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:200.0
```

```
##  Median :  2.0   Median : 13.5   Median :  0.00   Median :210.0
##  Mean   : 33.9   Mean   :132.6   Mean   : 16.31   Mean   :217.1
##  3rd Qu.: 45.0   3rd Qu.:189.5   3rd Qu.: 25.00   3rd Qu.:225.0
##  Max.   :187.0   Max.   :630.0   Max.   : 98.00   Max.   :585.0
##
##      height          salary           birthDate          career.length
##  Min.   :68.00   Min.   :     500   Min.   :1979-01-18   Min.   : 0.3422
##  1st Qu.:72.00   1st Qu.:  515988   1st Qu.:1983-04-08   1st Qu.: 2.5804
##  Median :74.00   Median : 1231000   Median :1985-12-14   Median : 5.3908
##  Mean   :73.44   Mean   : 4663137   Mean   :1985-10-26   Mean   : 5.8028
##  3rd Qu.:75.00   3rd Qu.: 5868750   3rd Qu.:1988-03-20   3rd Qu.: 8.3354
##  Max.   :79.00   Max.   :24000000   Max.   :2000-09-23   Max.   :35.4127
##
##  bats        age           hit.ind
##  B: 4   Min.   :20.91   Min.   :0.000
##  L:25   1st Qu.:26.78   1st Qu.:0.000
##  R:51   Median :29.05   Median :1.000
##         Mean   :29.24   Mean   :0.575
##         3rd Qu.:31.69   3rd Qu.:1.000
##         Max.   :35.95   Max.   :1.000
##
```

```
# There are other ways to find mean(), median(), min(), max() functions, bu
t in summary() we can do all these in one go

# We can check that if there are any NA values in our data frame.

anyNA(myTeamsData)
```

```
## [1] FALSE
```

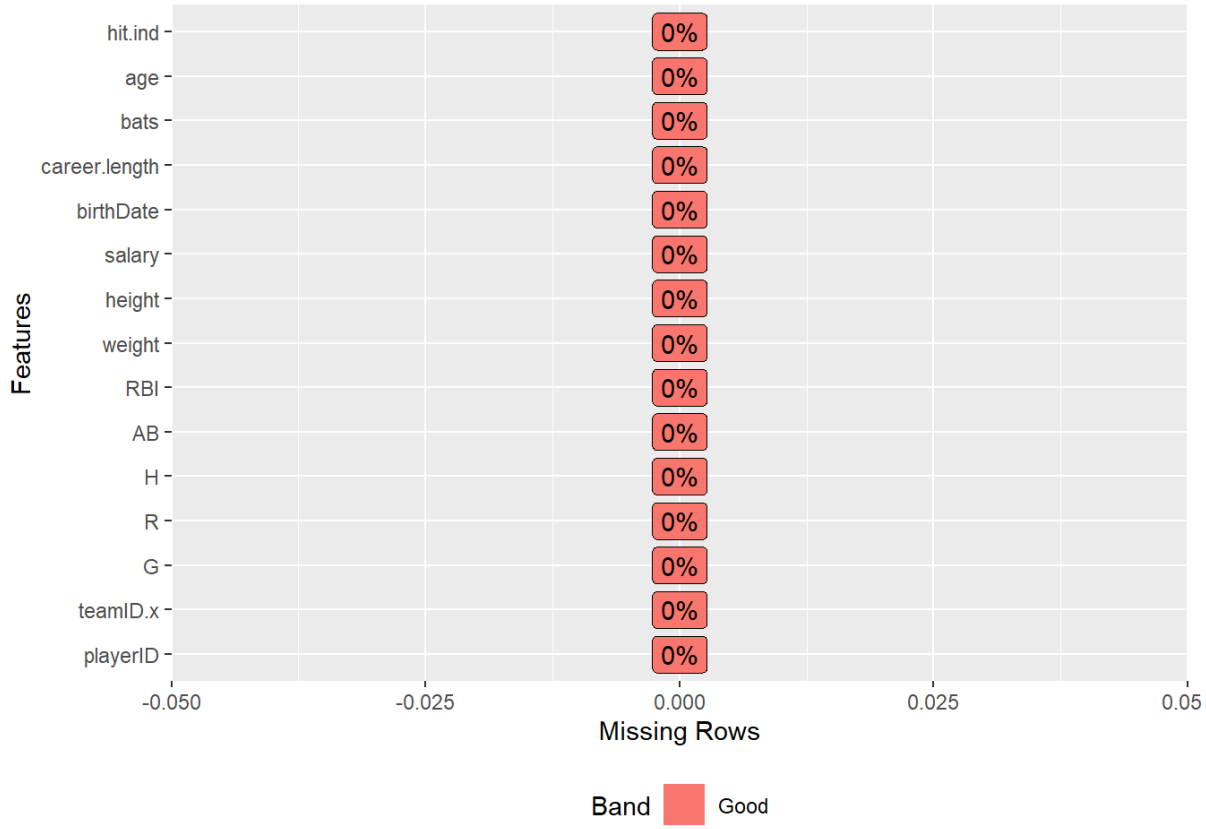
```
#We can apply some rules to our data frame using validate package and then
look at the results

baseballRules <- validator(
  okG = G >= 0,
  okR = R >= 0,
  okH = H >= 0,
  okAB = AB >= 0,
  okRBI = RBI >= 0,
  okWeight = weight >= 150 &
    weight <= 322,
  okHeight = height >= 60 & height <= 82,
  okSalary = salary >= 507500 & salary <= 30000000,
```

```
  okCareer = career.length >= 0 &
    career.length < 25,
  okBats = is.element(bats, c("B", "L", "R")),
  okAge = age >= 20 & age <= 42
)
#Ref: Baseball Statistics

ruleCheck <- confront(myTeamsData, baseballRules)
summary(ruleCheck)
##         name items passes fails nNA error warning
## 1        okG    80     80     0   0 FALSE   FALSE
## 2        okR    80     80     0   0 FALSE   FALSE
## 3        okH    80     80     0   0 FALSE   FALSE
## 4       okAB    80     80     0   0 FALSE   FALSE
## 5     okRBI    80     79     1   0 FALSE   FALSE
## 6  okWeight    80     79     1   0 FALSE   FALSE
## 7  okHeight    80     80     0   0 FALSE   FALSE
## 8  okSalary    80     77     3   0 FALSE   FALSE
## 9  okCareer    80     79     1   0 FALSE   FALSE
## 10   okBats    80     80     0   0 FALSE   FALSE
## 11    okAge    80     80     0   0 FALSE   FALSE
##                                   expression
## 1                          (G - 0) >= -1e-08
## 2                          (R - 0) >= -1e-08
## 3                          (H - 0) >= -1e-08
## 4                         (AB - 0) >= -1e-08
## 5                        (RBI - 0) >= -1e-08
## 6             weight >= 150 & weight <= 322
## 7               height >= 60 & height <= 82
## 8        salary >= 507500 & salary <= 3e+07
## 9  career.length >= 0 & career.length < 25
## 10      is.element(bats, c("B", "L", "R"))
## 11                    age >= 20 & age <= 42
#We can see that we have 3 observations that failed the rules provided base
d on meta data and based on baseball research data.


#We also need to check if we have duplicate values in our data frame
```

```
duplicated(myTeamsData)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
#The result shows there are no duplicates.


#plot_missing() from DataExplorer helps us check for any missing values vis
ually.
plot_missing(myTeamsData)
```



We have 80 rows and 15 columns in our data frame and details of those are as follow:

1. playerID is a character vector which contains unique Player ID.
2. teamID.x is a factor which stores a 3 char team ID of the player.

3. G is an integer vector storing the number of games played by a player and the maximum number of matches in a regular season can be 162 and our data is in that range. (Ref: Wikipedia)
4. Similarly R is amount of runs scored by a player and is an integer vector, the top scorer had scored 122 runs and our data meets that range.(Ref: Baseball Reference)
5. H too is an integer vector of number of successful hits by a player. The top player had 205 hits and our data is in that range.(Ref: Baseball Reference)
6. AB is an integer vector and it stores the number of times a player had a chance to play.
7. RBI is an integer vector which stores number of runs batted, A run is batted in if it results in a point to be scored.
8. birthDate is date of birth of a player. 42 was the age of the oldest player who played 2015 season and our data meets that criteria.(Ref: Baseball Reference)
9. weight is an integer vector containing weight of a player in pounds. The lower range looks fine for our data but the heaviest player to play baseball was 322 pounds and our data looks little suspicious and we need to look at our data.(Superbaseball2020)
10. height is an integer vector containing height of a player in inches and the data looks plausible. salary is a number vector and the range looks fine. But some values are below the minimum salary bar which we need to check.
11. career.length are number vector which stores length of career of a player. One of the players has a career length of 35+ years which is not possible according to sources.(Ref: Papp, Charlie. Longest Career in MLB History.)
12. bats is a factor vector which stores the info Whether a player bat with their Left (L) or Right (R) hand or Both (B)
13. hit.ind is number vector here which stores 1 if he made a hit in 2015 season else stores 0.
14. age is a number vector storing the age of the player and the data looks fine.

We can see that there are no missing values in our data frame.

1.3 Data cleaning

The column name for most of the columns are not understandable for someone who has no knowledge of baseball domain, so we can update the column names to something easily readable and understandable. RBI has a minimum value of -21, which is not possible, so we need to check this and act on it. weight of one of the players is 585 pounds which looks implausible and some research needs to be done. The minimum and maximum salary is 500 and 24000000 which looks alarming and will be looked upon. career length for one player is over 35 years and we need to take a look at that value too. Hit indicator should be a binary variable but that is loaded as a number, so we will need to change its data type appropriately. We should also compare/check if the birth date and age values match for all the players.

```
# The code written below will update the column names of data frame to mean
ingful names.

colnames(myTeamsData) <-

  c(

    "playerID",

    "teamID",
```

```
    "games",
    "runs",
    "hits",
    "atBats",
    "runsBattedIn",
    "weight",
    "height",
    "salary",
    "birthDate",
    "careerLength",
    "bats",
    "age",
    "hitInd"
  )


# We can also update teamID column by dropping the unwanted levels. droplev
els() will drop the unwanted levels

myTeamsData$teamID <- droplevels(myTeamsData$teamID)

# (Ref: Schork, Jaochin. Drop levels in R)
```

Changing the names made it easier for the reader to understand what a column represents.

We have one player with a runsBattedIn value of -21, which is not possible as described on our meta data, so we need to apply some data cleaning techniques on that observation.

A value of -21 can significantly disturb our analysis, imputing an appropriate value is one of the options but if we see at the values in runsBattedIn column the values are so non-uniform that mean/median imputation wont be a good option, so in this case we will replace -21 with NA. This will cause loss of 1 observation but we can manage that as we have a total of 80 observations.

```
# slice_min() function from tidyverse package returns the observation with
the minimum value from the column provided as the argument.

minRunBattedIn <- myTeamsData %>% slice_min(runsBattedIn)

#We can display the observation with a runsBattedIn of -21

minRunBattedIn
```

```
##    playerID teamID games runs hits atBats runsBattedIn weight height sal
ary

## 1 perezjj03    TEX    70   30   42    184          -21    220     71 508
600

##    birthDate careerLength bats      age hitInd

## 1 1988-12-23    0.4791239    R 26.02327      1
```

```
#We can directly replace the value as myTeamsData$runsBattedIn[myTeamsData$
runsBattedIn <0] <- NA, although we have just 1 value in this column which
```

```
needs to be updated but we will do it using loops which is a better way to
do it if you have multiple values which needs to be changed in your data.


count <- 1 # A variable which will keep track of the item number in the loo
p


#for() loop will go through each item from myTeamsData$runsBattedIn and the
internal ifelse() will check for any value less than 0 and then replace tha
t item based on the count with NA if required otherwise the value will rema
in the same.
for (runs in myTeamsData$runsBattedIn)
{
  ifelse(runs < 0,
         myTeamsData$runsBattedIn[count] <- NA,
         myTeamsData$runsBattedIn[count] <- runs)
  count <- count + 1
}


#The negative value has been replaced with NA
is.na(myTeamsData$runsBattedIn)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
#We could have created a function to delete the row with negative value, in
stead we will use the na.omit() function to remove the observation.
myTeamsData <- na.omit(myTeamsData)


#Observation with NA value has been removed from our data frame.
is.na(myTeamsData$runsBattedIn)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE
```

```
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE

## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE

## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE

## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE F
ALSE

## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Weight of one of the player is 585pounds which is over the weight of heaviest player ever to play baseball.(Ref: Superbaseball2020)

It looks like some data entry error or some other data quality issue which is totally at random and in this case the range of weight is 180-275 pounds and the weight data is well spread so we can use imputation methods to impute an appropriate value so that we do not miss any data.

“Data imputation: is the substitution of an estimated value that is as realistic as possible for a missing or problematic data item. The substituted value is intended to enable subsequent data analysis to proceed.” (Ref: Shepperd, Martin.)

"Three very simple imputation methods are:

1. Mean imputation, where the missing value is replaced with the sample mean. This preserves the estimate of central tendency but at the expense of deflating the estimate of variance. This is potentially problematic especially when many values are imputed.
2. Regression-based imputation, where the missing value is replaced by the predicted value from a regression model (often a simple linear regression model) over the complete cases of the sample. This may bias the variance less than mean imputation but still may not be satisfactory. It assumes the data are missing completely at random (MCAR).
3. Hot deck imputation, where the missing value is replaced by a randomly selected value from the sample. This has the advantage of not biasing the variance estimate but still assumes the data are MCAR." (Ref: Shepperd, Martin.)s

We cannot use the 2. option because the correlation between weight and height is very low and a simple linear regression model won’t be able to predict a good value.

Option 3 can be used in this case to impute/replace the value with a random value from the sample. But we will use option 1. as the data is well spread and we have to impute only 1 value and that will not make our analysis biased.

```
summary(myTeamsData$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

##     180     200     210     217     225     585
```

```
# cor() check the correlation between two continuous variables.

cor(myTeamsData$weight, myTeamsData$height)
```

```
## [1] 0.1271961
```

```
#We can directly replace the value as NA, as we have only one  value which
does not meet the criteria but we will do it using loops which is a better
way to do it if you have multiple values which needs to be changed in your
data.
```

```
count <- 1 # A variable which will keep track of the item number in the loo
p


#for() loop will go through each item from myTeamsData$runsBattedIn and the
internal ifelse() will check for any value less than 0 and then replace tha
t item based on the count with NA if required otherwise the value will rema
in the same.
for (playerWeight in myTeamsData$weight)
{
  ifelse(
    playerWeight < 150 |
      playerWeight > 322,
    myTeamsData$weight[count] <-
      NA,
    myTeamsData$weight[count] <- playerWeight
  )
  count <- count + 1
}


myTeamsData$weight
```

```
##  [1] 180 200 195 200 220 205 215 210 210 180 240 190 235 190 210 200 212
240 235
## [20] 275 180 213 190 205 185 195 205 240 213 230 200 255 245 240 200 200
245 230
## [39] 220 210 225 210 225 195 200 210 235 200 200 210 200 195 200 205 200
215 200
## [58] 225 205 210 205 205 214 235 205 190 210 230 210 185 210 245 195 235
218 240
## [77] 210 210  NA
```

The out of range weight value has been replaced with NA.

Now we will apply option 3. imputation.

```
myTeamsData$weight
```

```
##  [1] 180 200 195 200 220 205 215 210 210 180 240 190 235 190 210 200 212
240 235
## [20] 275 180 213 190 205 185 195 205 240 213 230 200 255 245 240 200 200
245 230
## [39] 220 210 225 210 225 195 200 210 235 200 200 210 200 195 200 205 200
215 200
## [58] 225 205 210 205 205 214 235 205 190 210 230 210 185 210 245 195 235
218 240
## [77] 210 210  NA
```

```
# na_mean() will replace tha NA values from the vector with value given in option=" ". (Ref: Rdocumentation)

myTeamsData$weight <- as.integer(na_mean(myTeamsData$weight, option = "mean"))

myTeamsData$weight
```

```
##  [1] 180 200 195 200 220 205 215 210 210 180 240 190 235 190 210 200 212 240 235

## [20] 275 180 213 190 205 185 195 205 240 213 230 200 255 245 240 200 200 245 230

## [39] 220 210 225 210 225 195 200 210 235 200 200 210 200 195 200 205 200 215 200

## [58] 225 205 210 205 205 214 235 205 190 210 230 210 185 210 245 195 235 218 240

## [77] 210 210 212
```

The NA value has been replaced by a mean value from the sample.

Now we will look at the salary of a player. In 2015 season the minimum salary was 507500USD and the maximum salary was 30000000USD. We will check our data and act upon it accordingly. (Ref: Baseball Reference) (Ref: Gaines, C.)

```
#We can make a user-defined function to count the number of out of range values and then replace them with NA in another function.

count <- 0 # Initial count of the invalid values is 0.

countOutOfRange <- function(val) {

  for (values in val)

  {

    if (values < 507500 | values > 30000000)

      count <- count + 1

  }


  return(count)

}

# Calling to user defined function to get the count of out of range values.

countOutOfRange(myTeamsData$salary)
```

```
## [1] 3
```

```
#We can see the sorted values

sort(myTeamsData$salary)
```

```
##  [1] 5.002550e+02 6.500075e+02 5.070000e+05 5.075000e+05 5.085000e+05

##  [6] 5.085000e+05 5.085000e+05 5.085000e+05 5.095000e+05 5.095000e+05

## [11] 5.095000e+05 5.110000e+05 5.112500e+05 5.125000e+05 5.130000e+05

## [16] 5.138500e+05 5.140000e+05 5.150000e+05 5.152000e+05 5.162500e+05

## [21] 5.170000e+05 5.175000e+05 5.182900e+05 5.197000e+05 5.254150e+05
```

```
## [26] 5.355000e+05 5.375000e+05 5.500000e+05 7.000000e+05 7.250000e+05
## [31] 9.000000e+05 9.750000e+05 9.875000e+05 9.950000e+05 1.000000e+06
## [36] 1.000000e+06 1.000000e+06 1.100000e+06 1.150000e+06 1.312000e+06
## [41] 1.370000e+06 1.700000e+06 2.290000e+06 2.500000e+06 2.950000e+06
## [46] 3.200000e+06 3.200000e+06 3.450000e+06 4.000000e+06 4.000000e+06
## [51] 4.125000e+06 4.125000e+06 4.250000e+06 4.750000e+06 4.750000e+06
## [56] 5.000000e+06 5.250000e+06 5.525000e+06 5.825000e+06 6.000000e+06
## [61] 6.000000e+06 6.083000e+06 6.425000e+06 7.000000e+06 7.400000e+06
## [66] 8.500000e+06 9.400000e+06 1.300000e+07 1.400000e+07 1.400000e+07
## [71] 1.500000e+07 1.600000e+07 1.600000e+07 1.800000e+07 1.800000e+07
## [76] 2.270875e+07 2.350000e+07 2.400000e+07 2.400000e+07
```

We can see that there are 3 values which are out of range and when we sort the salary data we can see those 3 minimum values.

We need to either delete all the 3 observations, which will cause loss of data, so we will impute some feasible values.

```
count <- 1 # A variable which will keep track of the item number in the loo
p
# Replace the 3 minimum values with NA
for (sal in myTeamsData$salary)
{
  ifelse(sal < 507500 |
           sal > 30000000,
         myTeamsData$salary[count] <- NA,
         myTeamsData$salary[count] <- sal)
  count <- count + 1
}
myTeamsData$salary
```

```
##  [1]   509500 15000000  8500000   725000 16000000   987500   537500   51
8290
##  [9] 14000000   508500   975000   995000   512500  5000000  3450000   53
5500
## [17]   507500   508500  4125000 24000000   517000  6425000  1100000 1400
0000
## [25]   516250   550000 23500000 22708749  7400000  5525000  4750000   50
9500
## [33]  9400000  4000000  4750000   515000   509500  2950000   514000  600
0000
## [41] 16000000   511000   525415   513850  1000000  1150000 24000000  131
2000
```

```
## [49]   513000  3200000   511250       NA   900000   508500  1370000  229
0000

## [57]   515200   517500  5250000   508500  7000000  5825000   519700  608
3000

## [65]  4250000 13000000 18000000   700000 18000000  1700000  3200000  100
0000

## [73]       NA  4125000  1000000  4000000       NA  6000000  2500000
```

We can see that 3 out-of-range values has been replaced with NA. If you notice, the values out of range were all so low, it might be pointing to an issue that data is not missing/incorrect at random. If we impute these values based on the understanding that data is totally missing at random then imputation might cause wrong values in data and it may cause bias while doing analysis and fitting any model

So we will delete these 3 observations based on NA values.

```
# Total of 79 observations with 3 NA's in salary

summary(myTeamsData)
```

```
##    playerID         teamID       games             runs             hits
##  Length:79          LAA:38   Min.   :  2.00   Min.   :  0.00   Min.   :
0.0

##  Class :character   TEX:41   1st Qu.: 24.50   1st Qu.:  0.00   1st Qu.:
0.0

##  Mode  :character            Median : 35.00   Median :  2.00   Median :
2.0

##                             Mean   : 56.24   Mean   : 16.65   Mean   :
33.8

##                             3rd Qu.: 80.00   3rd Qu.: 22.00   3rd Qu.:
47.0

##                             Max.   :160.00   Max.   :104.00   Max.   :1
87.0

##

##      atBats       runsBattedIn       weight          height

##  Min.   :  0.0   Min.   : 0.00   Min.   :180.0   Min.   :68.00

##  1st Qu.:  0.0   1st Qu.: 0.00   1st Qu.:200.0   1st Qu.:72.00

##  Median :  6.0   Median : 0.00   Median :210.0   Median :74.00

##  Mean   :131.9   Mean   :16.78   Mean   :212.3   Mean   :73.47

##  3rd Qu.:189.5   3rd Qu.:25.00   3rd Qu.:225.0   3rd Qu.:75.00

##  Max.   :630.0   Max.   :98.00   Max.   :275.0   Max.   :79.00

##

##      salary           birthDate           careerLength     bats

##  Min.   :  507500   Min.   :1979-01-18   Min.   : 0.3422   B: 4

##  1st Qu.:  518092   1st Qu.:1983-04-08   1st Qu.: 2.5982   L:25

##  Median : 1535000   Median :1985-09-30   Median : 5.4127   R:50
```

```
##  Mean   : 4895187   Mean   :1985-10-11   Mean   : 5.8702
##  3rd Qu.: 6000000   3rd Qu.:1988-02-11   3rd Qu.: 8.3395
##  Max.   :24000000   Max.   :2000-09-23   Max.   :35.4127
##  NA's   :3
##       age            hitInd
##  Min.   :20.91   Min.   :0.0000
##  1st Qu.:26.89   1st Qu.:0.0000
##  Median :29.25   Median :1.0000
##  Mean   :29.28   Mean   :0.5696
##  3rd Qu.:31.71   3rd Qu.:1.0000
##  Max.   :35.95   Max.   :1.0000
##
```

```
# Remove the observations with NA values
myTeamsData <- na.omit(myTeamsData)
# Total of 76 observations with no NA's
summary(myTeamsData)
```

```
##    playerID         teamID       games             runs             hits
##  Length:76          LAA:37   Min.   :  2.00   Min.   :  0.00   Min.   :
0
##  Class :character   TEX:39   1st Qu.: 24.75   1st Qu.:  0.00   1st Qu.:
0
##  Mode  :character            Median : 35.00   Median :  1.00   Median :
2
##                             Mean   : 55.53   Mean   : 16.09   Mean   :
33
##                             3rd Qu.: 74.25   3rd Qu.: 18.25   3rd Qu.:
43
##                             Max.   :160.00   Max.   :104.00   Max.   :1
87
##      atBats       runsBattedIn       weight          height
##  Min.   :  0.0   Min.   : 0.00   Min.   :180.0   Min.   :68.00
##  1st Qu.:  0.0   1st Qu.: 0.00   1st Qu.:200.0   1st Qu.:72.00
##  Median :  5.5   Median : 0.00   Median :210.0   Median :74.00
##  Mean   :128.2   Mean   :16.39   Mean   :212.8   Mean   :73.55
##  3rd Qu.:170.8   3rd Qu.:23.50   3rd Qu.:225.0   3rd Qu.:75.00
##  Max.   :630.0   Max.   :98.00   Max.   :275.0   Max.   :79.00
##      salary           birthDate           careerLength     bats
##  Min.   :  507500   Min.   :1979-04-07   Min.   : 0.3422   B: 3
##  1st Qu.:  518092   1st Qu.:1983-04-23   1st Qu.: 2.5804   L:24
```

```
##  Median : 1535000   Median :1986-03-02   Median : 5.3306   R:49
##  Mean   : 4895187   Mean   :1985-12-17   Mean   : 5.7677
##  3rd Qu.: 6000000   3rd Qu.:1988-03-20   3rd Qu.: 8.3313
##  Max.   :24000000   Max.   :2000-09-23   Max.   :35.4127
##       age            hitInd
##  Min.   :20.91   Min.   :0.0000
##  1st Qu.:26.78   1st Qu.:0.0000
##  Median :28.83   Median :1.0000
##  Mean   :29.11   Mean   :0.5526
##  3rd Qu.:31.63   3rd Qu.:1.0000
##  Max.   :35.74   Max.   :1.0000
```

Career length of one of the player is 35years but the record of longest baseball career is 25 years. (Ref: Papp, C.) We will try to impute a proper value using the option regression-based imputation.

Let's try to impute a value based in simple linear regression of career length and age of the player.

```
# Shows the correlation between career length and age.
cor(myTeamsData$careerLength, myTeamsData$age)
```

```
## [1] 0.5164625
```

```
count <- 1 # A variable which will keep track of the item number in the loo
p
# Replace the value above 25 and below 0 to NA
for (career in myTeamsData$careerLength)
{
  ifelse(
    career < 0 |
      career > 25,
    myTeamsData$careerLength[count] <-
      0 ,
    myTeamsData$careerLength[count] <- career
  )
  count <- count + 1
}
#Shows correlation after changing the 35 years career.length value to 0.
cor(myTeamsData$careerLength, myTeamsData$age, use = "complete.obs")
```

```
## [1] 0.788473
```

We can notice that one value i.e 35 years was causing a lot of issues in the correlation of age and career.length. Once the value got removed, The 0.789 correlation coefficient shows a strong positive relation between career length and age.

We will now use linear regression model between career.length and age to predict a value of career length for the observation have a career length as 0.

```
lmCareerLenght <- lm(myTeamsData$careerLength ~ myTeamsData$age)
summary(lmCareerLenght)
```

```
##
## Call:
## lm(formula = myTeamsData$careerLength ~ myTeamsData$age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1245 -1.3009 -0.1035  1.7122  8.0838
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -19.62650    2.27598  -8.623 8.57e-13 ***
## myTeamsData$age   0.85647    0.07767  11.028  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.307 on 74 degrees of freedom
## Multiple R-squared:  0.6217, Adjusted R-squared:  0.6166
## F-statistic: 121.6 on 1 and 74 DF,  p-value: < 2.2e-16
```

The model gives us significant p-values and R-squared value of 0.62, which can be considered fine. So we will use the equation:

$$careerLength = -19.31956 + age \times 0.84740$$careerLength=−19.31956+age×0.84740

to predict the career length for this 1 player.
We noticed that the age of that player is 26.66 years so we will put that value in our model to predict the careerLength.

```
predictionAge <- 26.66
predictedCareer <- -19.31956 + predictionAge * 0.84740
predictedCareer
```

```
## [1] 3.272124
```

The predicted value for careerLength is 3.27 years and we will replace NA with this value in our data frame.

```
myTeamsData$careerLength[myTeamsData$careerLength==0] <- predictedCareer
myTeamsData$careerLength
```

```
##  [1]  1.5633128  5.7385352  8.6297057  3.5537303 16.5229295  4.7310062
##  [7]  2.6119097  3.4579055  9.6974675  0.3860370  5.6563997  3.4360027
## [13]  0.6652977 11.3319644  7.3182752  2.6310746  2.4914442  0.3805613
## [19]  5.4127310  9.5523614  2.5845311  5.7385352  1.7522245  7.5400411
## [25]  3.4086242  6.3107461  8.6406571  7.7508556  5.6947296  8.3477070
## [31]  6.6584531  0.4188912  5.4537988 12.7529090  3.3319644  0.7419576
## [37]  0.4845996  4.4271047  0.6735113  8.3312799  8.6625599  3.3319644
## [43]  2.7351129  0.6516085  2.5133470  4.2737851 13.7494867  5.2922656
## [49]  1.6125941  3.3949350  0.6762491  6.3956194  0.3422313  4.5968515
## [55]  3.4907598  2.5681040  1.2813142  7.7535934  0.4873374  9.7385352
## [61]  5.3689254  2.5681040  3.4852841  6.3408624 11.7508556  8.5995893
## [67]  6.6776181  9.5605749  6.7542779  3.3949350  7.4031485  3.2721240
## [73]  9.2813142 12.7529090  8.3312799 10.3025325
```

The value 0 has been replaced with the predictedCareer value i.e(3.27 years)

hitInd column is loaded as an integer but it should be a factor, so we will convert it to factor using as.factor()

```
myTeamsData$hitInd <- as.factor(myTeamsData$hitInd)
table(myTeamsData$hitInd)
```

```
##
##  0  1
## 34 42
```

hitInd has been changed to factor with levels 0 for no hits and 1 for at least 1 hit.

Another data quality issue can be comparison of birthDate and age. age_calc() will calculate the age of the player based on given dob till 01-01-2015 and add it to a new column calAge.

```
# scipen = 999 suppresses the scientific interger values and displays it in integers.
options(scipen = 999)
myTeamsData$calAge <-
  age_calc(
    myTeamsData$birthDate,
    enddate = as.Date("2015-01-01"),
    units = "years",
    precise = TRUE
  )
```

```
myTeamsData$ageDiff <- myTeamsData$calAge - myTeamsData$age


sort(myTeamsData$ageDiff)
```

```
##  [1] -15.00186774329  -0.00067508650  -0.00066195981  -0.00065445885
##  [5]  -0.00065080060  -0.00064519025  -0.00064133217  -0.00061152815
##  [9]  -0.00060030745  -0.00058347640  -0.00057944924  -0.00057007304
## [13]  -0.00053631872  -0.00051615220  -0.00041255286  -0.00040880238
## [17]  -0.00040880238  -0.00039942618  -0.00039942618  -0.00039942384
## [21]  -0.00037129757  -0.00035629565  -0.00032629181  -0.00026065840
## [25]  -0.00022877931  -0.00021940311  -0.00018189831  -0.00018189831
## [29]  -0.00018002307  -0.00017439734  -0.00003750481  -0.00003000384
## [33]   0.00012001538   0.00015752018   0.00016689638   0.00018002307
## [37]   0.00027941080   0.00030003844   0.00032254133   0.00032441274
## [41]   0.00042005382   0.00043505574   0.00055507112   0.00056444732
## [45]   0.00057757400   0.00062257977   0.00067696174   0.00080072759
## [49]   0.00082510572   0.00085886004   0.00090386581   0.00108951459
## [53]   0.00110639176   0.00113452036   0.00114577180   0.00117015110
## [57]   0.00127516338   0.00129579102   0.00131829391   0.00241245049
## [61]   0.00250221609   0.00273223710   0.00273224044   0.00276029219
## [65]   0.00277151289   0.00287249617   0.00287249919   0.00292299234
## [69]   0.00301836829   0.00318106844   0.00325400298   0.00338304103
## [73]   0.00360184468   0.00385431043   0.00392163463  10.00082510205
```

```
options(scipen = 0)
# scipen = 0 turns the setting into default.
```

We can see from the data that there are 2 players who have a huge age difference with birthDate i.e -15 years and 10 years. It can be due to user error or these can be special cases. But we cannot decide on them without discussing with the data owner and in this case we do not want to miss other data like runs, hits and salary for these records so we will ignore these 2 records but we know we have some issue in our age data.

2. Exploratory Data Analysis (EDA)

2.1 EDA plan

“Exploratory data analysis (EDA) aims to describe the main characteristics of a data set often using visual techniques. Wikipedia defines it as “an approach to analyzing data sets to summarise their main characteristics, often with visual methods”." (Ref: Shepperd, M.)

In our data frame we have a mixture of continuous and categorical variables.

There are different ways in which we can explore the data of both continuous and categorical variables. We can generate histograms, scatter plots, box-plots, line graphs, bar graphs and many more to visualise the data. Our data set contains numerical variables such as games, runs, hits, atBats, runsBattedIn, weight, height, salary, careerLength, age and we have some categorical variables such as teamID, bats and hitInd.

We will start with univariate(single variable) analysis and then we will move to multivariate(multiple variable) analysis. We will look at the spread of data, outliers, range for univariate analysis and then check correlation between multiple variables using scatter plots and other visual means, we will also look and any patterns. If we notice any patterns or issues then we will try to transform the data so that we can do the analysis in a better sense.

2.2 EDA and summary of results

We will start with univariate analysis.

```
# lapply() is used to apply same function to multiple columns in one go.
lapply(myTeamsData, summary)
```

```
## $playerID
##    Length     Class      Mode
##        76 character character
##
## $teamID
## LAA TEX
##  37  39
##
## $games
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   24.75   35.00   55.53   74.25  160.00
##
## $runs
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    1.00   16.09   18.25  104.00
##
## $hits
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       2      33      43     187
##
## $atBats
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     5.5   128.2   170.8   630.0
##
```

```
## $runsBattedIn
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00   16.39   23.50   98.00
##
## $weight
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   180.0   200.0   210.0   212.8   225.0   275.0
##
## $height
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   68.00   72.00   74.00   73.55   75.00   79.00
##
## $salary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   507500   518092  1535000  4895187  6000000 24000000
##
## $birthDate
##         Min.      1st Qu.       Median         Mean      3rd Qu.
Max.
## "1979-04-07" "1983-04-23" "1986-03-02" "1985-12-17" "1988-03-20" "2000-0
9-23"
##
## $careerLength
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3422  2.5804  5.0116  5.3448  7.8980 16.5229
##
## $bats
##  B  L  R
##  3 24 49
##
## $age
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.91   26.78   28.83   29.11   31.63   35.74
##
## $hitInd
##  0  1
## 34 42
##
```

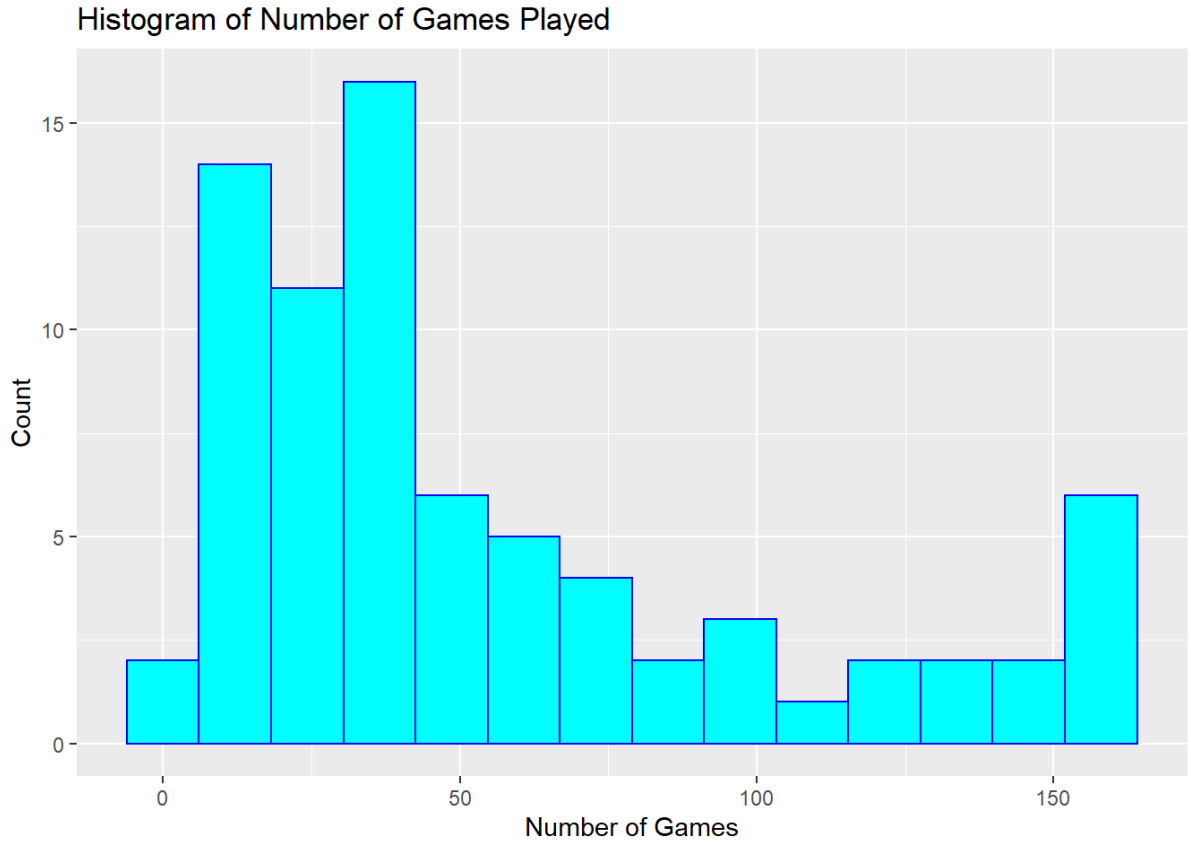
```
## $calAge
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.27   26.79   28.83   29.04   31.69   35.74
##
## $ageDiff
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -15.001868  -0.000399   0.000311  -0.065086   0.001280  10.000825
```

```
#plot_str() from DataExplorer package helps us plot a structure of of data frame.
plot_str(myTeamsData)
```

We can see the range, min, max, mean, quantiles values for different columns.

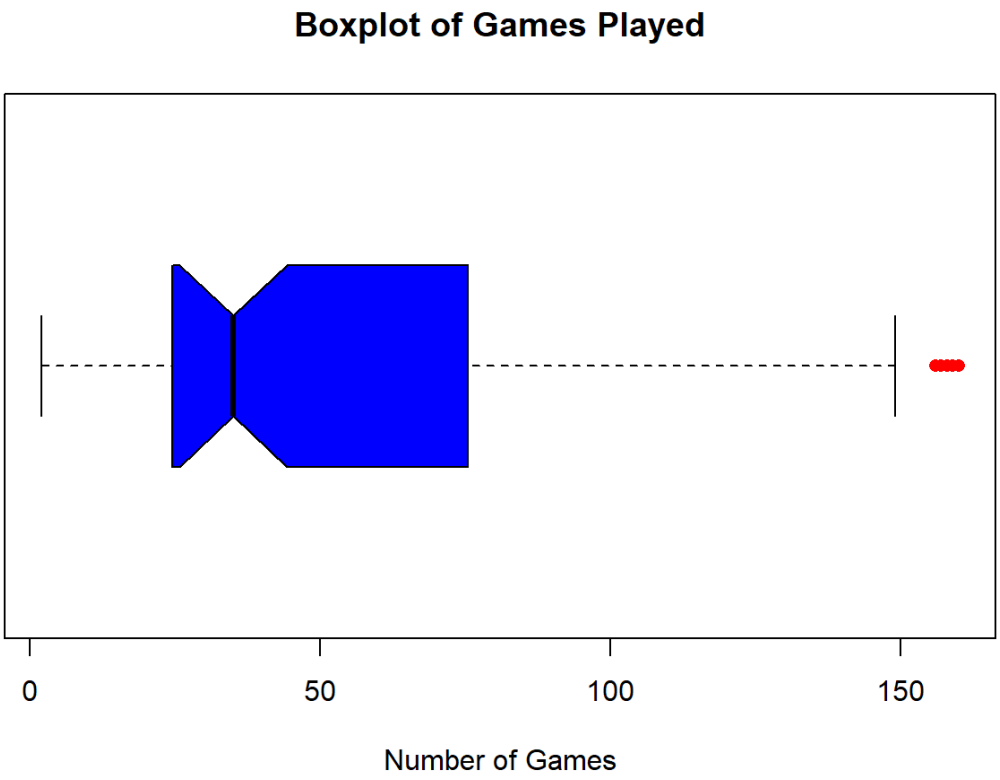
The plot shows us all the variables/columns and also shows us visually which one is continuous and which one is categorical.

```
# Let's start with no of games played by a player


ggplot(myTeamsData,aes(x=games))+geom_histogram(color="blue",bins = 14,fill="cyan")+labs(title="Histogram of Number of Games Played",x="Number of Games",y="Count")
```



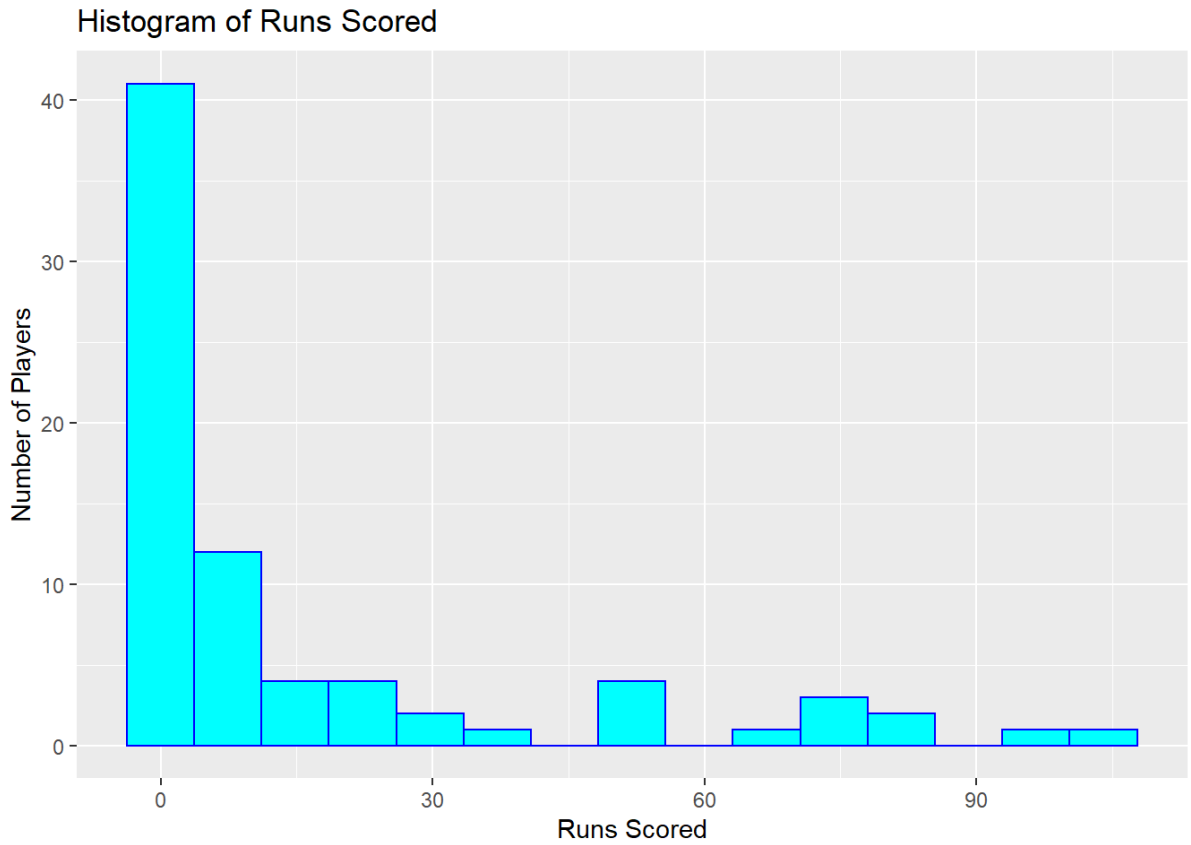

```
boxplot(myTeamsData$games, horizontal = T, notch = T,
```

```
        xlab = "Number of Games", col = 'blue', pars = list(outcol = 'red',
pch = 16))

        title("Boxplot of Games Played")
```



We can notice that the data is skewed to the right with most of the players having a count of matches played between 10 and 50. The boxplot displays some outliers in the data for number of games for a player.

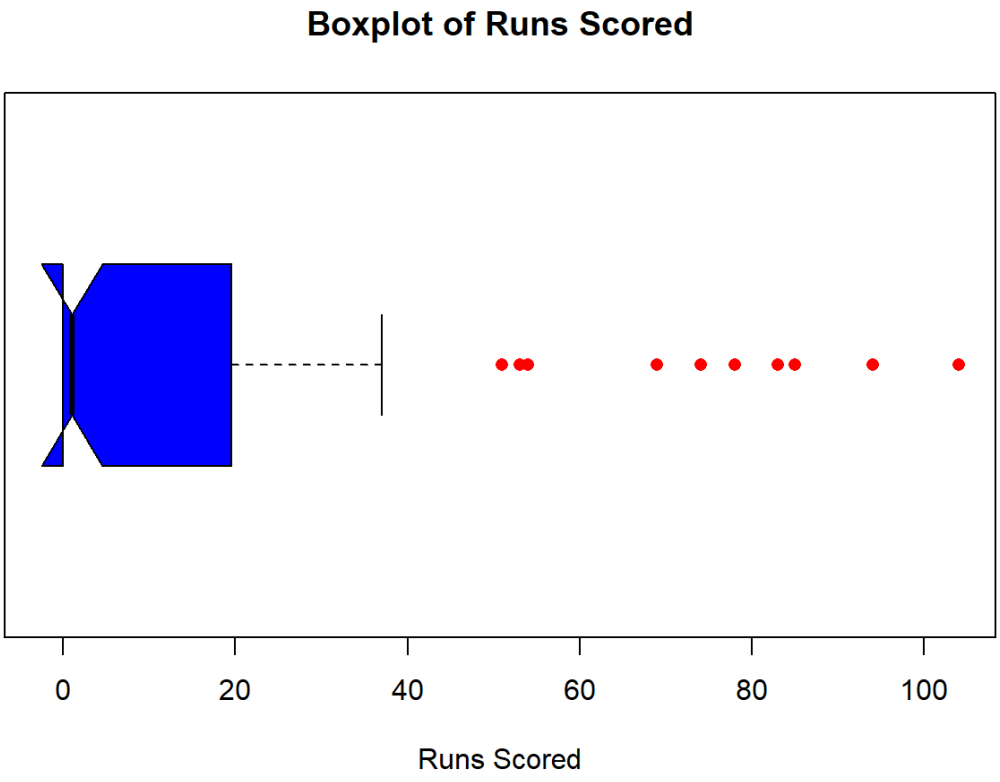
```
# Runs Scored

ggplot(myTeamsData,aes(x=runs))+geom_histogram(color="blue",fill="cyan",bin
s=15)+labs(title="Histogram of Runs Scored",x="Runs Scored",y="Number of Pl
ayers")
```




```
boxplot(myTeamsData$runs, horizontal = T, notch = T,
        xlab = "Runs Scored", col = 'blue', pars = list(outcol = 'red', pch
= 16))
```

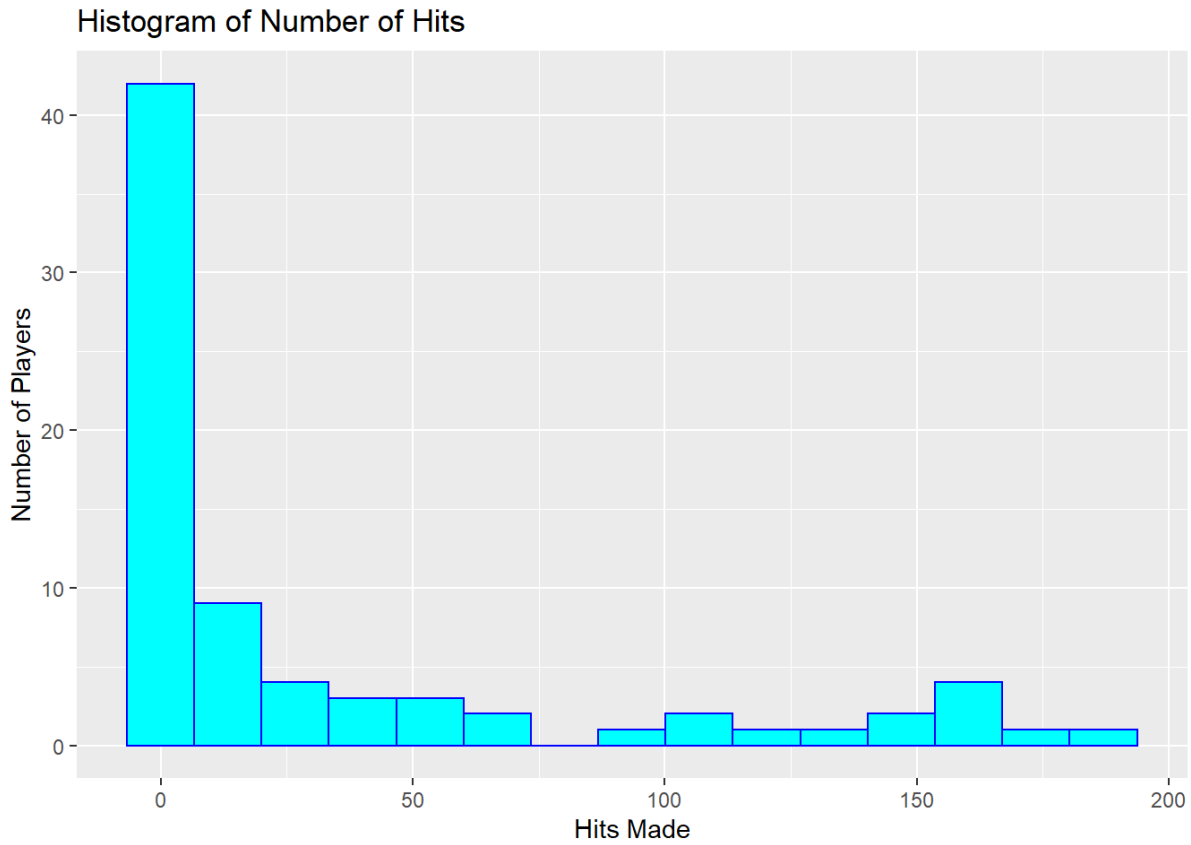
```
## Warning in bxp(list(stats = structure(c(0, 0, 1, 19.5, 37), .Dim = c(5L,
: some
## notches went outside hinges ('box'): maybe set notch=FALSE
```

```
        title("Boxplot of Runs Scored")
```

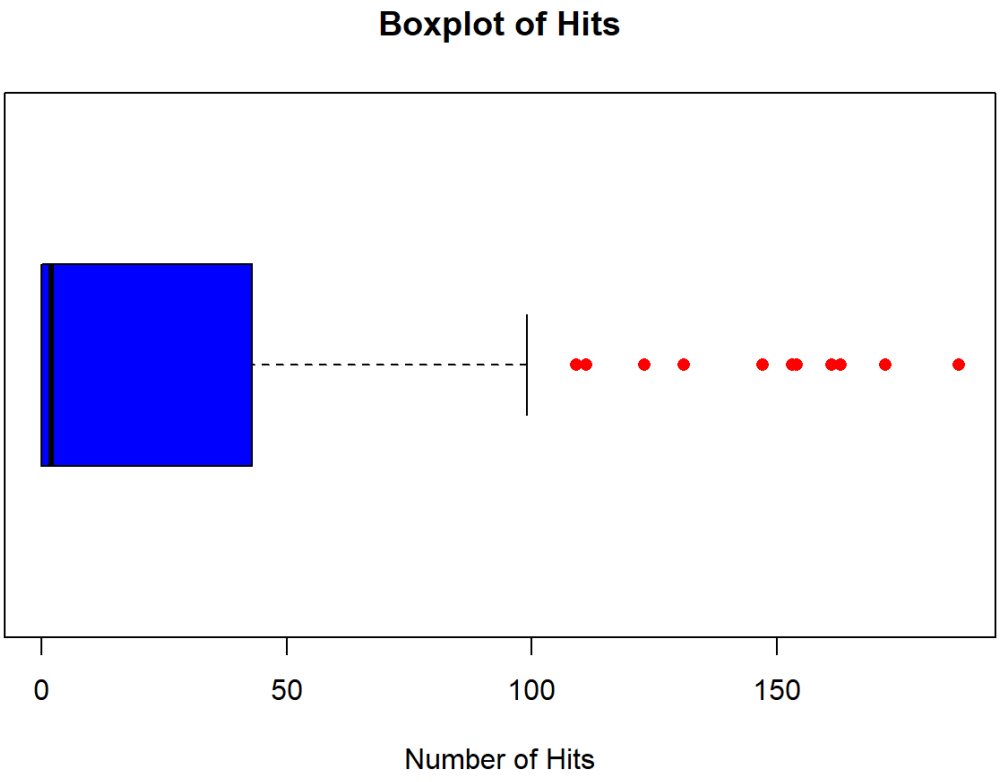


We can notice that most of the players have a score in range from 0 to 30 and only a few players have high runs scored in the season. It is clear fro the boxplot that there are many outliers where players has scored more than 50 runs in the season.

```
# Number of Hits

ggplot(myTeamsData,aes(x=hits)) + geom_histogram(color="blue",fill="cyan",b
ins=15) + labs(title="Histogram of Number of Hits",x="Hits Made",y="Number
of Players")
```

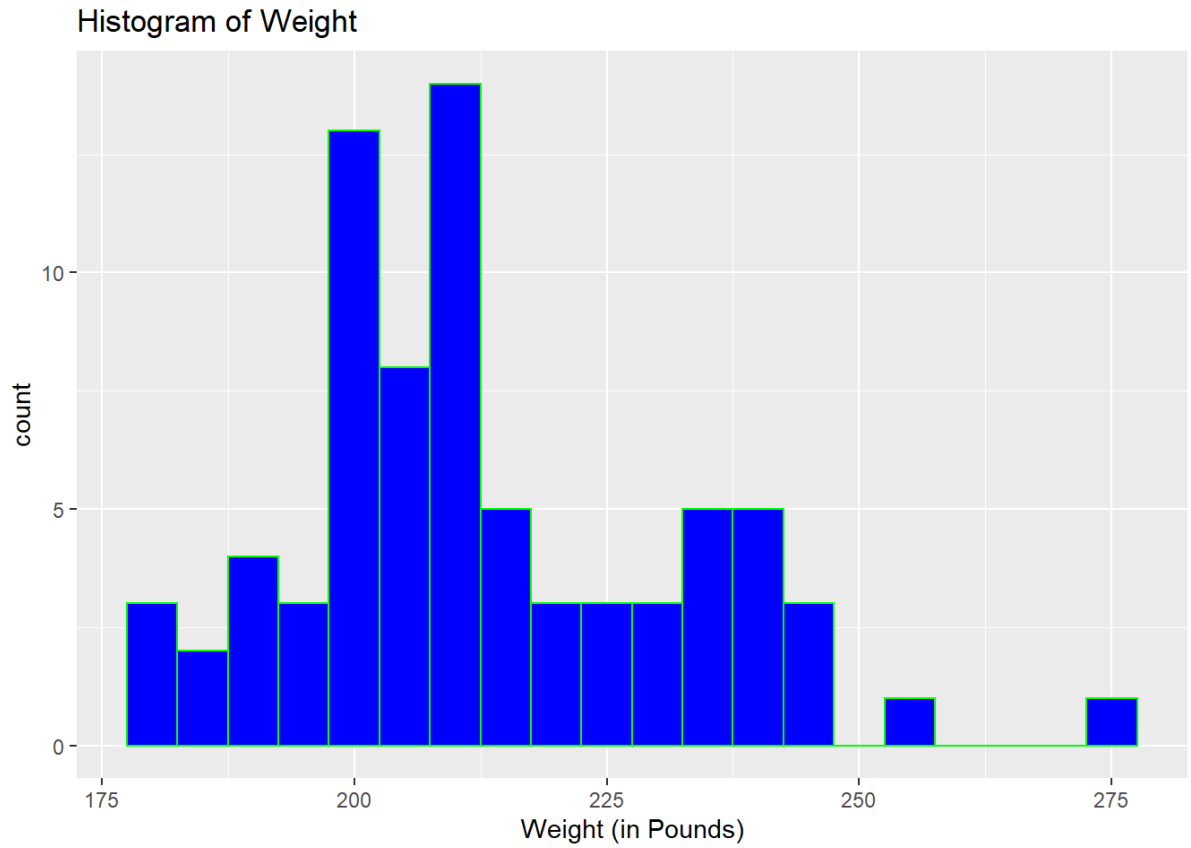



```
boxplot(myTeamsData$hits, horizontal = T, notch = FALSE,
        xlab = "Number of Hits", col = 'blue', pars = list(outcol = 'red',
pch = 16))
        title("Boxplot of Hits")
```

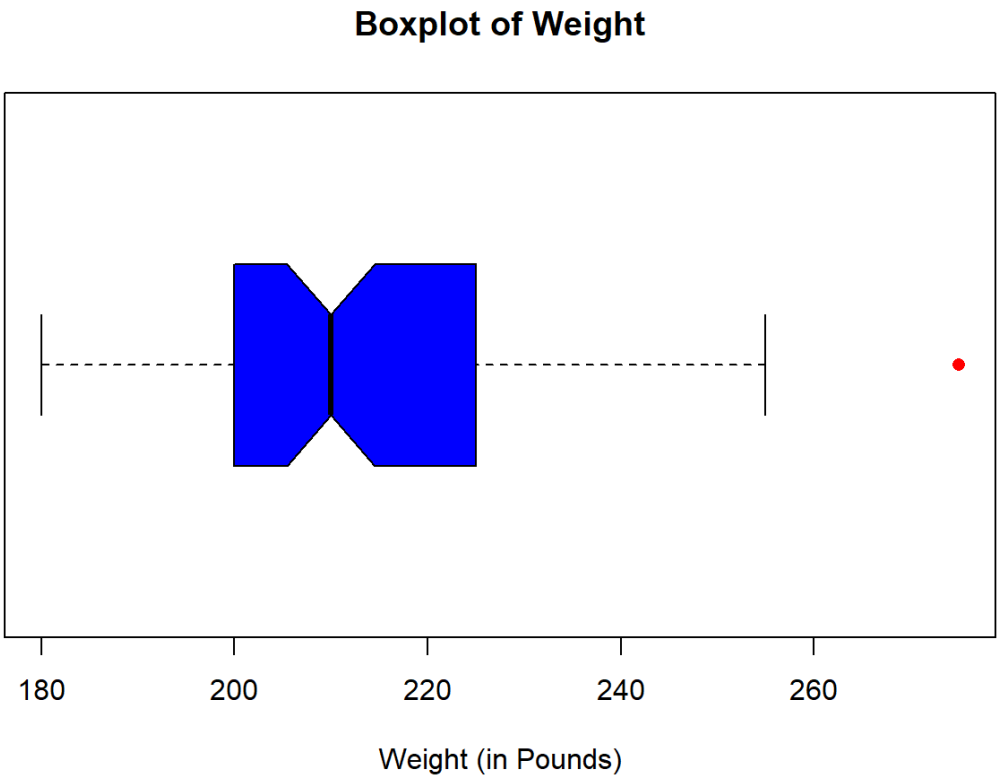


The plots show that the data is skewed to the right with most of the players having 0 hits in the season. On the other hand it can be seen that some of the players have made over 150 hits and are considered as outliers in the boxplot.

```
# Player Weight

ggplot(myTeamsData,aes(x=weight)) + geom_histogram(color="green",fill="blue
",bins=20) + labs(title = "Histogram of Weight",x="Weight (in Pounds)")
```

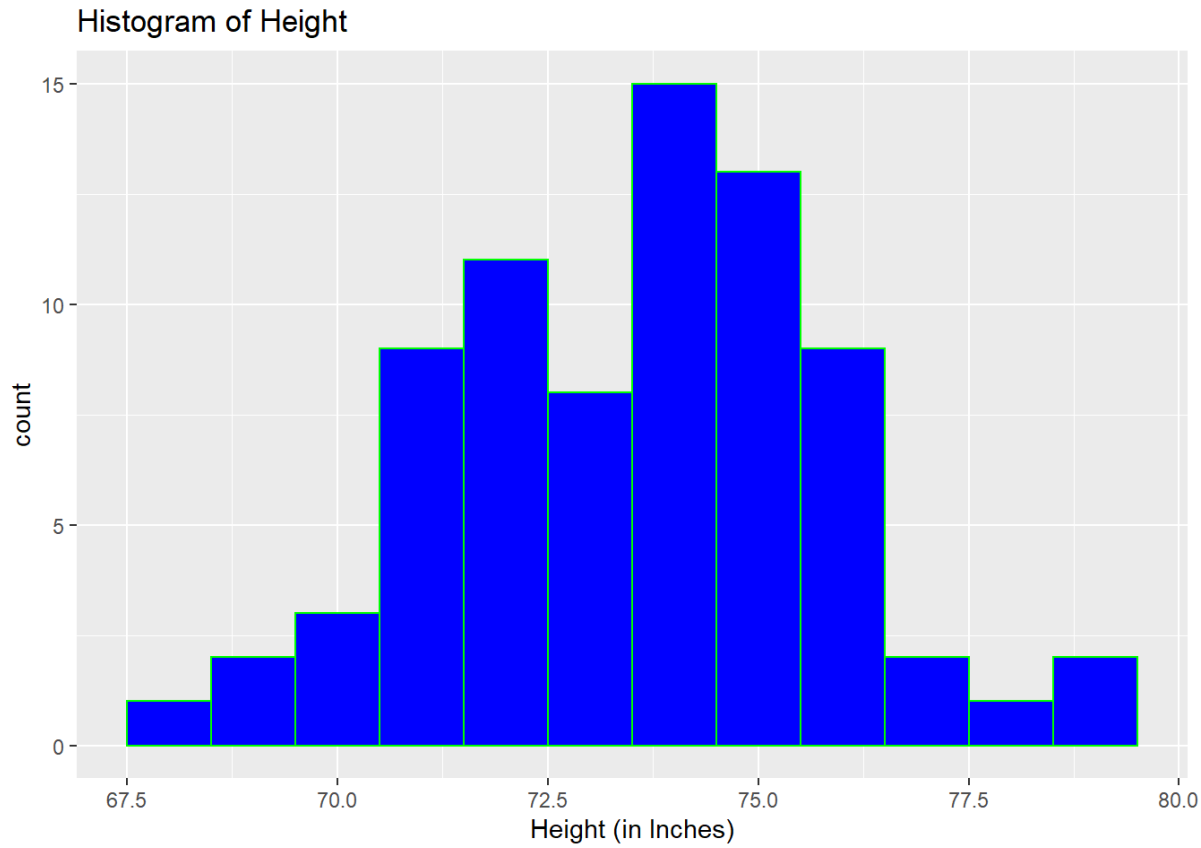



```
boxplot(myTeamsData$weight, horizontal = T, notch = T,
        xlab = "Weight (in Pounds)", col = 'blue', pars = list(outcol = 'red', pch = 16))
        title("Boxplot of Weight")
```

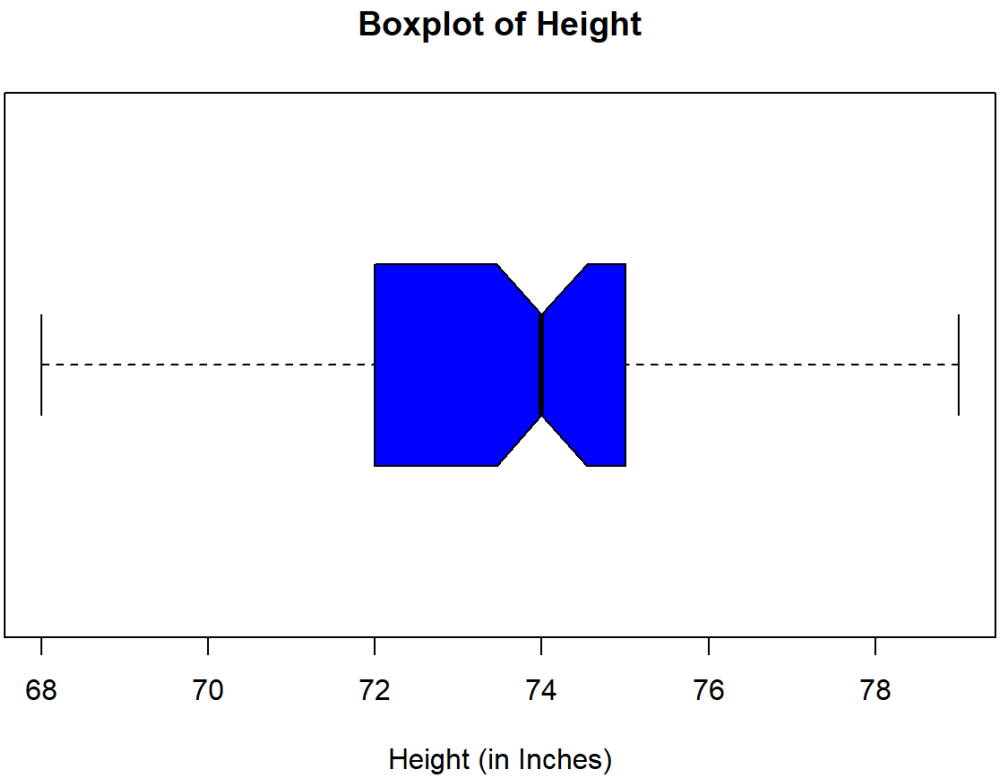


The data for weight looks normally distributed and there is one outlier with a weight of above 270 pounds.

```
# Player Height

ggplot(myTeamsData,aes(x=height)) + geom_histogram(color="green",fill="blue
",bins=12) + labs(title = "Histogram of Height",x="Height (in Inches)")
```

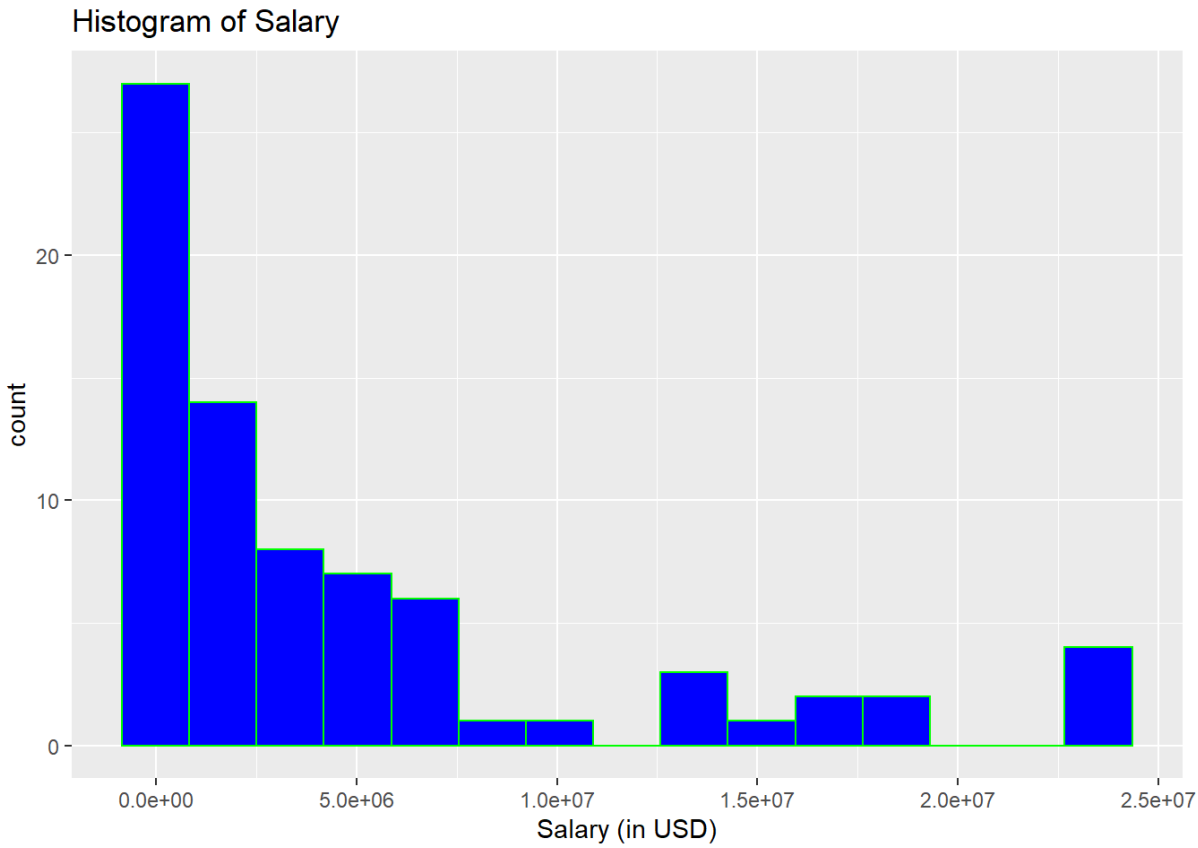



```
boxplot(myTeamsData$height, horizontal = T, notch = T,
        xlab = "Height (in Inches)", col = 'blue', pars = list(outcol = 're
d', pch = 16))
        title("Boxplot of Height")
```

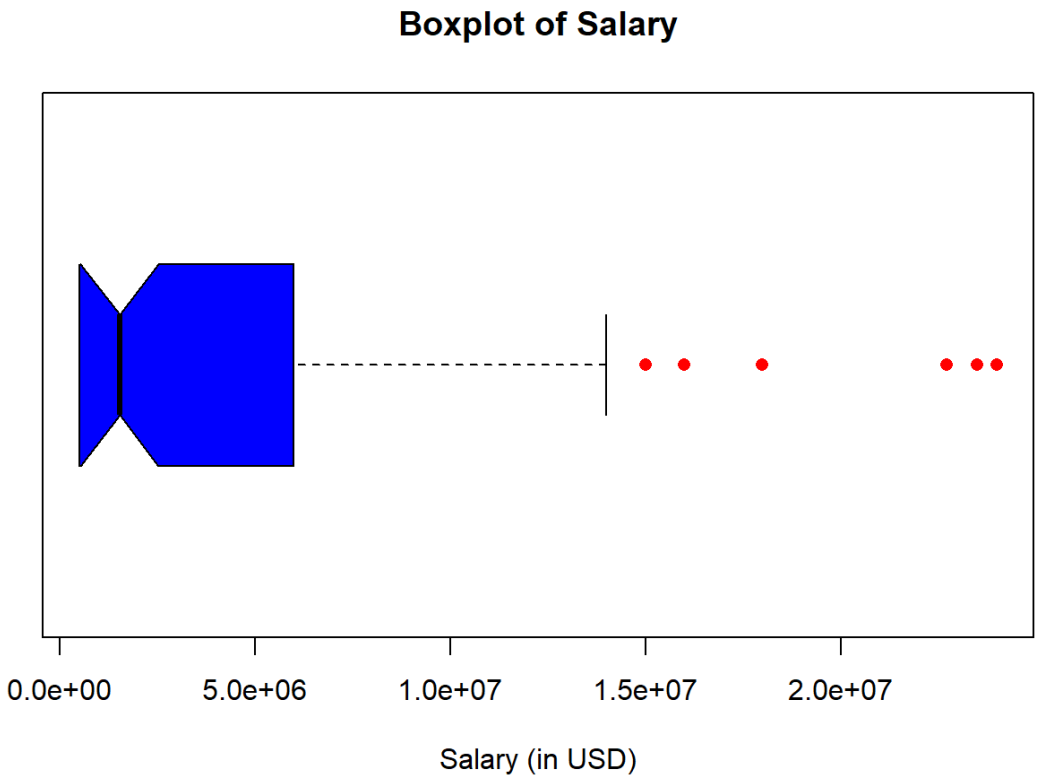


The data for player height is normally distributed and there are no visible outliers.

```
# Player Salary

ggplot(myTeamsData,aes(x=salary)) + geom_histogram(color="green",fill="blue
",bins=15) + labs(title = "Histogram of Salary",x="Salary (in USD)")
```

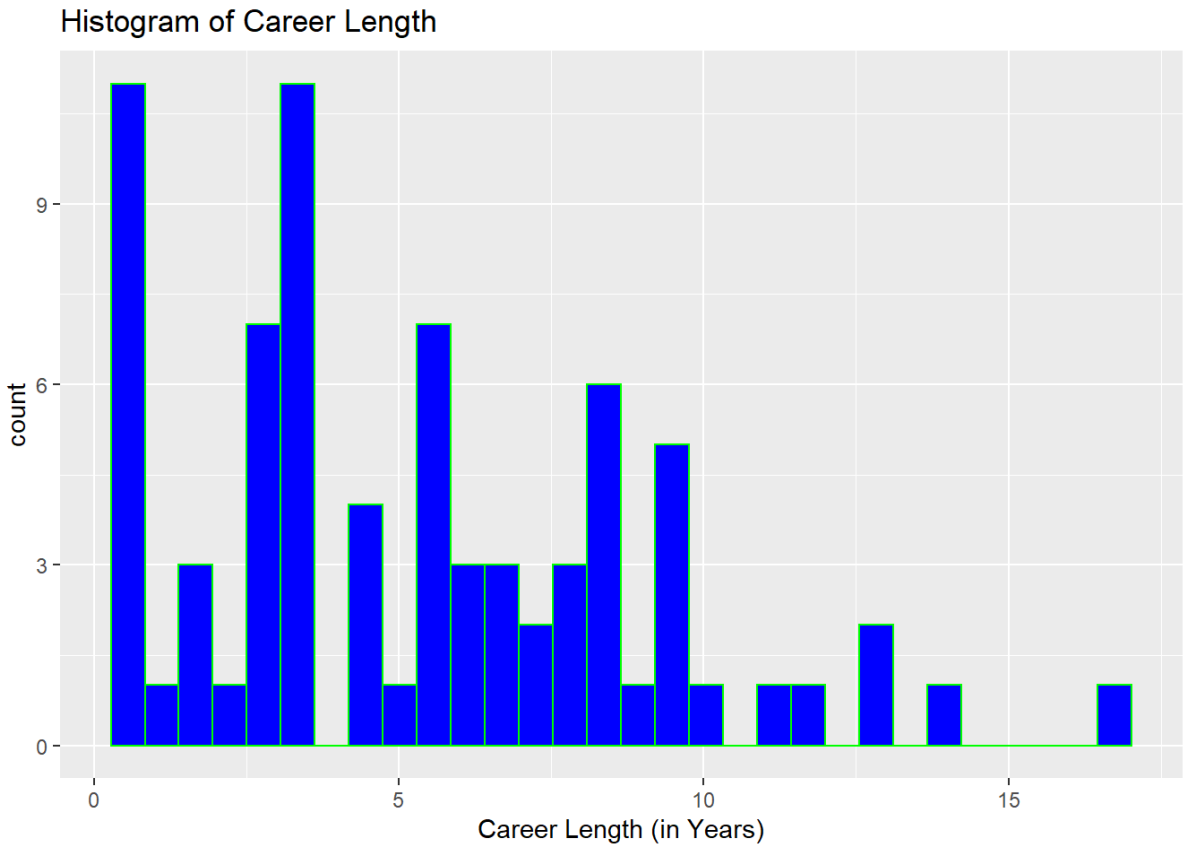



```
boxplot(myTeamsData$salary, horizontal = T, notch = T,
        xlab = "Salary (in USD)", col = 'blue', pars = list(outcol = 'red',
pch = 16))
        title("Boxplot of Salary")
```



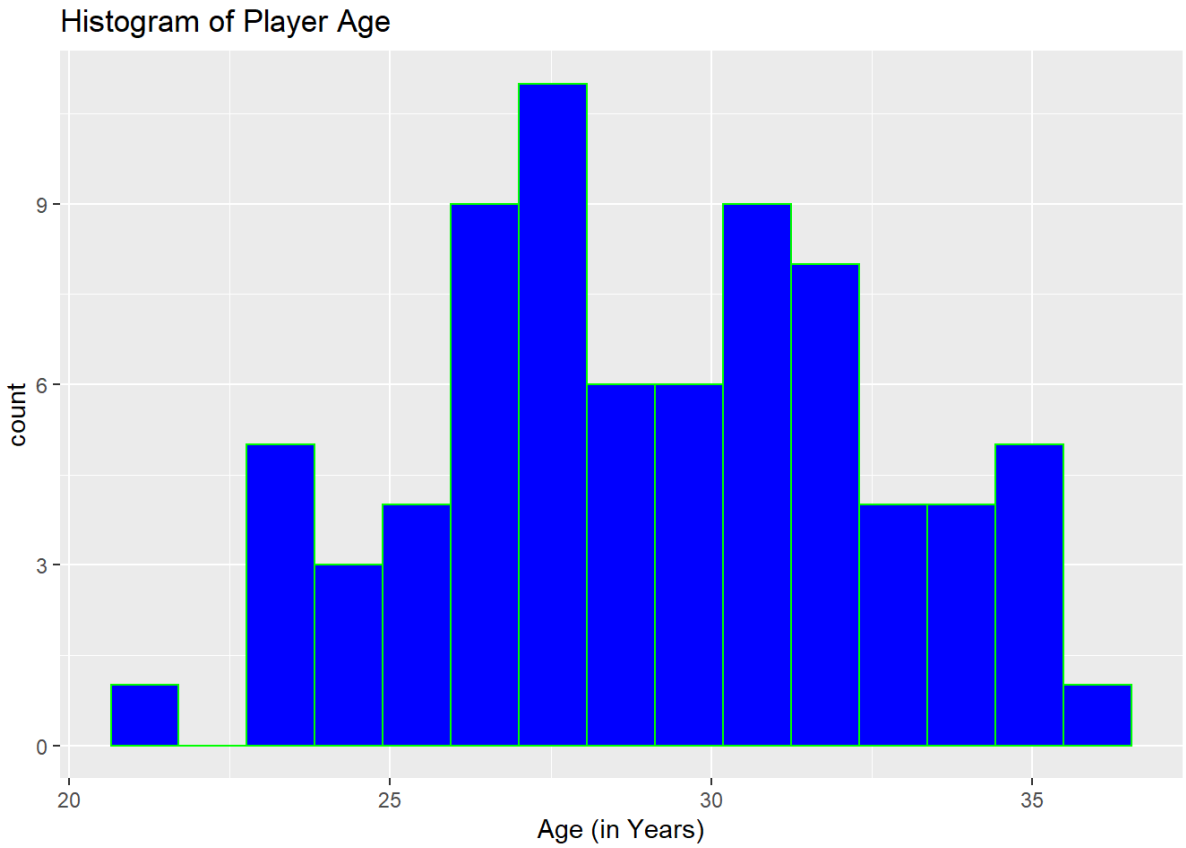

```
# Player Career Length

ggplot(myTeamsData,aes(x=careerLength)) + geom_histogram(color="green",fill
="blue") + labs(title = "Histogram of Career Length",x="Career Length (in Y
ears)")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



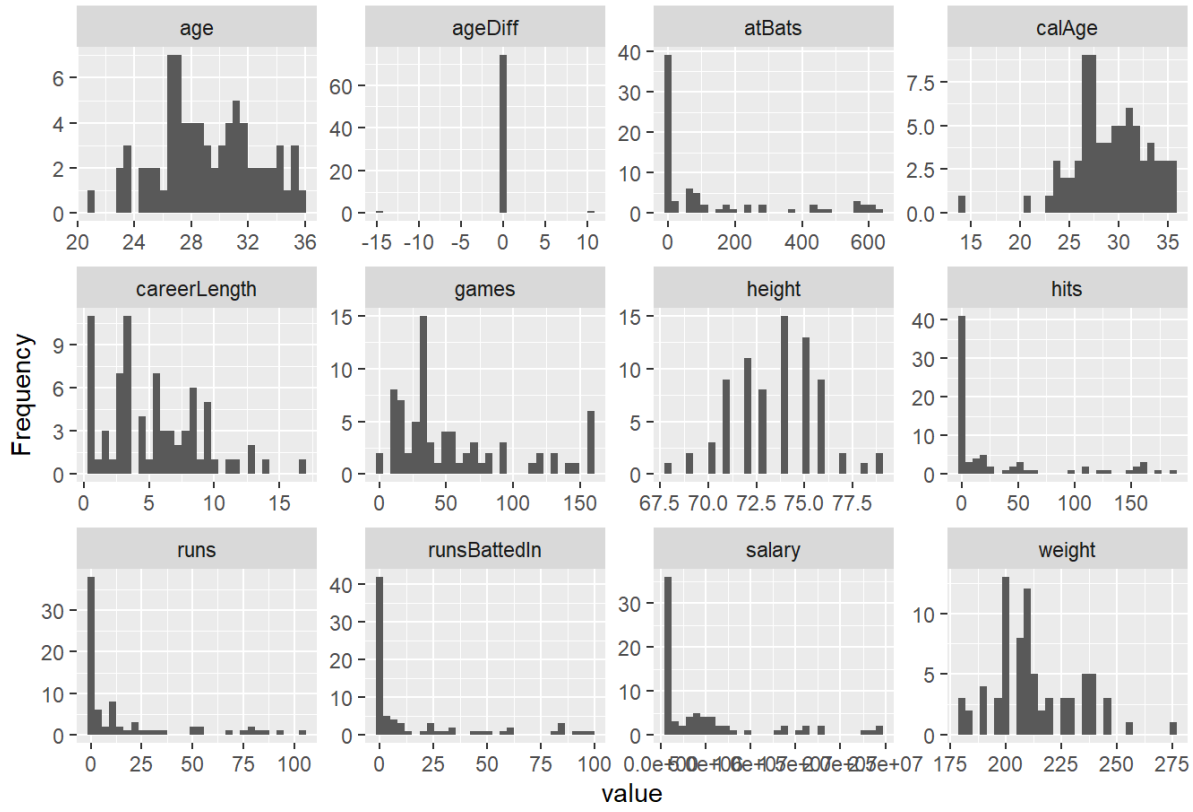
The data for player careerLength is skewed to the right and it also has a few outliers.

```
# Player Age

ggplot(myTeamsData,aes(x=age)) + geom_histogram(color="green",fill="blue",b
ins=15) + labs(title = "Histogram of Player Age",x="Age (in Years)")
```

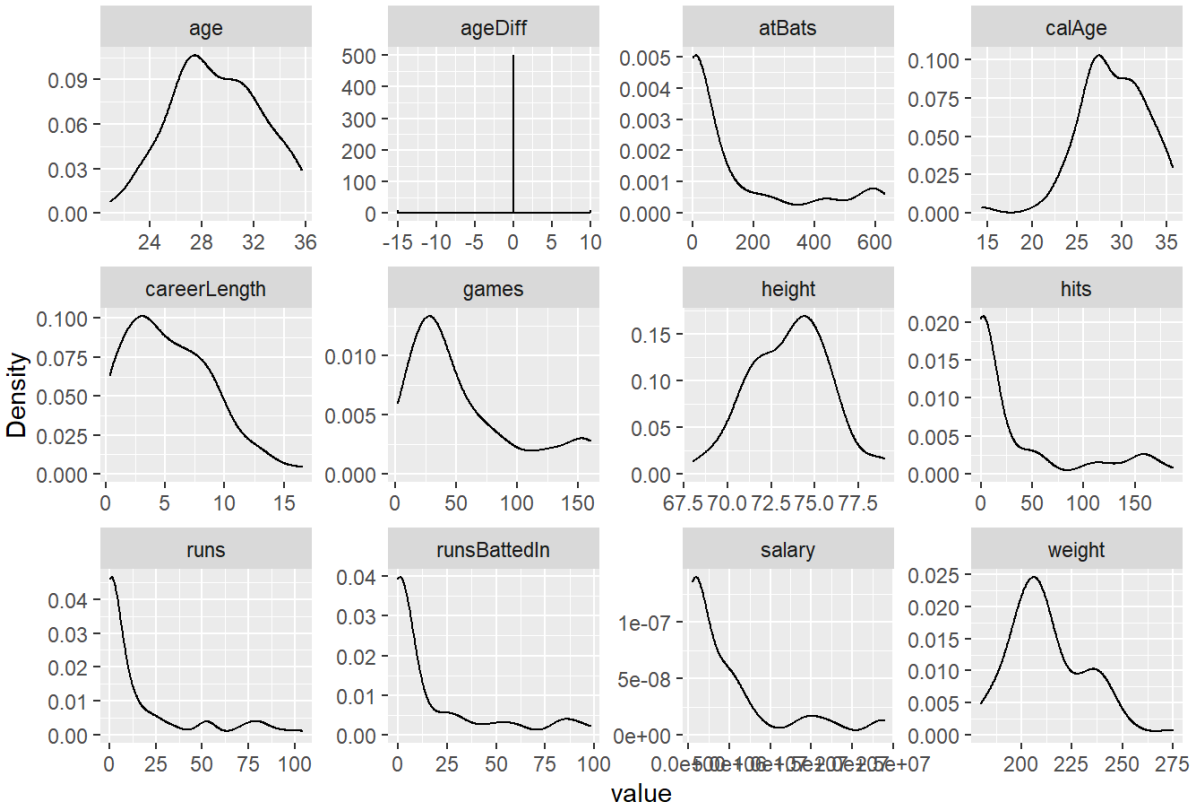


The player age data looks normally distributed in the histogram.

```
# Plots histogram for all continuous variables in the data frame.
plot_histogram(myTeamsData)
```




```
# Plots density plot for all continuous variables in the data frame.
plot_density(myTeamsData)
```



plot_histogram() from DataExplorer allows us to plot histogram for all the continuous variables in a single pane. plot_density() from DataExplorer allows us to plot density for all the continuous variables in a single pane.

```
# Bats
table(myTeamsData$bats)
```

```
##
##  B  L  R
##  3 24 49
```

We have 3 players in our data frame that can bath with both hands, where as 24 can bat with left hand and 49 can bat with right hand only.

```
# Hit Indicator
table(myTeamsData$hitInd)
```

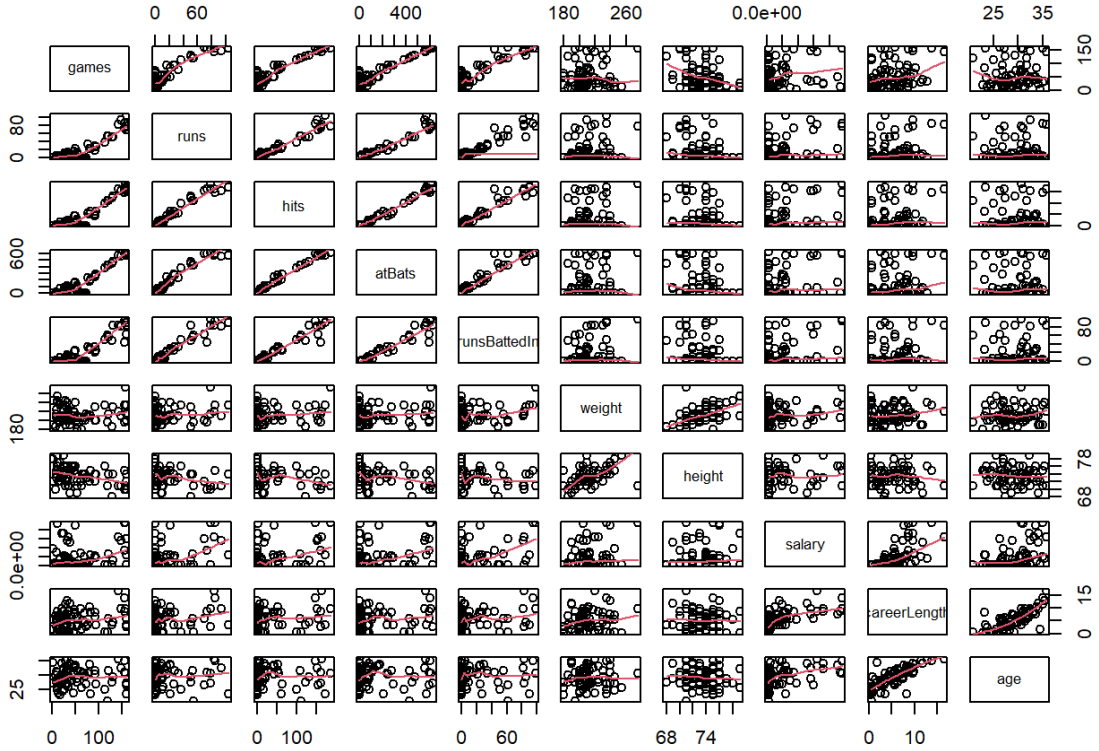
```
##
##  0  1
## 34 42
```

We can see that 34 players did not make a single hit in the season and 42 players did make one or more hits.

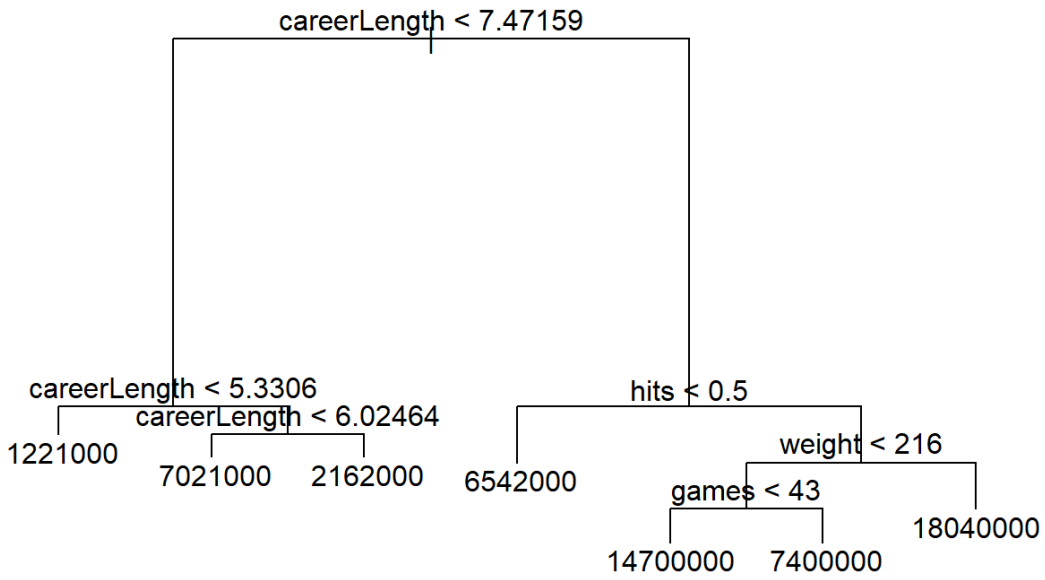
Now we can start with Multivariate Analysis.

```
numMyTeamsData <- myTeamsData[, c(3, 4, 5, 6, 7, 8, 9, 10, 12, 14)]
```

```
pairs(numMyTeamsData, panel = panel.smooth)
```




```
salaryTree <- tree(myTeamsData$salary ~ games + runs + hits + atBats+ runsB
attedIn + weight + height + careerLength + bats + age, data = myTeamsData)

plot(salaryTree)

text(salaryTree)
```

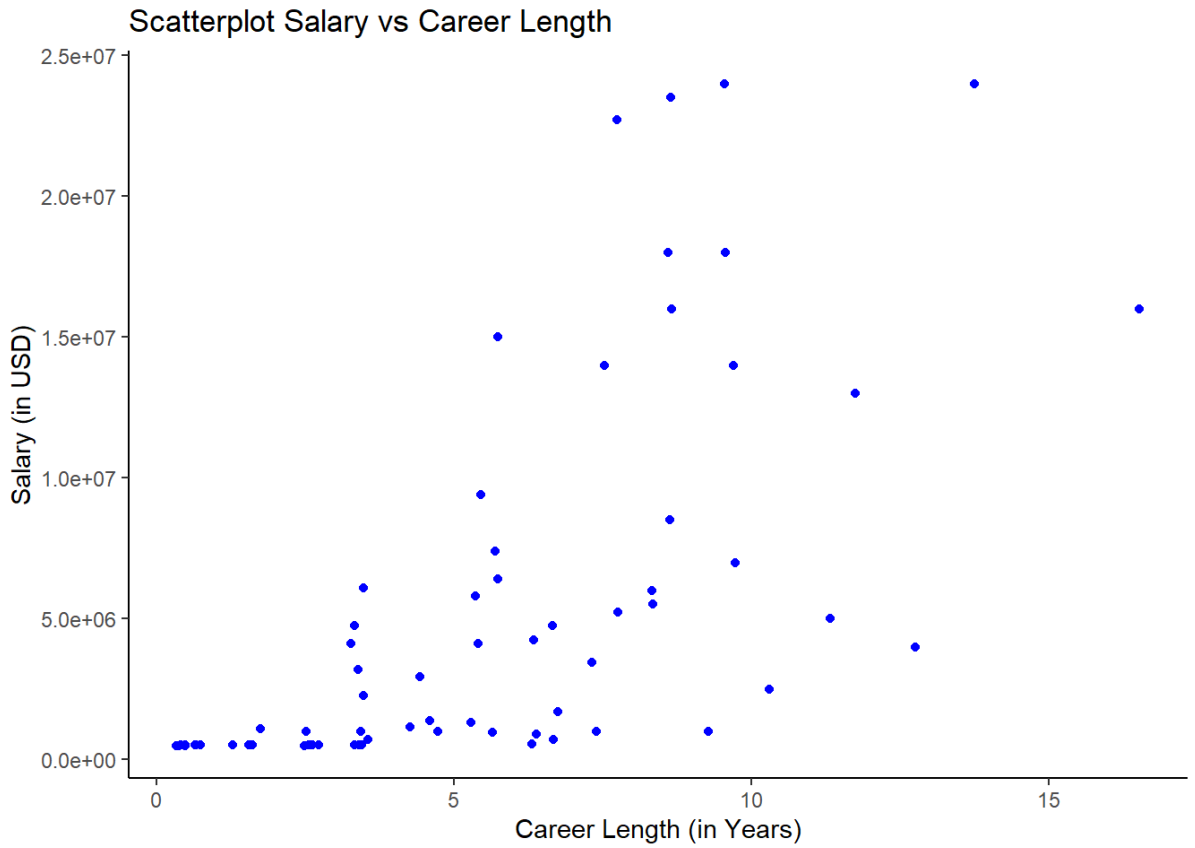


We can see from the pairs() output that some of the variables have very strong positive correlation whereas some variables do not have any strong relationships.

The regression tree also helps us in checking the relation of salary with other variables. This can tell us about the structure of the data, and the top factors affecting the value of our dependent variable i.e. salary.

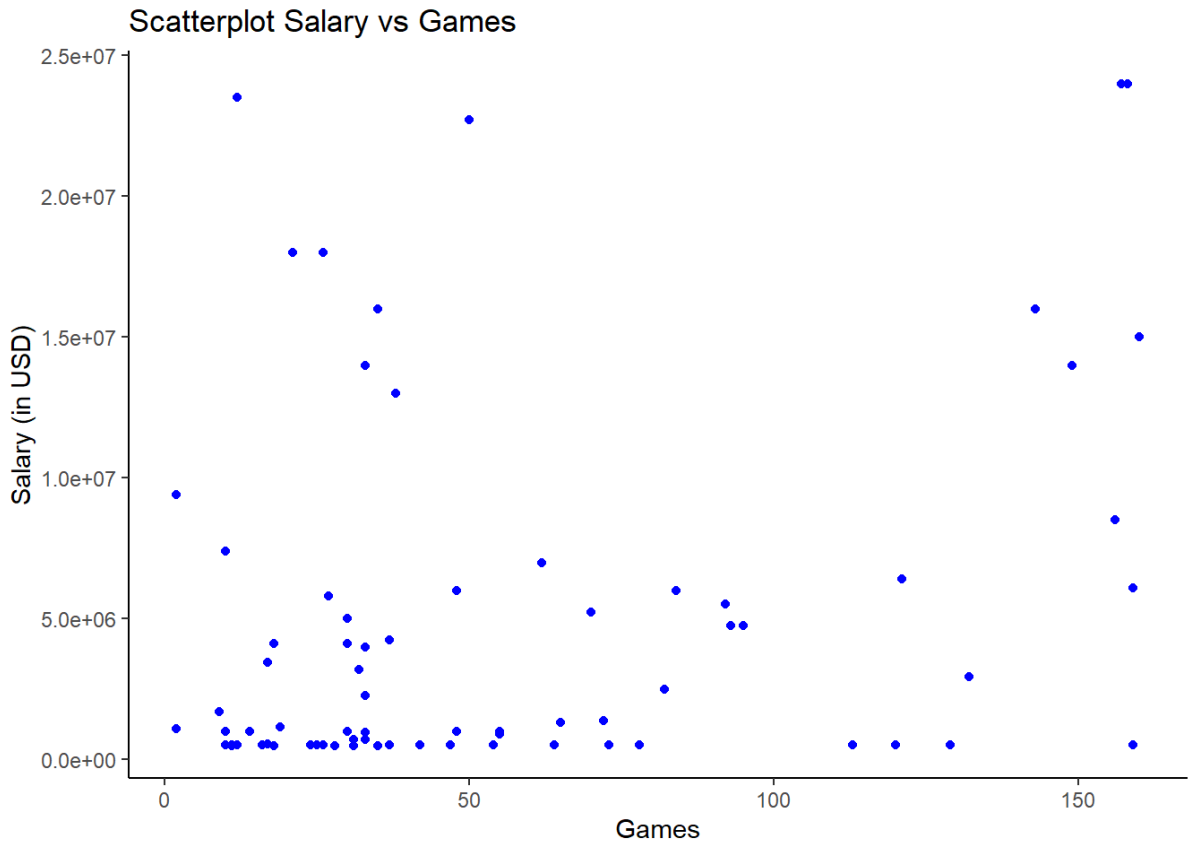
In this case it shows that careerLength is the most important factor affecting salary because it has longer tree branches. Games, hits and weight also has some significance on player salary. We can check these relations with scatter plots.

```
# Scatter Plot for Salary and Career Length

ggplot(myTeamsData,aes(x=careerLength,y=salary)) + geom_point(color="blue")
+ labs(title="Scatterplot Salary vs Career Length",x="Career Length (in Yea
rs)",y="Salary (in USD)")+theme_classic()
```



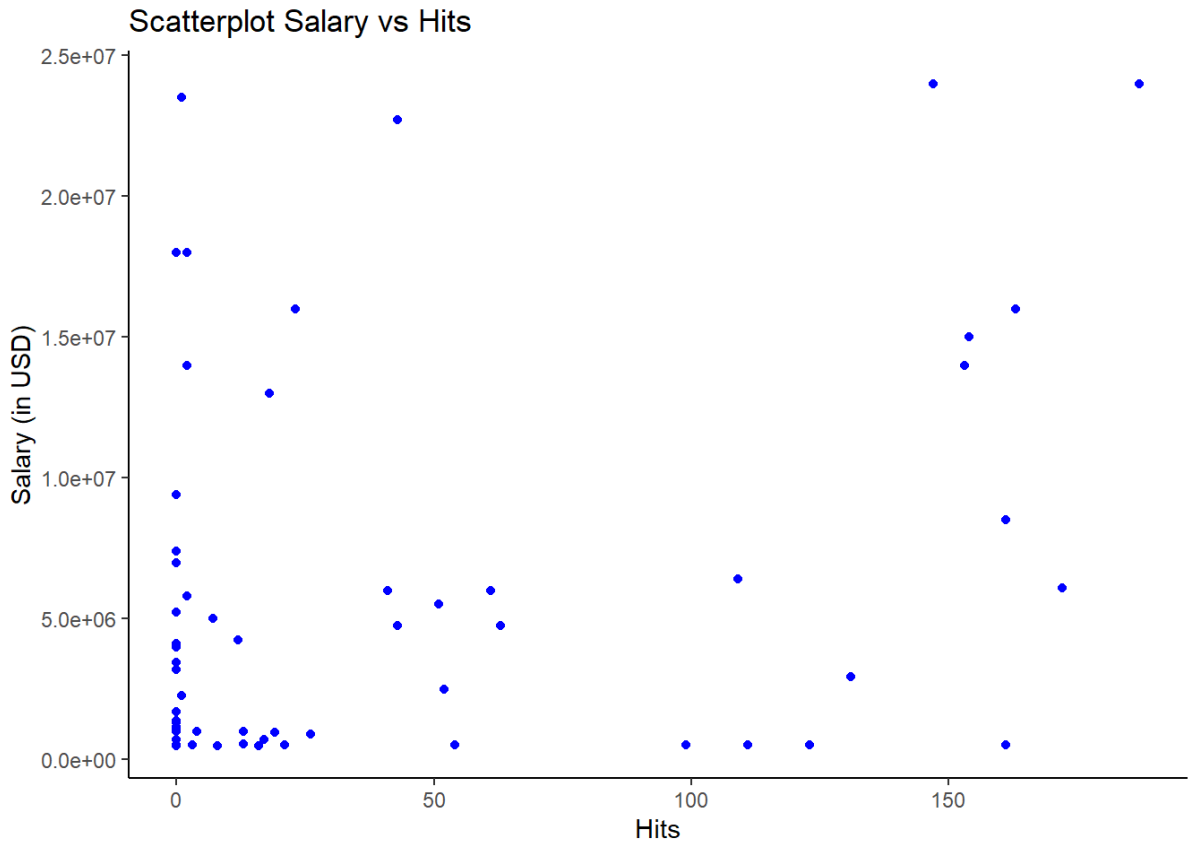
We can see that salary tends to increase with increase in careerLength.

```
# Scatter Plot for Salary and No of Games Played

ggplot(myTeamsData,aes(x=games,y=salary)) + geom_point(color="blue") + labs
(title="Scatterplot Salary vs Games",x="Games",y="Salary (in USD)")+theme_c
lassic()
```



Salary and games seem to have some relation but there are many values which are not in a linear relation.

```
# Scatter Plot for Salary and Hits

ggplot(myTeamsData,aes(x=hits,y=salary)) + geom_point(color="blue") + labs(
title="Scatterplot Salary vs Hits",x="Hits",y="Salary (in USD)")+theme_clas
sic()
```



Salary and hits do not have any clear relation. There is a scatter of players having no hits but good amount of salary.

2.3 Additional insights and issues

While we visualised the data for all the variables, most of them were not normally distributed and we can check that by Shapiro-Wilkin Normality Test. Library moments has a skewness() function which can also be used.

```
# Normality Test for Games
shapiro.test(myTeamsData$games)
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$games
## W = 0.8416, p-value = 1.515e-07
# Normality Test for Runs
shapiro.test(myTeamsData$runs)
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$runs
```

```
## W = 0.65671, p-value = 4.831e-12
```

```
# Normality Test for Hits
shapiro.test(myTeamsData$hits)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$hits
## W = 0.65778, p-value = 5.065e-12
```

```
# Normality Test for atBats
shapiro.test(myTeamsData$atBats)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$atBats
## W = 0.68008, p-value = 1.393e-11
```

```
# Normality Test for runsBattedIn
shapiro.test(myTeamsData$runsBattedIn)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$runsBattedIn
## W = 0.64691, p-value = 3.144e-12
```

```
# Normality Test for Weight
shapiro.test(myTeamsData$weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  myTeamsData$weight
## W = 0.9485, p-value = 0.003829
```

```
# We can also use skewness() from moments library. An integer away from 0 r
epresents skewness. Far the value, stronger the skewness
# Normality Test for Height
skewness(myTeamsData$height)
```

```
## [1] -0.0239857
```

```
# Skewness Test for Salary
skewness(myTeamsData$salary)
```

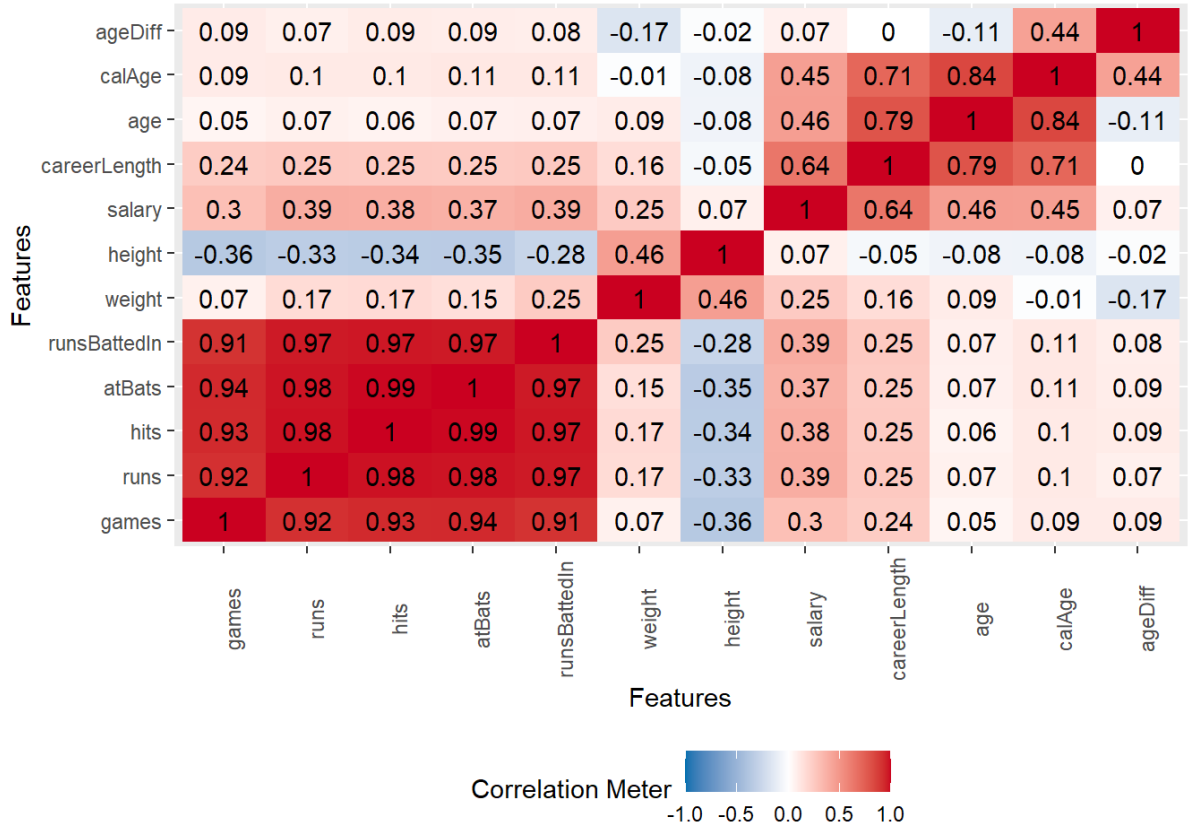
```
## [1] 1.711006
```

```
# Skewness Test for careerLength
skewness(myTeamsData$careerLength)
## [1] 0.638329
# Skewness Test for Age
skewness(myTeamsData$age)
## [1] 0.01581288
```

The output of these test results approve what we saw visually that distribution for Games, Runs, Hits, atBats, runsBattedIn, salary is not normally distributed, whereas age and height are normally distributed. Weight and careerLength are not really normally distributed.

To curb this issue we will need to apply some transformations to our data so that a meaningful analysis could be possible.

```
plot_correlation(myTeamsData, type = 'continuous')
```



With plot_correlation() we can check the correlation between all the variables and type='continuous' species that we we want to check the correlation between continuous variables only. A benefit of using this function is that it shows the strength of the correlation and differentiate it based on different colors from the Correlation Meter.

Variables such as Games, Runs, Hits, atBats, runsBattedIn are highly correlated and while building a model these should be kept in mind. This issue was already raised by the tree model where only careerLength, hits, weight and games had a significant effect on salary and all other were not included in the tree model.

3. Modelling

3.1 Build a model for player salary

We need to build a suitable model for player salary, so salary will be our dependent/response variable. The model should be able to predict the salary of a player based on different independent variables such as career length, runs scored, hits etc.

Our dependent variable is continuous and we have a combination of both numerical and categorical independent variables. To build a suitable model we can use multiple regression method here. In our analysis we saw that data for most of the independent variables was not normally distributed. Even our dependent variable salary was skewed to the right. The correlation between some independent variables also pointed that we should not include all those having a strong correlation.

As seen earlier salary was skewed to the right we can try to use some transformations on the dependent variable. Also we can try to include the interactions and polynomial terms to our model so that we can find the best model to predict a player salary.

We will build a model below. We can start with the maximal model and then apply step() to reach to a minimal adequate model.

```
#Let us start with maximal model derived from our tree model
salaryModel1 <- lm(salary ~ games * weight * careerLength * hits, data = myTeamsData)
summary(salaryModel1)
```

```
##
## Call:
## lm(formula = salary ~ games * weight * careerLength * hits, data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7751247 -2183174  -446075  1721902 12640081
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1.423e+07  1.777e+07  -0.801   0.4263
## games                          1.467e+05  6.132e+05   0.239   0.8118
## weight                         6.160e+04  8.202e+04   0.751   0.4556
## careerLength                   4.021e+06  3.638e+06   1.105   0.2735
## hits                          -5.294e+05  1.574e+06  -0.336   0.7378
## games:weight                  -7.937e+02  2.962e+03  -0.268   0.7897
## games:careerLength             5.833e+04  1.119e+05   0.521   0.6041
## weight:careerLength           -1.197e+04  1.679e+04  -0.713   0.4786
```

```
## games:hits                       5.761e+03  1.072e+04   0.537   0.5932
## weight:hits                      2.737e+03  7.383e+03   0.371   0.7121
## careerLength:hits               -4.249e+05  2.482e+05  -1.712   0.0921 .
## games:weight:careerLength       -3.173e+02  5.227e+02  -0.607   0.5461
## games:weight:hits               -2.720e+01  4.949e+01  -0.550   0.5846
## games:careerLength:hits          1.883e+03  1.586e+03   1.187   0.2399
## weight:careerLength:hits         1.937e+03  1.144e+03   1.694   0.0954 .
## games:weight:careerLength:hits -8.300e+00  7.274e+00  -1.141   0.2584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4255000 on 60 degrees of freedom
## Multiple R-squared:  0.6475, Adjusted R-squared:  0.5594
## F-statistic: 7.347 on 15 and 60 DF,  p-value: 7.883e-09
```

Although we have a r-squared value of almost 65% and a significant F-statistic but none of the coefficient is significant.

Let's apply step() function to remove insignificant terms.

```
salaryModel2 <- step(salaryModel1)
## Start:  AIC=2334.1
## salary ~ games * weight * careerLength * hits
##
##                                   Df  Sum of Sq        RSS    AIC
## - games:weight:careerLength:hits  1 2.3572e+13 1.1099e+15 2333.7
## <none>                                          1.0863e+15 2334.1
##
## Step:  AIC=2333.73
## salary ~ games + weight + careerLength + hits + games:weight +
##     games:careerLength + weight:careerLength + games:hits + weight:hits
+
##     careerLength:hits + games:weight:careerLength + games:weight:hits +
##     games:careerLength:hits + weight:careerLength:hits
##
##                               Df  Sum of Sq        RSS    AIC
## - games:weight:careerLength  1 7.2108e+12 1.1171e+15 2332.2
## - games:careerLength:hits    1 8.1145e+12 1.1180e+15 2332.3
## <none>                                    1.1099e+15 2333.7
## - weight:careerLength:hits   1 5.1073e+13 1.1609e+15 2335.2
```

```
## - games:weight:hits          1 1.8026e+14 1.2901e+15 2343.2
##
## Step:  AIC=2332.23
## salary ~ games + weight + careerLength + hits + games:weight +
##     games:careerLength + weight:careerLength + games:hits + weight:hits
+
##     careerLength:hits + games:weight:hits + games:careerLength:hits +
##     weight:careerLength:hits
##
##                             Df  Sum of Sq        RSS    AIC
## - games:careerLength:hits    1 7.3342e+12 1.1244e+15 2330.7
## <none>                                    1.1171e+15 2332.2
## - weight:careerLength:hits   1 1.4972e+14 1.2668e+15 2339.8
## - games:weight:hits          1 1.8287e+14 1.2999e+15 2341.8
##
## Step:  AIC=2330.72
## salary ~ games + weight + careerLength + hits + games:weight +
##     games:careerLength + weight:careerLength + games:hits + weight:hits
+
##     careerLength:hits + games:weight:hits + weight:careerLength:hits
##
##                             Df  Sum of Sq        RSS    AIC
## - games:careerLength         1 1.3122e+13 1.1375e+15 2329.6
## <none>                                    1.1244e+15 2330.7
## - weight:careerLength:hits   1 1.7442e+14 1.2988e+15 2339.7
## - games:weight:hits          1 1.8123e+14 1.3056e+15 2340.1
##
## Step:  AIC=2329.6
## salary ~ games + weight + careerLength + hits + games:weight +
##     weight:careerLength + games:hits + weight:hits + careerLength:hits +
##     games:weight:hits + weight:careerLength:hits
##
##                             Df  Sum of Sq        RSS    AIC
## <none>                                    1.1375e+15 2329.6
## - weight:careerLength:hits   1 1.6760e+14 1.3051e+15 2338.1
## - games:weight:hits          1 1.8716e+14 1.3247e+15 2339.2
summary(salaryModel2)
##
```

```
## Call:
## lm(formula = salary ~ games + weight + careerLength + hits +
##     games:weight + weight:careerLength + games:hits + weight:hits +
##     careerLength:hits + games:weight:hits + weight:careerLength:hits,
##     data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7407317 -2152187  -591787  1886813 13705109
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.587e+07  1.314e+07  -1.208  0.23157
## games                     4.745e+05  3.747e+05   1.266  0.20999
## weight                    7.792e+04  6.080e+04   1.282  0.20457
## careerLength              4.283e+06  1.830e+06   2.340  0.02242 *
## hits                     -2.357e+06  7.662e+05  -3.076  0.00308 **
## games:weight             -2.589e+03  1.812e+03  -1.429  0.15781
## weight:careerLength      -1.498e+04  8.464e+03  -1.770  0.08156 .
## games:hits                1.708e+04  5.192e+03   3.290  0.00163 **
## weight:hits               1.116e+04  3.601e+03   3.100  0.00288 **
## careerLength:hits        -1.072e+05  3.525e+04  -3.040  0.00342 **
## games:weight:hits        -7.771e+01  2.395e+01  -3.245  0.00187 **
## weight:careerLength:hits  4.892e+02  1.593e+02   3.071  0.00313 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4216000 on 64 degrees of freedom
## Multiple R-squared:  0.6309, Adjusted R-squared:  0.5674
## F-statistic: 9.944 on 11 and 64 DF,  p-value: 3.154e-10
```

The r-squared values is still good and F-statistic is also significant. Some of the coefficients are now significant. We can see that it is a very complex model as it contains so many interactions.

But according to Parsimony : Parsimonious models are simple models with great explanatory predictive power. They explain data with a minimum number of parameters, or predictor variables.The idea behind parsimonious models stems from Occam's razor, or "the law of briefness" (sometimes called lex parsimoniae in Latin). The law states that you should use no more "things" than necessary; In the case of parsimonious models, those "things" are parameters.

We can try to build a model which is less complex with less interactions.

```
salaryModel3 <- lm(salary ~ games + weight + careerLength + hits, data = myTeamsData)
summary(salaryModel3)
```

```
##
## Call:
## lm(formula = salary ~ games + weight + careerLength + hits, data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7947455 -2636586  -349381  1613007 15246357
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5063710    6652136  -0.761   0.4490
## games          -60365      33499  -1.802   0.0758 .
## weight          25756      30134   0.855   0.3956
## careerLength  1005397     152225   6.605 6.18e-09 ***
## hits            74438      28680   2.595   0.0115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4655000 on 71 degrees of freedom
## Multiple R-squared:  0.5008, Adjusted R-squared:  0.4726
## F-statistic:  17.8 on 4 and 71 DF,  p-value: 3.662e-10
```

This model gives us R-squared values of 50% with a good F-statistic value and few of the coefficients are significant. We can apply step() to check if this model can be simplified.

```
salaryModel4 <- step(salaryModel3)
```

```
## Start:  AIC=2338.55
## salary ~ games + weight + careerLength + hits
##
##                Df  Sum of Sq        RSS    AIC
## - weight        1 1.5830e+13 1.5543e+15 2337.3
## <none>                       1.5385e+15 2338.6
## - games         1 7.0364e+13 1.6089e+15 2339.9
## - hits          1 1.4597e+14 1.6845e+15 2343.4
## - careerLength  1 9.4524e+14 2.4837e+15 2372.9
```

```
##
## Step:  AIC=2337.33
## salary ~ games + careerLength + hits
##
##                Df  Sum of Sq        RSS    AIC
## <none>                       1.5543e+15 2337.3
## - games         1 9.6336e+13 1.6507e+15 2339.9
## - hits          1 1.9334e+14 1.7477e+15 2344.2
## - careerLength  1 9.9745e+14 2.5518e+15 2373.0
```

```
summary(salaryModel4)
```

```
##
## Call:
## lm(formula = salary ~ games + careerLength + hits, data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7309483 -2754797  -267320  1422397 14887243
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    506920    1329093   0.381  0.70403
## games          -68044      32211  -2.112  0.03812 *
## careerLength  1023141     150520   6.797 2.61e-09 ***
## hits            81759      27320   2.993  0.00379 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4646000 on 72 degrees of freedom
## Multiple R-squared:  0.4956, Adjusted R-squared:  0.4746
## F-statistic: 23.58 on 3 and 72 DF,  p-value: 9.722e-11
```

The model got simplified by removing some insignificant coefficients but the R-squared value also went down from 50% to 46%.

The above two models looked fine, they do have some issues so we will try some more models. After looking at the tree model we can assume that some polynomial relation is possible. Therefore our most complicated model will include some quadratic terms and interactions. Also as the dependent variable is skewed so we can use transformation for that too.

```
salaryModel5 <- lm(salary ~ games * weight * hits * careerLength + I(games^
2) + I(weight^2) + I(hits^2) + I(careerLength^2), data = myTeamsData)
```

```
summary(salaryModel5)
```

```
##
## Call:
## lm(formula = salary ~ games * weight * hits * careerLength +
##     I(games^2) + I(weight^2) + I(hits^2) + I(careerLength^2),
##     data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8687985 -1706015     5243  1780068 10837434
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -2.090e+07  8.051e+07  -0.260  0.79616
## games                          -3.430e+05  7.109e+05  -0.482  0.63138
## weight                          1.718e+05  7.142e+05   0.241  0.81077
## hits                            2.184e+06  1.861e+06   1.174  0.24545
## careerLength                   -2.516e+06  4.025e+06  -0.625  0.53444
## I(games^2)                     -6.916e+02  1.525e+03  -0.454  0.65194
## I(weight^2)                    -3.950e+02  1.572e+03  -0.251  0.80253
## I(hits^2)                      -5.036e+03  1.637e+03  -3.077  0.00324 **
## I(careerLength^2)               3.460e+04  3.928e+04   0.881  0.38215
## games:weight                    1.913e+03  3.433e+03   0.557  0.57962
## games:hits                     -7.814e+03  1.154e+04  -0.677  0.50099
## weight:hits                    -1.143e+04  8.968e+03  -1.274  0.20785
## games:careerLength              3.186e+05  1.385e+05   2.299  0.02523 *
## weight:careerLength             2.015e+04  1.844e+04   1.093  0.27924
## hits:careerLength              -9.947e+05  3.140e+05  -3.168  0.00249 **
## games:weight:hits               7.024e+01  5.724e+01   1.227  0.22490
## games:weight:careerLength      -1.636e+03  6.673e+02  -2.452  0.01733 *
## games:hits:careerLength         4.581e+03  1.814e+03   2.525  0.01444 *
## weight:hits:careerLength        4.695e+03  1.475e+03   3.183  0.00238 **
## games:weight:hits:careerLength -2.094e+01  8.376e+00  -2.500  0.01538 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3934000 on 56 degrees of freedom
```

```
## Multiple R-squared:  0.7187, Adjusted R-squared:  0.6233
## F-statistic:  7.53 on 19 and 56 DF,  p-value: 1.592e-09
```

This model has a very good R-squared value but only few of the coefficients are significant and its too complex. So let us use step() to simplify it and remove unwanted terms.

```
salaryModel6 <- step(salaryModel5)
## Start:  AIC=2324.96
## salary ~ games * weight * hits * careerLength + I(games^2) +
##     I(weight^2) + I(hits^2) + I(careerLength^2)
##
##                                  Df  Sum of Sq        RSS    AIC
## - I(weight^2)                     1 9.7729e+11 8.6787e+14 2323.0
## - I(games^2)                      1 3.1837e+12 8.7007e+14 2323.2
## - I(careerLength^2)               1 1.2012e+13 8.7890e+14 2324.0
## <none>                                         8.6689e+14 2325.0
## - games:weight:hits:careerLength  1 9.6749e+13 9.6364e+14 2331.0
## - I(hits^2)                       1 1.4654e+14 1.0134e+15 2334.8
##
## Step:  AIC=2323.04
## salary ~ games + weight + hits + careerLength + I(games^2) +
##     I(hits^2) + I(careerLength^2) + games:weight + games:hits +
##     weight:hits + games:careerLength + weight:careerLength +
##     hits:careerLength + games:weight:hits + games:weight:careerLength +
##     games:hits:careerLength + weight:hits:careerLength + games:weight:hi
ts:careerLength
##
##                                  Df  Sum of Sq        RSS    AIC
## - I(games^2)                      1 3.6798e+12 8.7155e+14 2321.4
## - I(careerLength^2)               1 1.2184e+13 8.8005e+14 2322.1
## <none>                                         8.6787e+14 2323.0
## - games:weight:hits:careerLength  1 1.2330e+14 9.9116e+14 2331.1
## - I(hits^2)                       1 1.7058e+14 1.0384e+15 2334.7
##
## Step:  AIC=2321.36
## salary ~ games + weight + hits + careerLength + I(hits^2) + I(careerLeng
th^2) +
##     games:weight + games:hits + weight:hits + games:careerLength +
##     weight:careerLength + hits:careerLength + games:weight:hits +
```

```
##     games:weight:careerLength + games:hits:careerLength + weight:hits:ca
reerLength +
##     games:weight:hits:careerLength
##
##                                    Df  Sum of Sq        RSS    AIC
## - I(careerLength^2)                 1 1.2743e+13 8.8429e+14 2320.5
## <none>                                           8.7155e+14 2321.4
## - games:weight:hits:careerLength    1 1.2049e+14 9.9204e+14 2329.2
## - I(hits^2)                         1 2.0774e+14 1.0793e+15 2335.6
##
## Step:  AIC=2320.47
## salary ~ games + weight + hits + careerLength + I(hits^2) + games:weight
+
##     games:hits + weight:hits + games:careerLength + weight:careerLength
+
##     hits:careerLength + games:weight:hits + games:weight:careerLength +
##     games:hits:careerLength + weight:hits:careerLength + games:weight:hi
ts:careerLength
##
##                                    Df  Sum of Sq        RSS    AIC
## <none>                                           8.8429e+14 2320.5
## - games:weight:hits:careerLength    1 1.0835e+14 9.9264e+14 2327.2
## - I(hits^2)                         1 2.0200e+14 1.0863e+15 2334.1
summary(salaryModel6)
##
## Call:
## lm(formula = salary ~ games + weight + hits + careerLength +
##     I(hits^2) + games:weight + games:hits + weight:hits + games:careerLe
ngth +
##     weight:careerLength + hits:careerLength + games:weight:hits +
##     games:weight:careerLength + games:hits:careerLength + weight:hits:ca
reerLength +
##     games:weight:hits:careerLength, data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7748476 -2068979     2262  1744116 10803344
##
## Coefficients:
```

```
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -3.318e+06  1.644e+07  -0.202 0.840745
## games                           -3.744e+05  5.757e+05  -0.650 0.517997
## weight                           4.012e+03  7.626e+04   0.053 0.958218
## hits                             2.102e+06  1.602e+06   1.313 0.194383
## careerLength                    -1.179e+06  3.601e+06  -0.327 0.744583
## I(hits^2)                       -4.256e+03  1.159e+03  -3.671 0.000521 ***
## games:weight                     1.845e+03  2.790e+03   0.661 0.510881
## games:hits                      -8.444e+03  1.050e+04  -0.804 0.424394
## weight:hits                     -1.073e+04  7.653e+03  -1.402 0.166231
## games:careerLength               2.692e+05  1.169e+05   2.303 0.024840 *
## weight:careerLength              1.550e+04  1.701e+04   0.911 0.365832
## hits:careerLength               -9.112e+05  2.618e+05  -3.480 0.000949 ***
## games:weight:hits                6.529e+01  5.159e+01   1.265 0.210677
## games:weight:careerLength       -1.400e+03  5.597e+02  -2.502 0.015142 *
## games:hits:careerLength          4.316e+03  1.588e+03   2.718 0.008615 **
## weight:hits:careerLength         4.297e+03  1.223e+03   3.513 0.000856 ***
## games:weight:hits:careerLength -1.964e+01  7.304e+00  -2.689 0.009310 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3871000 on 59 degrees of freedom
## Multiple R-squared:  0.713,  Adjusted R-squared:  0.6352
## F-statistic: 9.163 on 16 and 59 DF,  p-value: 9.683e-11
```

This model is still too complicated and we need to try to build more models.

```
salaryModel7 <- lm(salary ~ games + weight + hits + careerLength + I(games^2) + I(weight^2) + I(hits^2) + I(careerLength^2), data = myTeamsData)
summary(salaryModel7)
```

```
##
## Call:
## lm(formula = salary ~ games + weight + hits + careerLength +
##     I(games^2) + I(weight^2) + I(hits^2) + I(careerLength^2),
##     data = myTeamsData)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -7531759 -2666733   -97027  1413197 14782438
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.478e+07  5.758e+07   0.604   0.5478
## games             -1.005e+05  7.080e+04  -1.419   0.1606
## weight            -3.417e+05  5.300e+05  -0.645   0.5213
## hits               2.988e+04  5.168e+04   0.578   0.5651
## careerLength       1.422e+06  4.546e+05   3.128   0.0026 **
## I(games^2)         4.987e+02  7.093e+02   0.703   0.4844
## I(weight^2)        8.396e+02  1.215e+03   0.691   0.4919
## I(hits^2)          5.202e+01  4.341e+02   0.120   0.9050
## I(careerLength^2) -3.053e+04  3.355e+04  -0.910   0.3662
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4680000 on 67 degrees of freedom
## Multiple R-squared:  0.5237, Adjusted R-squared:  0.4669
## F-statistic:  9.21 on 8 and 67 DF,  p-value: 1.87e-08
```

We can see that R-squared value has dropped to 47% and not many terms have a significant p-value. Let us use step() to simplify our model.

```
salaryModel8 <- step(salaryModel7)
## Start:  AIC=2342.97
## salary ~ games + weight + hits + careerLength + I(games^2) +
##     I(weight^2) + I(hits^2) + I(careerLength^2)
##
##                     Df  Sum of Sq        RSS    AIC
## - I(hits^2)          1 3.1453e+11 1.4680e+15 2341.0
## - hits               1 7.3229e+12 1.4750e+15 2341.3
## - weight             1 9.1047e+12 1.4767e+15 2341.4
## - I(weight^2)        1 1.0460e+13 1.4781e+15 2341.5
## - I(games^2)         1 1.0831e+13 1.4785e+15 2341.5
## - I(careerLength^2)  1 1.8134e+13 1.4858e+15 2341.9
## <none>                             1.4676e+15 2343.0
## - games              1 4.4095e+13 1.5117e+15 2343.2
```

```
## - careerLength       1 2.1434e+14 1.6820e+15 2351.3
##
## Step:  AIC=2340.99
## salary ~ games + weight + hits + careerLength + I(games^2) +
##     I(weight^2) + I(careerLength^2)
##
##                     Df  Sum of Sq        RSS    AIC
## - weight             1 1.1783e+13 1.4797e+15 2339.6
## - hits               1 1.3021e+13 1.4810e+15 2339.7
## - I(weight^2)        1 1.3590e+13 1.4815e+15 2339.7
## - I(careerLength^2)  1 1.7855e+13 1.4858e+15 2339.9
## - I(games^2)         1 3.5338e+13 1.5033e+15 2340.8
## <none>                            1.4680e+15 2341.0
## - games              1 9.8095e+13 1.5661e+15 2343.9
## - careerLength       1 2.1515e+14 1.6831e+15 2349.4
##
## Step:  AIC=2339.59
## salary ~ games + hits + careerLength + I(games^2) + I(weight^2) +
##     I(careerLength^2)
##
##                     Df  Sum of Sq        RSS    AIC
## - hits               1 1.3093e+13 1.4928e+15 2338.3
## - I(careerLength^2)  1 1.6666e+13 1.4964e+15 2338.4
## - I(weight^2)        1 1.8099e+13 1.4978e+15 2338.5
## <none>                            1.4797e+15 2339.6
## - I(games^2)         1 4.0878e+13 1.5206e+15 2339.7
## - games              1 1.1091e+14 1.5907e+15 2343.1
## - careerLength       1 2.0919e+14 1.6889e+15 2347.6
##
## Step:  AIC=2338.26
## salary ~ games + careerLength + I(games^2) + I(weight^2) + I(careerLengt
h^2)
##
##                     Df  Sum of Sq        RSS    AIC
## - I(careerLength^2)  1 1.4651e+13 1.5075e+15 2337.0
## - I(weight^2)        1 2.4602e+13 1.5174e+15 2337.5
## <none>                            1.4928e+15 2338.3
```

```
## - games              1 1.2104e+14 1.6139e+15 2342.2
## - I(games^2)         1 1.8012e+14 1.6730e+15 2344.9
## - careerLength       1 2.0514e+14 1.6980e+15 2346.1
##
## Step:  AIC=2337
## salary ~ games + careerLength + I(games^2) + I(weight^2)
##
##                Df  Sum of Sq        RSS    AIC
## - I(weight^2)   1 2.1876e+13 1.5294e+15 2336.1
## <none>                       1.5075e+15 2337.0
## - games         1 1.1515e+14 1.6226e+15 2340.6
## - I(games^2)    1 1.7136e+14 1.6788e+15 2343.2
## - careerLength  1 9.9533e+14 2.5028e+15 2373.5
##
## Step:  AIC=2336.1
## salary ~ games + careerLength + I(games^2)
##
##                Df  Sum of Sq        RSS    AIC
## <none>                       1.5294e+15 2336.1
## - games         1 1.4890e+14 1.6783e+15 2341.2
## - I(games^2)    1 2.1831e+14 1.7477e+15 2344.2
## - careerLength  1 1.0641e+15 2.5934e+15 2374.2
summary(salaryModel8)
##
## Call:
## lm(formula = salary ~ games + careerLength + I(games^2), data = myTeamsD
ata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7829311 -2580896  -160829  1221720 16998321
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1508220.8  1494345.0   1.009  0.31622
## games        -124723.2    47107.3  -2.648  0.00995 **
## careerLength 1054749.6   149021.6   7.078 7.94e-10 ***
```

```
## I(games^2)       905.3      282.4   3.206  0.00201 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4609000 on 72 degrees of freedom
## Multiple R-squared:  0.5037, Adjusted R-squared:  0.483
## F-statistic: 24.36 on 3 and 72 DF,  p-value: 5.469e-11
```

This model has many significant coefficients and a R-squared value of 46%. This model is not so complex as it has only few coefficients.

But even this model has an issue. If we do a vif analysis, we will have issues with our model because we are including two functionally related explanatory variables in our model.

```
# vif() is used to do do the vif analysis, if the values are greater than 5 then it suggests that there is some issue with our model
vif(salaryModel8)
```

```
##        games careerLength   I(games^2)
##    16.523150     1.062947    16.445454
```

The vif analysis points towards possible issues.

We will now remove games from our model and check how it effects our model.

```
salaryModel9 <- update(salaryModel8, . ~ . - games)
summary(salaryModel9)
```

```
##
## Call:
## lm(formula = salary ~ careerLength + I(games^2), data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7773417 -2644211  -760121  1524361 16132264
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.537e+06  9.925e+05  -1.549    0.126
## careerLength  1.028e+06  1.547e+05   6.644 4.74e-09 ***
## I(games^2)    1.821e+02  7.451e+01   2.444    0.017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4795000 on 73 degrees of freedom
## Multiple R-squared:  0.4554, Adjusted R-squared:  0.4405
## F-statistic: 30.52 on 2 and 73 DF,  p-value: 2.326e-10
```

```
vif(salaryModel9)
```

```
## careerLength   I(games^2)
##     1.057888     1.057888
```

This model looks decent as all the coefficients are significant, but the R-squared value is only 48%.

The vif analysis looks fine and do not point towards any issues.

Although this is not a very impressive model due to its R-squared value but this seems to be the simplest and significant model for salary prediction.

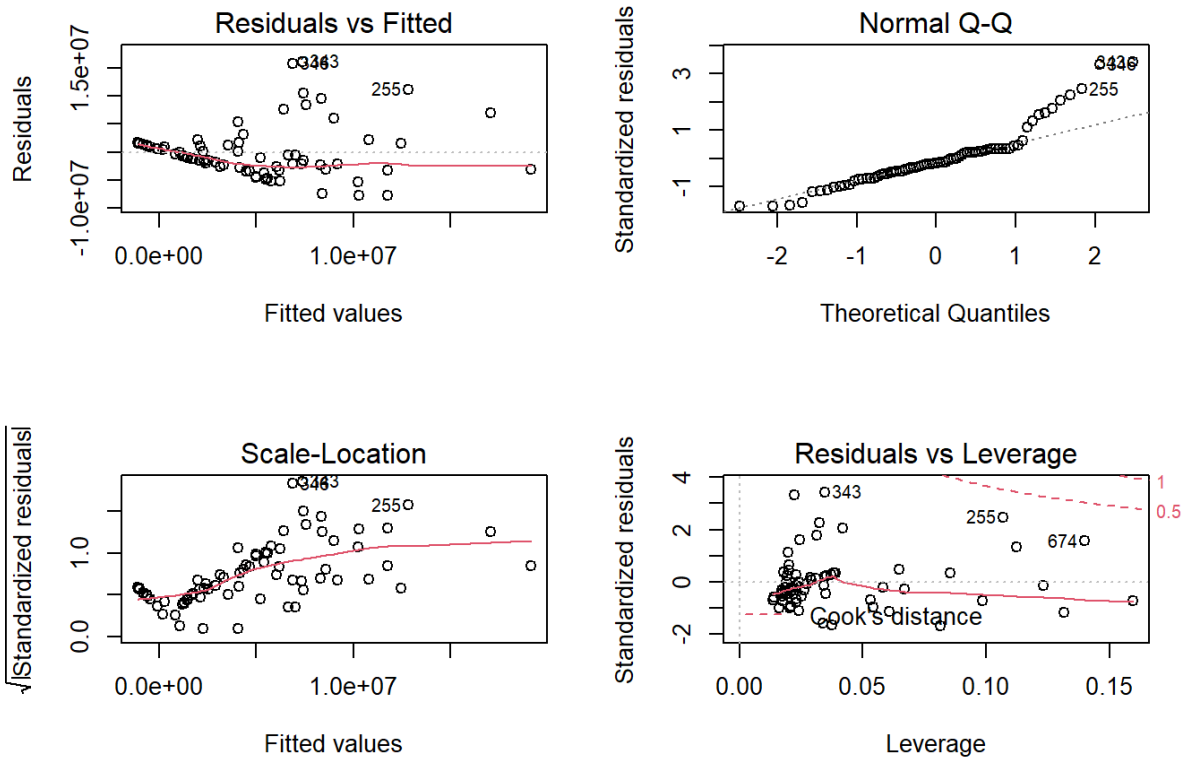
$$salary=(-1.537e+06)+(1.028e+06)\times careerLength+(1.821e+02)\times games2salary=(-1.537e+06)+(1.028e+06)\times careerLength+(1.821e+02)\times games2$$

3.2 Critique model using relevant diagnostics

```
par(mfrow=c(2,2))


summary(salaryModel9)
```

```
##
## Call:
## lm(formula = salary ~ careerLength + I(games^2), data = myTeamsData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7773417 -2644211  -760121  1524361 16132264
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.537e+06  9.925e+05  -1.549    0.126
## careerLength  1.028e+06  1.547e+05   6.644 4.74e-09 ***
## I(games^2)    1.821e+02  7.451e+01   2.444    0.017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4795000 on 73 degrees of freedom
## Multiple R-squared:  0.4554, Adjusted R-squared:  0.4405
## F-statistic: 30.52 on 2 and 73 DF,  p-value: 2.326e-10
```

```
plot(salaryModel9)
```



The summary() function shows that the residuals are well spread. An ideal residual spread occurs when the min and max values are equally far from 0. But our model depicts that it is not ideal. The R-squared value of 48% points that it will not explain more than half of the variance. It will not predict a good player salary.

plot() gives us 4 plots which can be used to analyse the goodness of the model visually. We can see in the first plot that there is a pattern and the spread of the residuals is greater for higher values on the x-axis. This is called heteroscedasticity.

While heteroscedasticity does not cause bias in the coefficient estimates, it does make them less precise. Lower precision increases the likelihood that the coefficient estimates are further from the correct population value. (Ref: Hayes, A.)

There are some outliers visible in the plot too.

There is also a S shape emerging in the qq-plot. The points should be on a straight line to have a good model.

3.3 Suggest improvements to your model

We saw that our model had a low R-squared value and the plots depicted heteroscedasticity. Most of the variables have skewness.

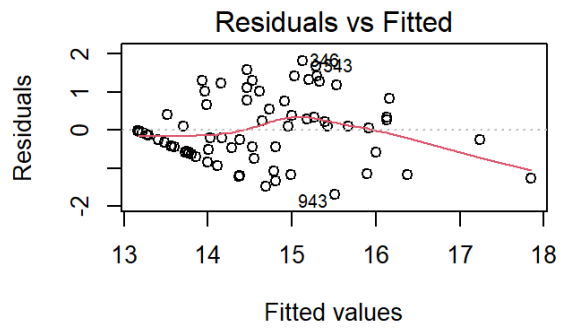
We can improve our model by transforming the dependent variable. When all the values are greater than 0 in our variable we can try log() and sqrt() transformation to the dependent variable.

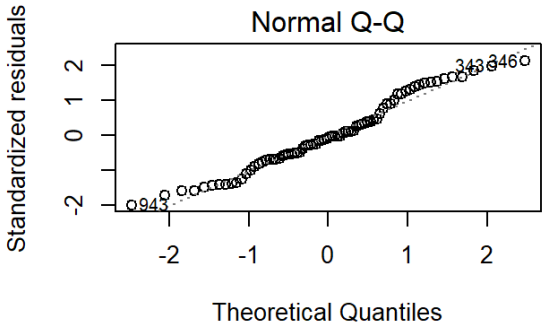
```
par(mfrow=c(2,2))
```

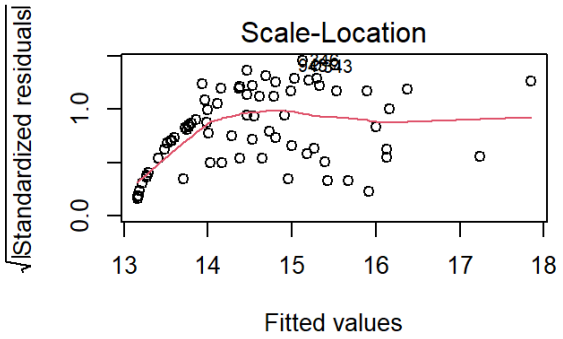
```
salaryModel10 <- lm(log(salary) ~ careerLength + I(games^2), data = myTeams
Data)
summary(salaryModel10)
```

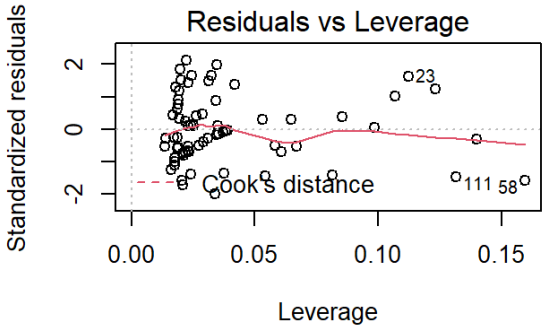
```
##
## Call:
## lm(formula = log(salary) ~ careerLength + I(games^2), data = myTeamsData
)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.69692 -0.58249 -0.06178  0.56357  1.81555
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.306e+01  1.792e-01  72.893  < 2e-16 ***
## careerLength 2.579e-01  2.792e-02   9.240 6.63e-14 ***
## I(games^2)   2.576e-05  1.345e-05   1.916   0.0593 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8655 on 73 degrees of freedom
## Multiple R-squared:  0.5851, Adjusted R-squared:  0.5738
## F-statistic: 51.48 on 2 and 73 DF,  p-value: 1.133e-14
```

```
plot(salaryModel10)
```





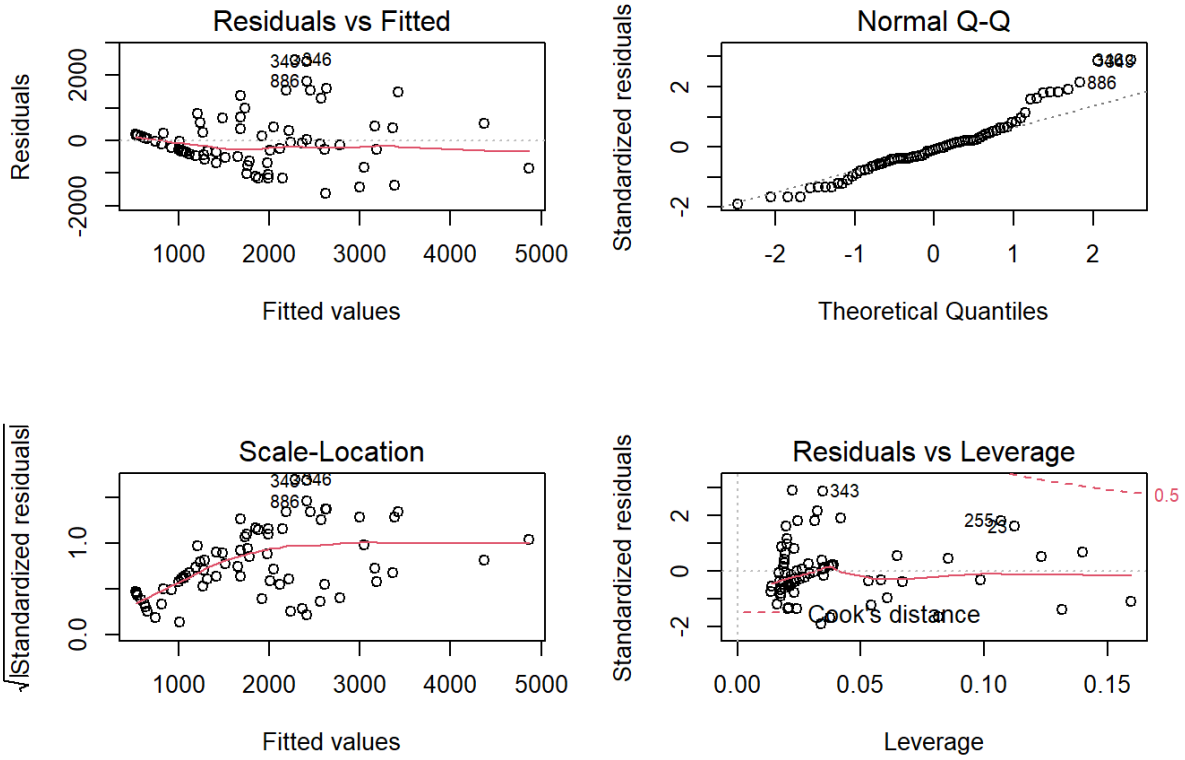





```
salaryModel11 <- lm(sqrt(salary) ~ careerLength + I(games^2), data = myTeam
sData)

summary(salaryModel11)
```

```
##
## Call:
## lm(formula = sqrt(salary) ~ careerLength + I(games^2), data = myTeamsDat
a)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1627.49  -474.77   -83.49   356.77  2480.73
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  435.58712  179.77788   2.423   0.0179 *
## careerLength 228.14899   28.01494   8.144 7.56e-12 ***
## I(games^2)     0.03228    0.01350   2.392   0.0193 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 868.5 on 73 degrees of freedom
## Multiple R-squared:  0.5405, Adjusted R-squared:  0.5279
## F-statistic: 42.93 on 2 and 73 DF,  p-value: 4.729e-13
plot(salaryModel11)
```



Both of the models have a better R-squared value and a significant F-statistic value, but they are complex as compared to the model developed in 3.1. So these two models can be considered an improvement in the model.

We have so many outliers in our data which are making it difficult to have a fit model. We can remove the outliers and then try building a model.

We have data for year 2015 only and salary of a player depends on form a of a player and many other factors. So if we had data for more seasons that might have helped in building a better model.

4. Extension work

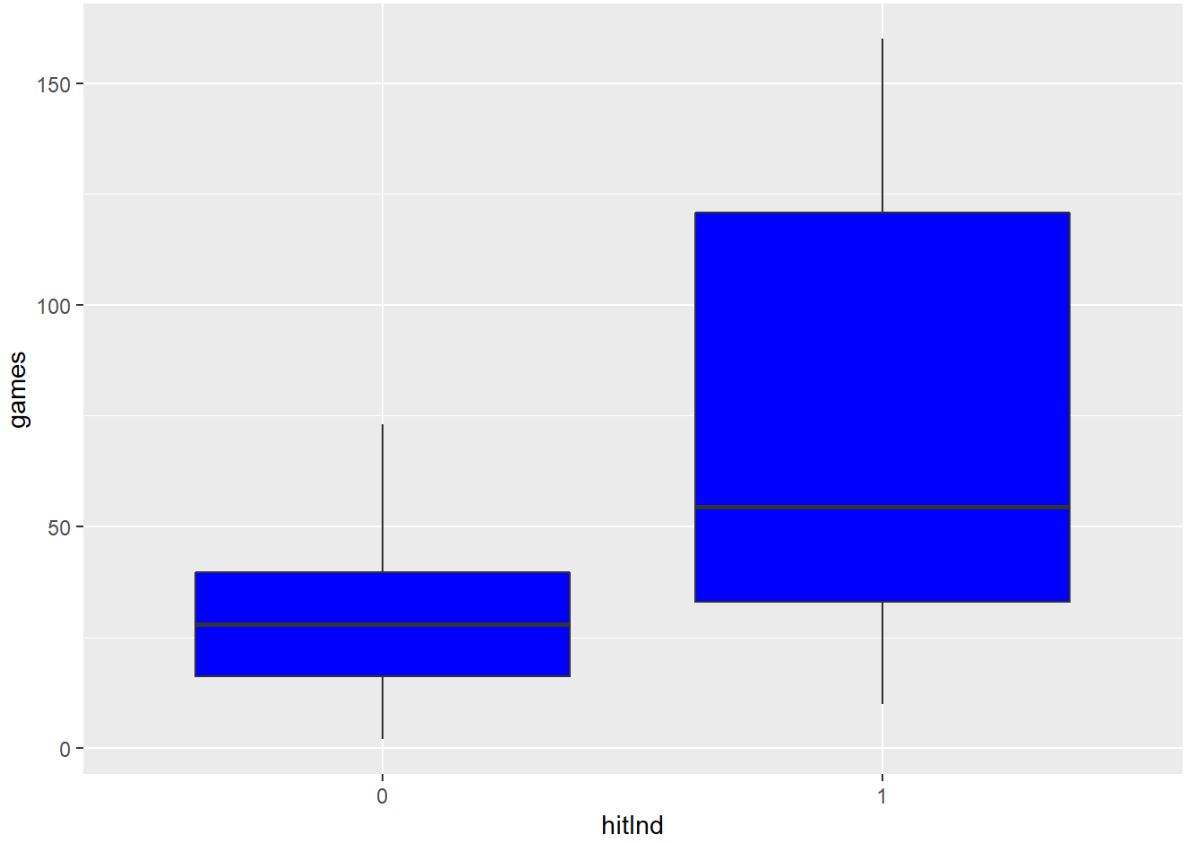
4.1 Model the likelihood of a player having scored a Hit (using the hit.ind variable provided).

We need to predict the likelihood of a player having scored a hit i.e. hitInd = 1 or not having a hit i.e hitInd = 0. In this case our dependent variable is of binary data type so we will be using a generalized linear model with a binomial family.

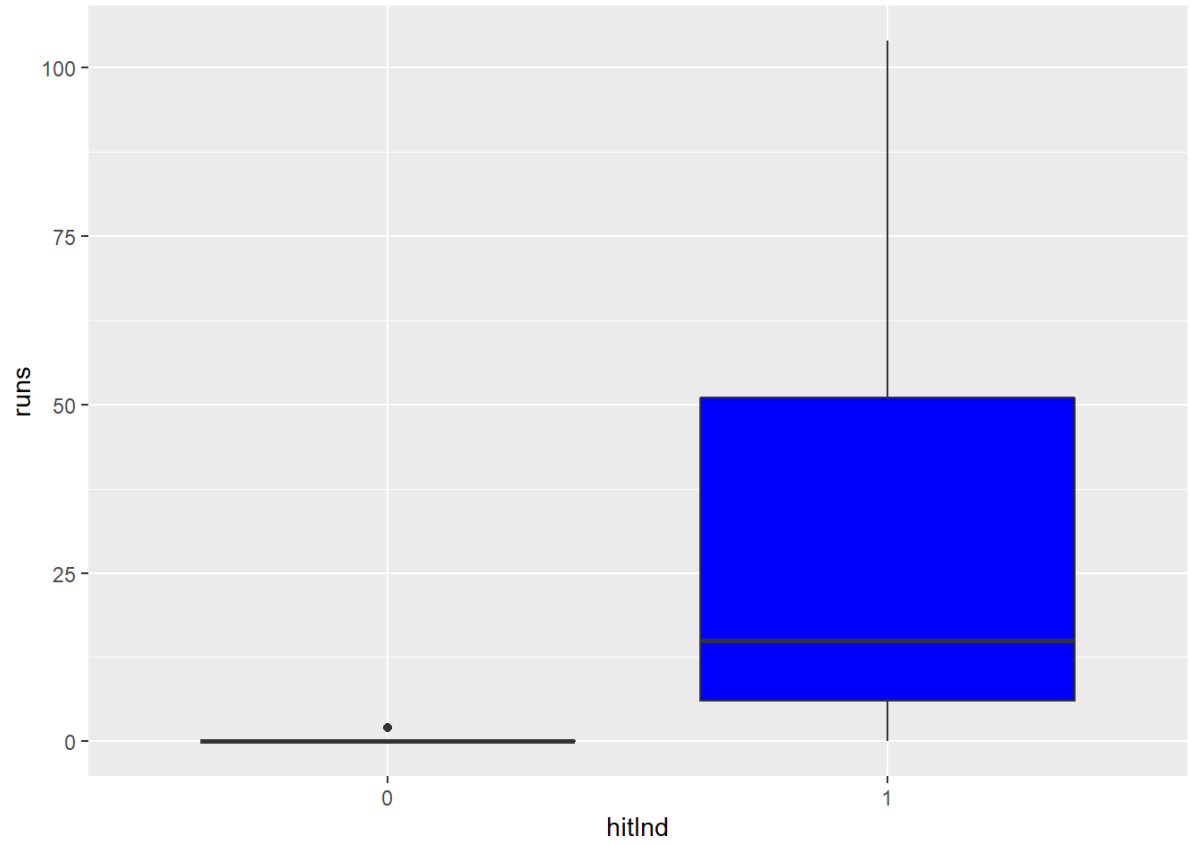
We have some independent variables which have strong correlation with our dependent variable so those should not be included in our model as it affects the prediction ability of the model.

We can check the variables and their relation with hitInd visually as below:

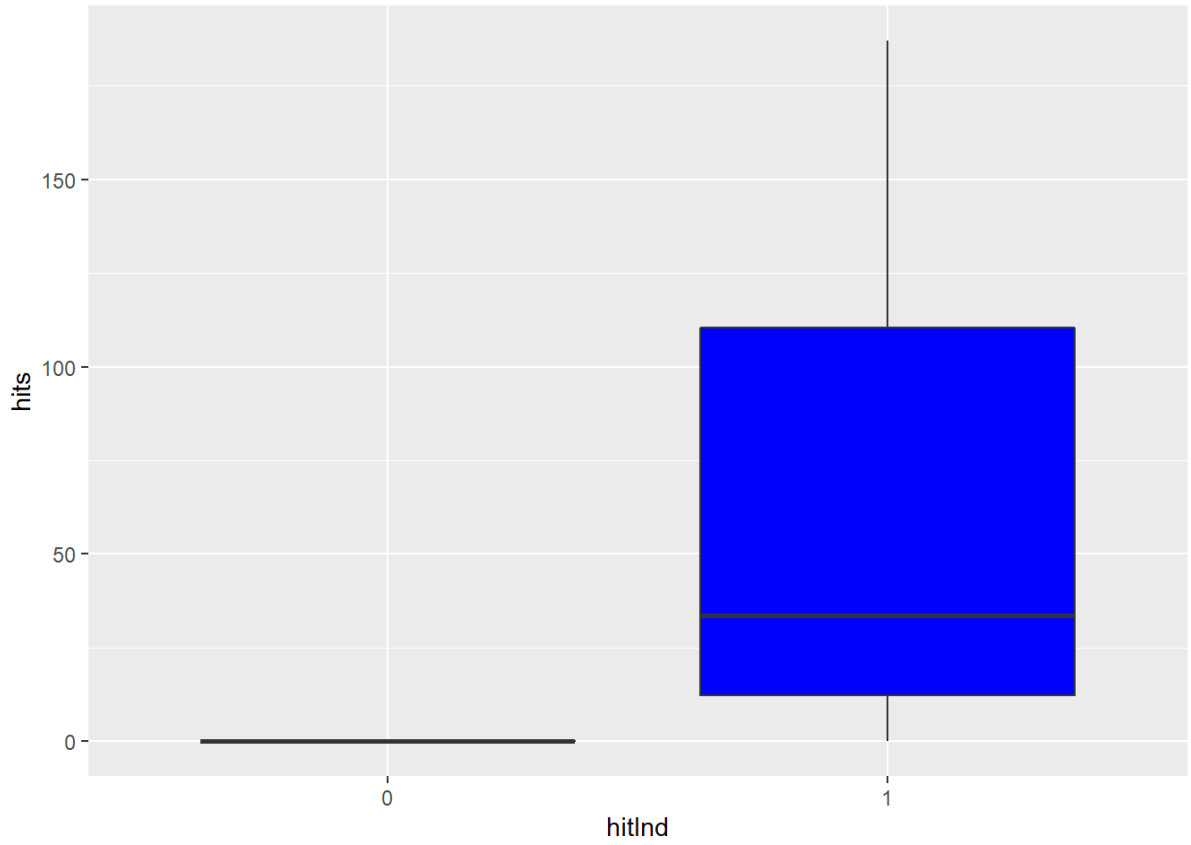
```
ggplot(myTeamsData,aes(x=hitInd,y=games))+geom_boxplot(fill = "blue")
```



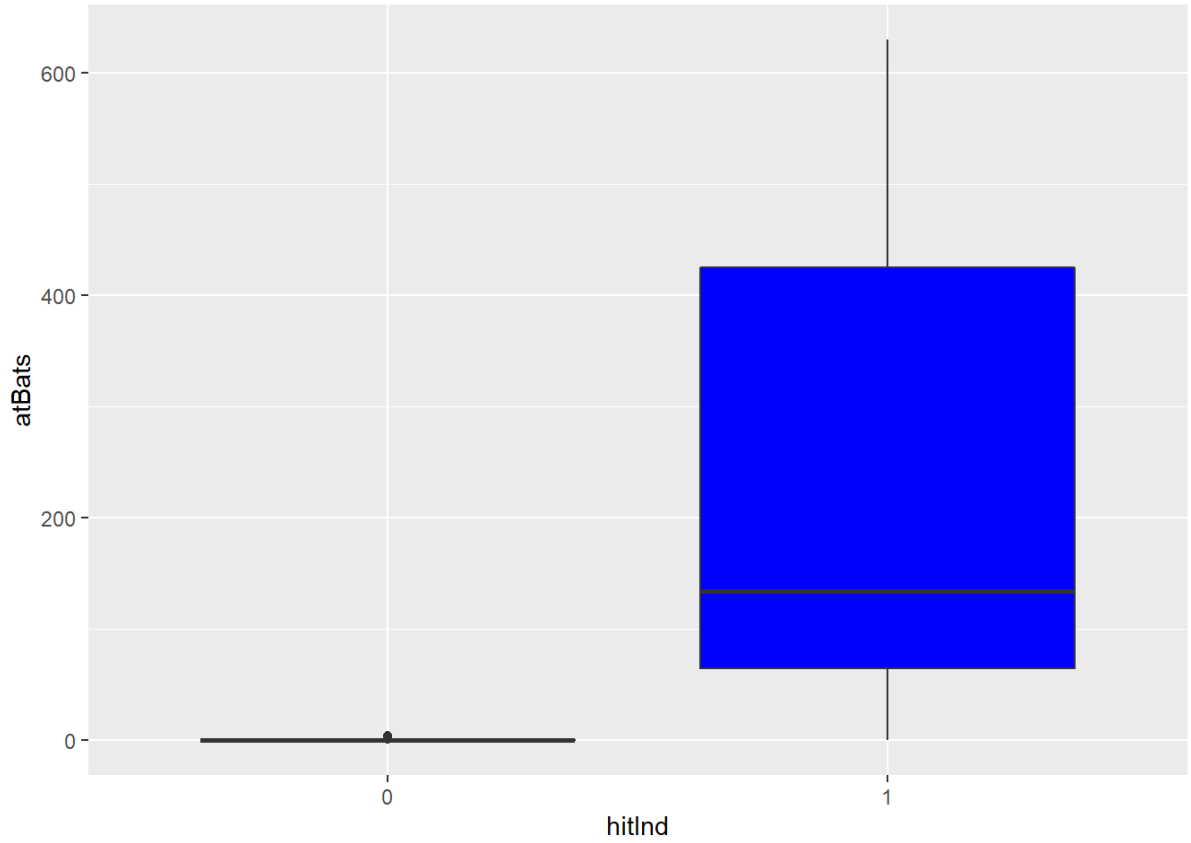

```
ggplot(myTeamsData,aes(x=hitInd,y=runs))+geom_boxplot(fill = "blue")
```



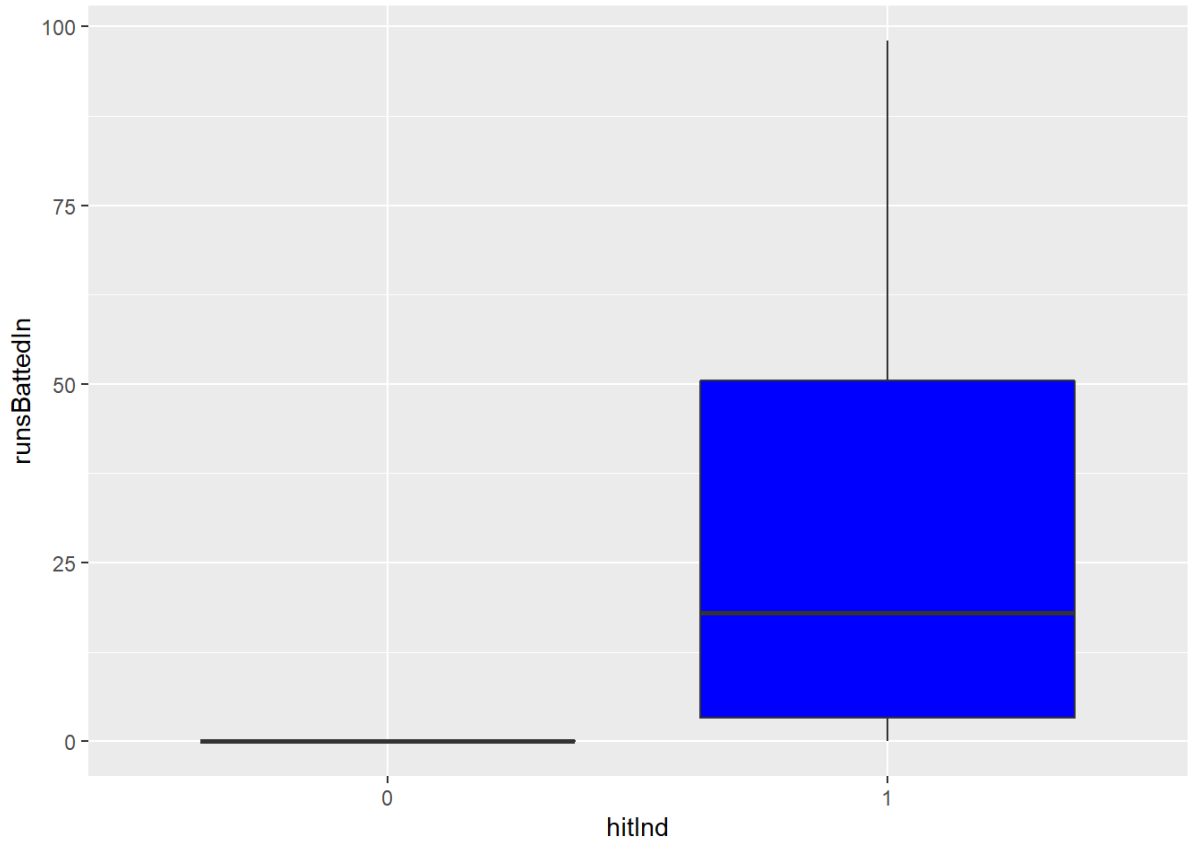

```
ggplot(myTeamsData,aes(x=hitInd,y=hits))+geom_boxplot(fill = "blue")
```



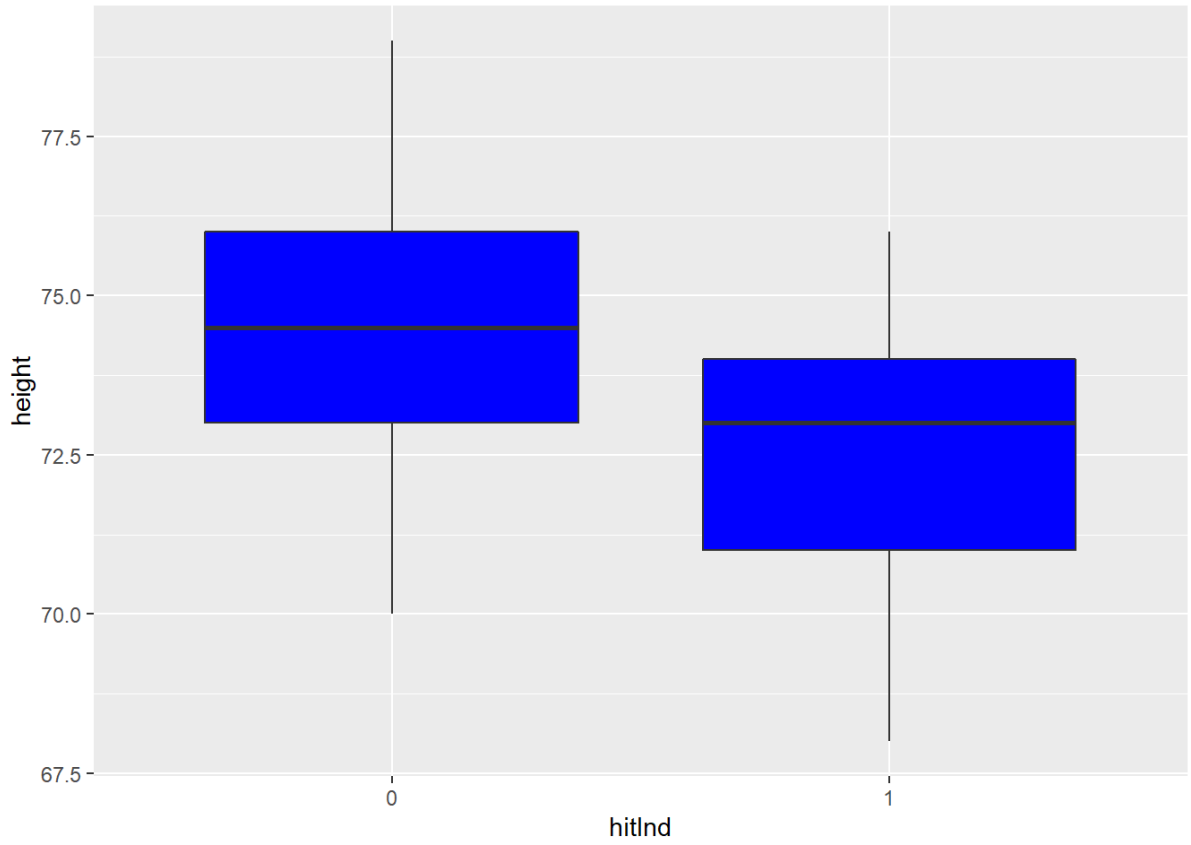
```
ggplot(myTeamsData,aes(x=hitInd,y=atBats))+geom_boxplot(fill = "blue")
```



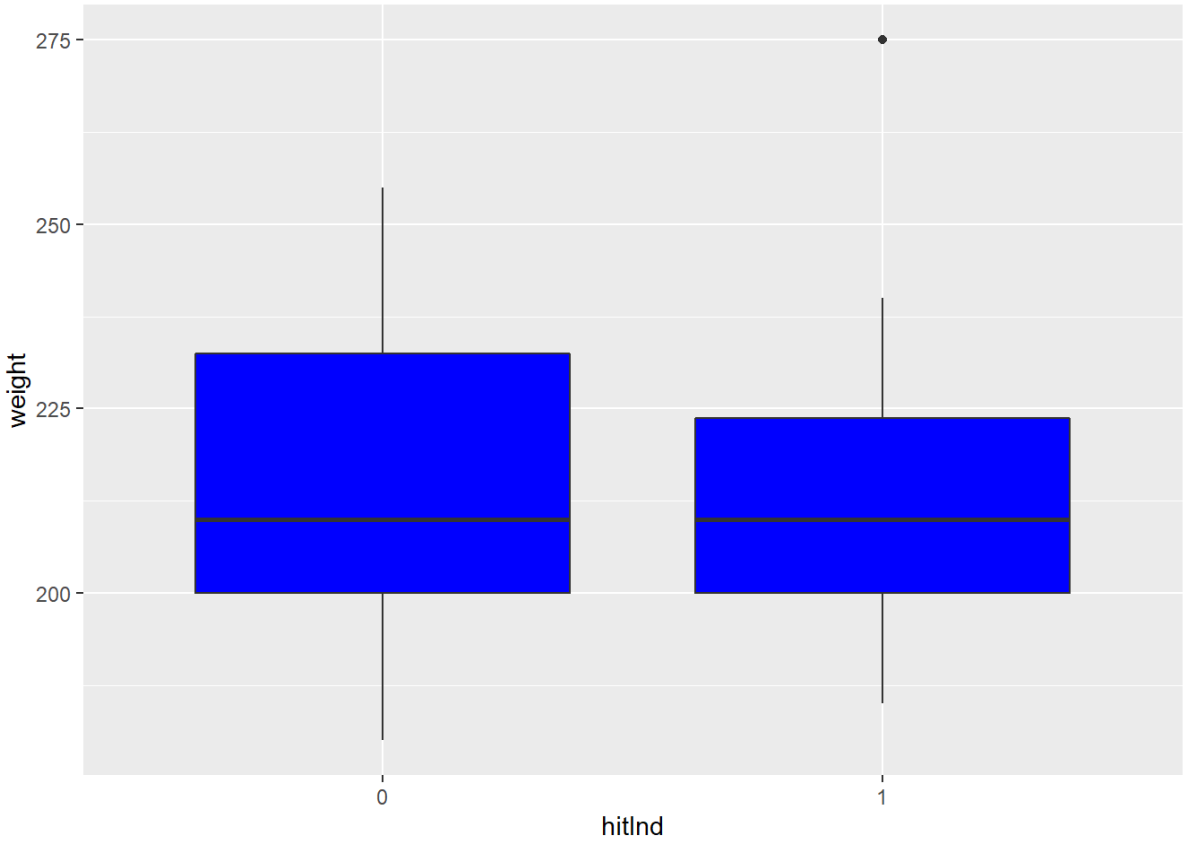

```
ggplot(myTeamsData,aes(x=hitInd,y=runsBattedIn))+geom_boxplot(fill = "blue"
)
```



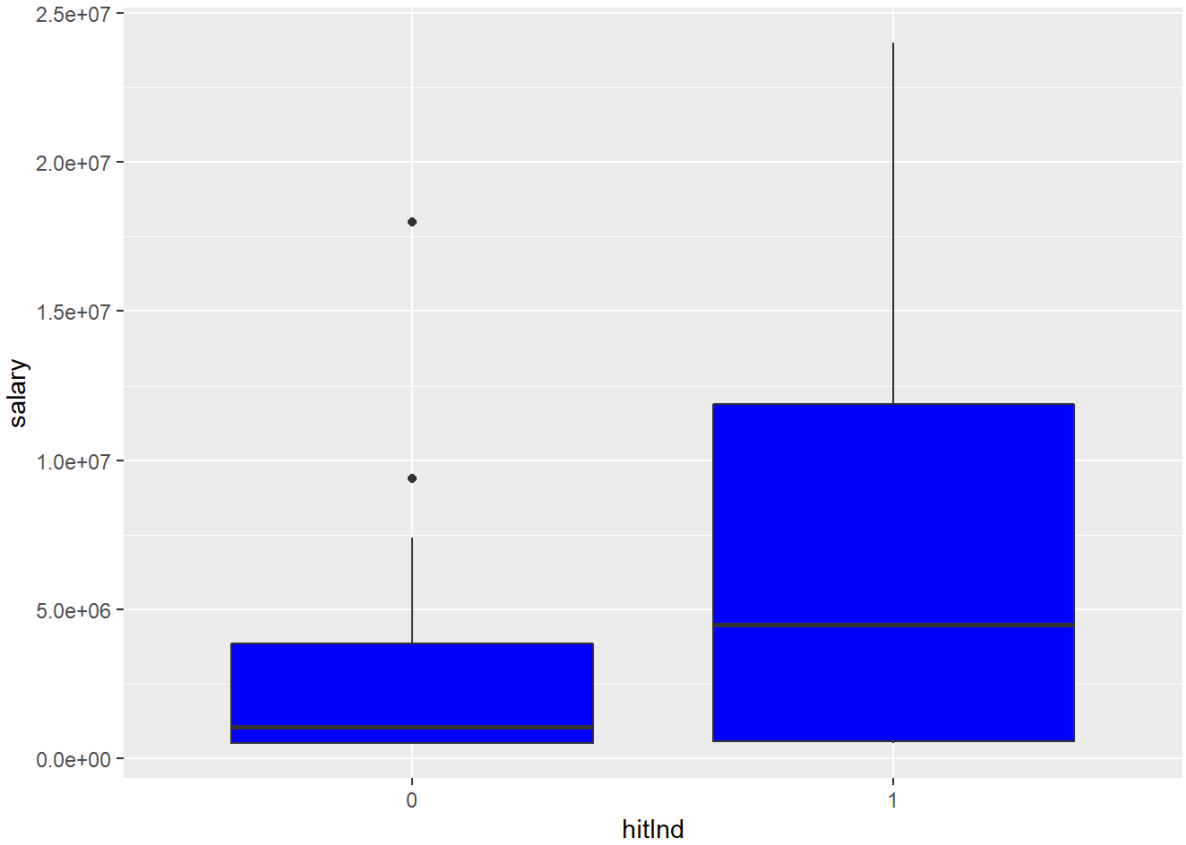

```
ggplot(myTeamsData,aes(x=hitInd,y=height))+geom_boxplot(fill = "blue")
```



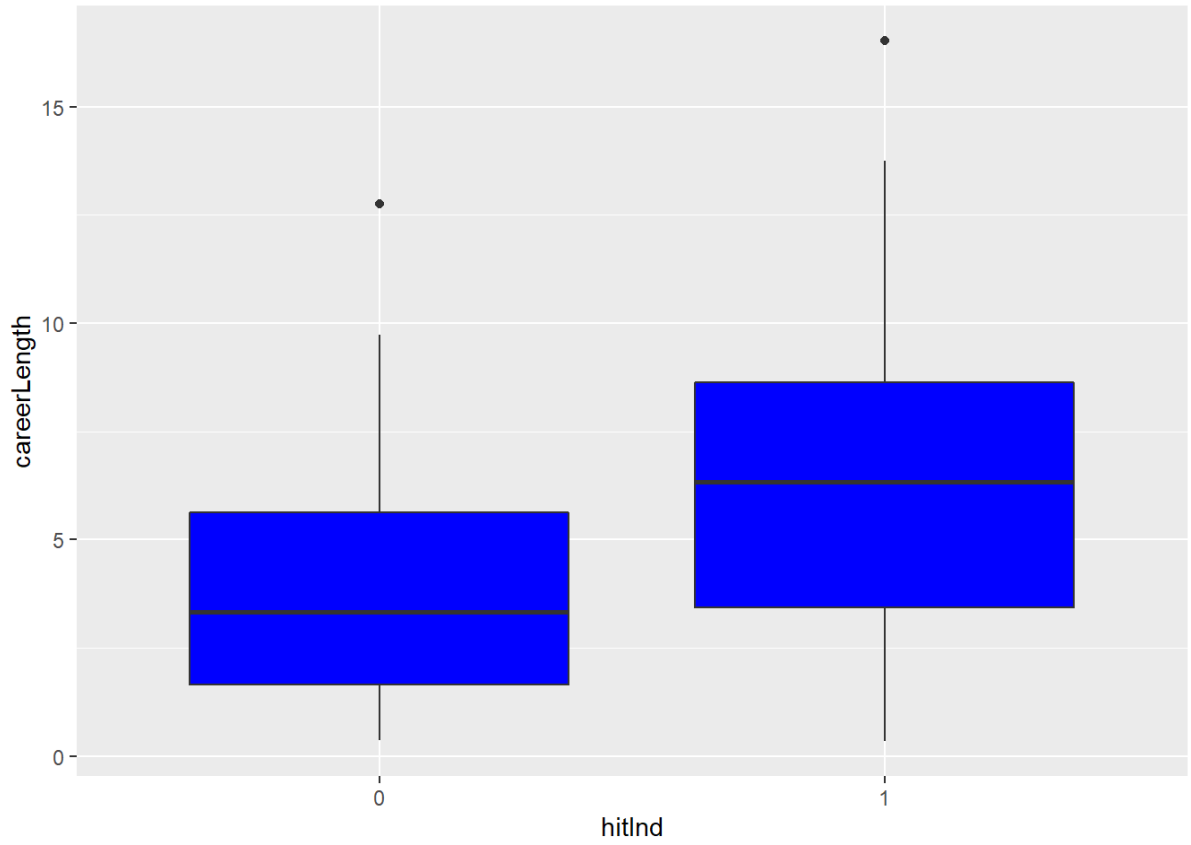
```
ggplot(myTeamsData,aes(x=hitInd,y=weight))+geom_boxplot(fill = "blue")
```



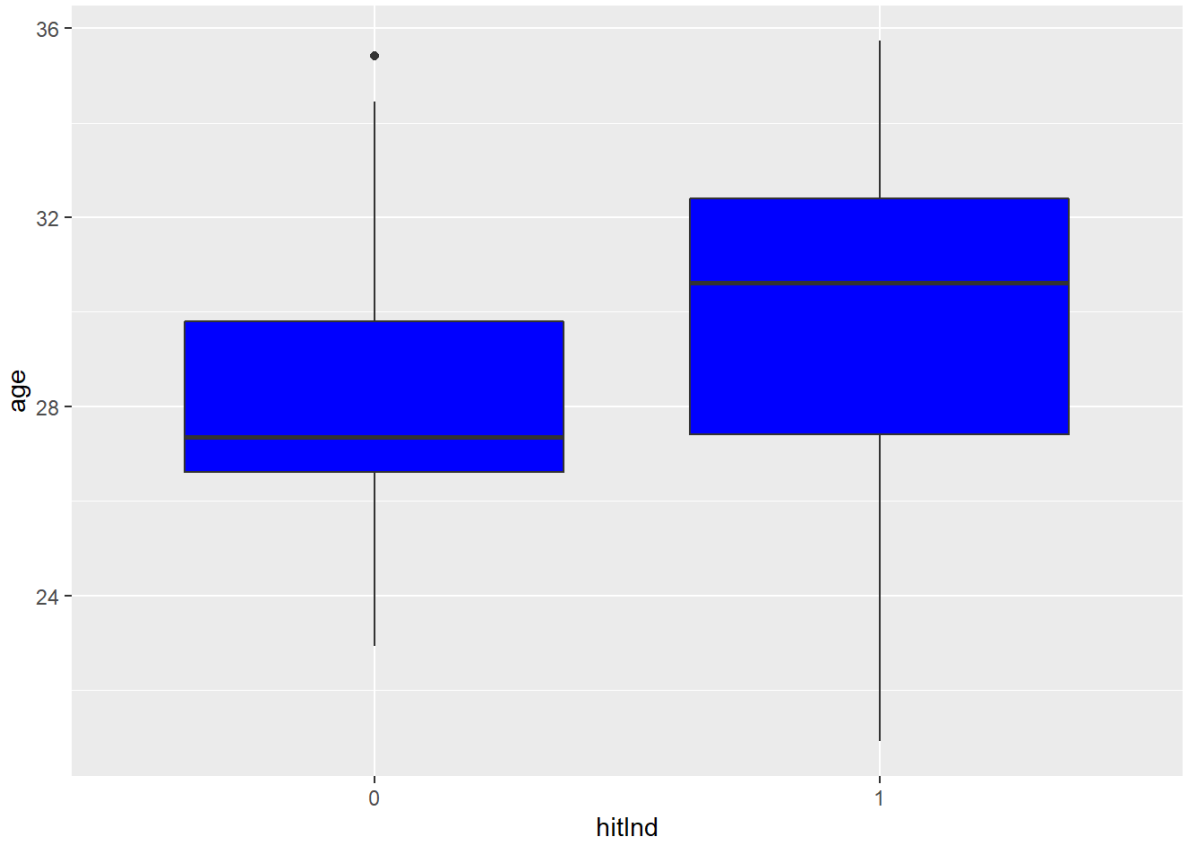

```
ggplot(myTeamsData,aes(x=hitInd,y=salary))+geom_boxplot(fill = "blue")
```




```
ggplot(myTeamsData,aes(x=hitInd,y=careerLength))+geom_boxplot(fill = "blue"
)
```

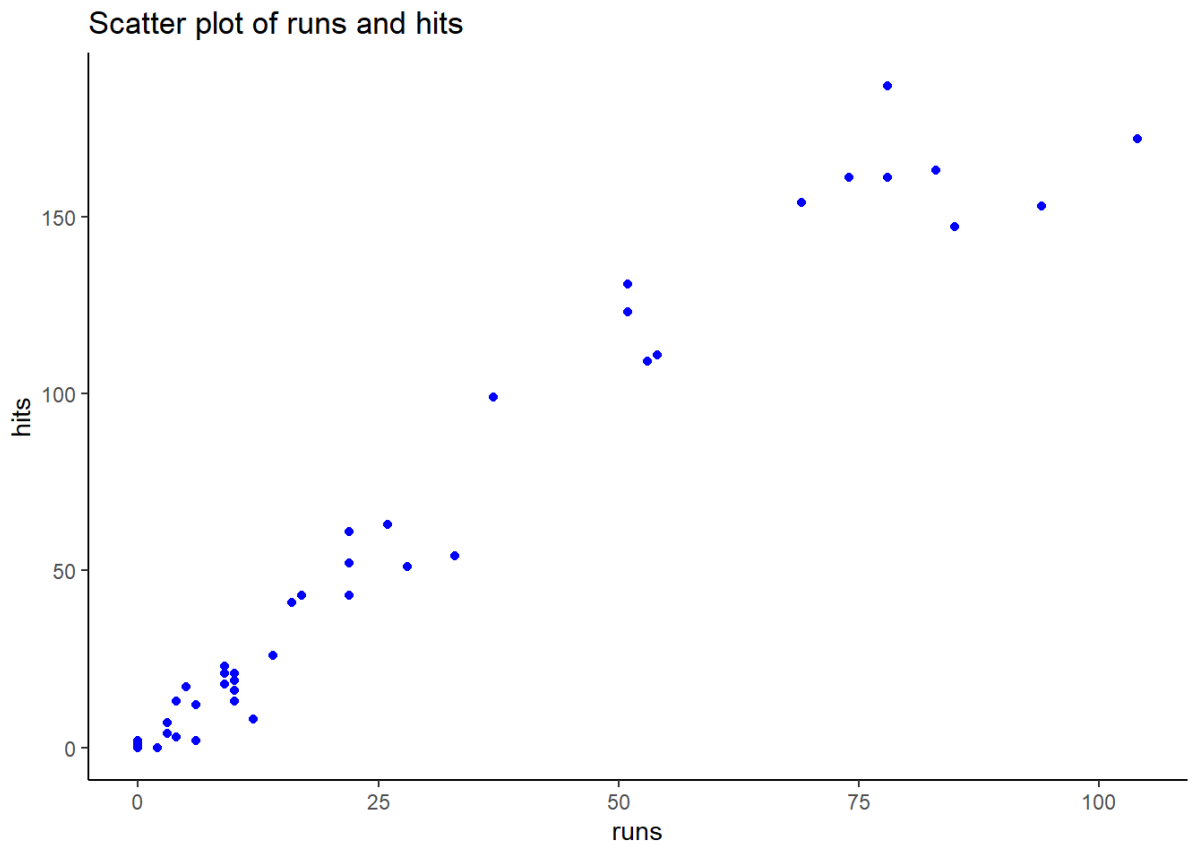



```
ggplot(myTeamsData,aes(x=hitInd,y=age))+geom_boxplot(fill = "blue")
```

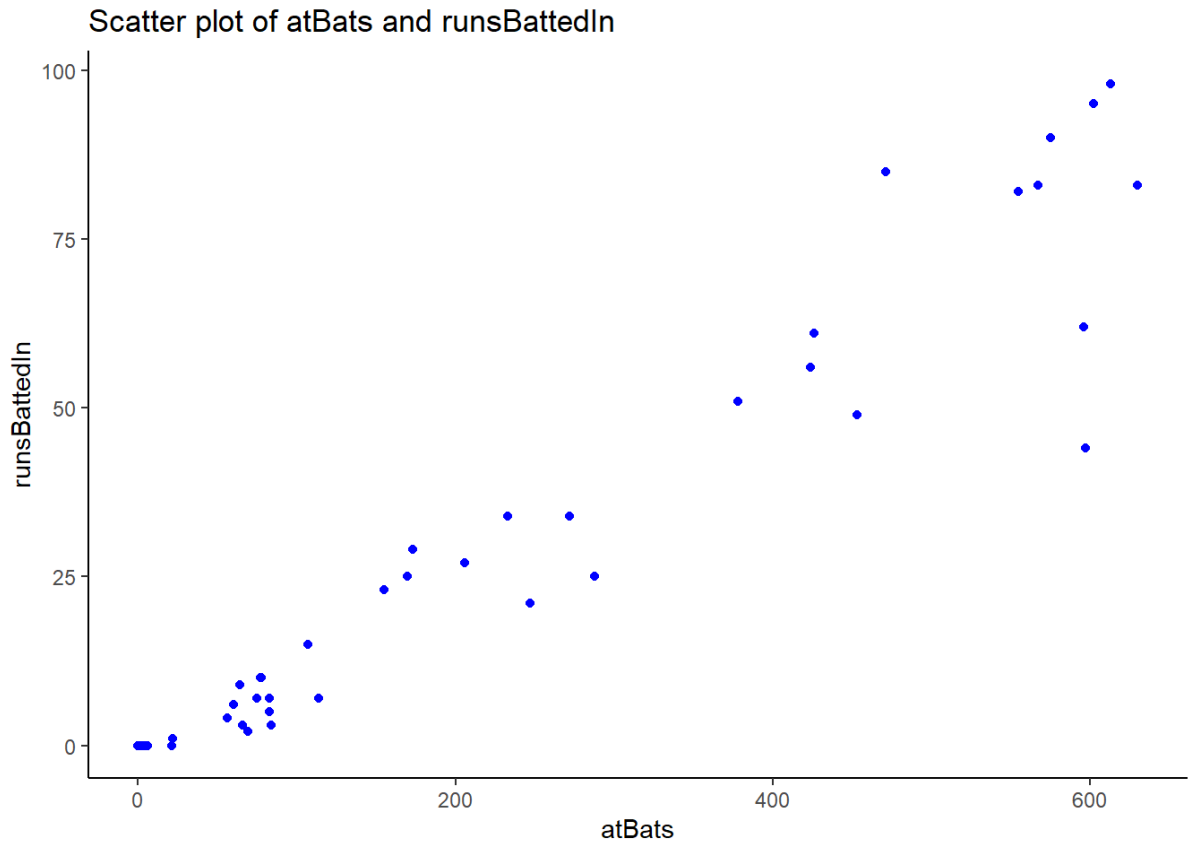


We should also check for collinearity between the numeric explanatory variables.

```
ggplot(myTeamsData, aes(x=runs, y=hits)) + geom_point(color="blue") + ggtit
le("Scatter plot of runs and hits") + theme_classic()
```




```
ggplot(myTeamsData, aes(x=atBats, y=runsBattedIn)) + geom_point(color="blue
") + ggtitle("Scatter plot of atBats and runsBattedIn") + theme_classic()
```



These variables are strongly correlated so we should not include these in our model as change in 1 of them wil have the same almost same effect in all other, so just include one of these..

Based on the plots now we will be making a model.

```
hitPredict1 <- glm(hitInd ~ games + height + weight + salary + careerLength
+ age + bats, data = myTeamsData, family = "binomial")

summary(hitPredict1)
```

```
##
## Call:
## glm(formula = hitInd ~ games + height + weight + salary + careerLength +
##     age + bats, family = "binomial", data = myTeamsData)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.80274  -0.75478   0.04807   0.57647   2.13358
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.296e+01  1.403e+01   1.637  0.10169
## games         3.377e-02  1.215e-02   2.779  0.00546 **
## height       -5.360e-01  2.197e-01  -2.440  0.01469 *
```

```
## weight        4.498e-02  2.667e-02   1.686  0.09172 .
## salary        1.761e-07  9.468e-08   1.860  0.06285 .
## careerLength -1.129e-01  1.594e-01  -0.708  0.47868
## age           1.924e-01  1.449e-01   1.328  0.18427
## batsL         1.030e+00  1.732e+00   0.594  0.55220
## batsR        -4.575e-01  1.583e+00  -0.289  0.77257
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 104.51  on 75  degrees of freedom
## Residual deviance:  64.05  on 67  degrees of freedom
## AIC: 82.05
##
## Number of Fisher Scoring iterations: 6
# Use step() to get a better model.
hitPredict2 <- step(hitPredict1)
## Start:  AIC=82.05
## hitInd ~ games + height + weight + salary + careerLength + age +
##     bats
##
##                Df Deviance    AIC
## - careerLength  1   64.557 80.557
## - age           1   65.818 81.818
## - bats          2   67.999 81.999
## <none>              64.050 82.050
## - weight        1   67.145 83.145
## - salary        1   68.649 84.649
## - height        1   71.543 87.543
## - games         1   75.155 91.155
##
## Step:  AIC=80.56
## hitInd ~ games + height + weight + salary + age + bats
##
##          Df Deviance    AIC
## - age     1   65.937 79.937
```

```
## - bats    2   68.328 80.328
## <none>        64.557 80.557
## - weight  1   67.289 81.289
## - salary  1   68.718 82.718
## - height  1   72.080 86.080
## - games   1   75.172 89.172
##
## Step:  AIC=79.94
## hitInd ~ games + height + weight + salary + bats
##
##          Df Deviance    AIC
## - bats    2   69.566 79.566
## <none>        65.937 79.937
## - weight  1   69.120 81.120
## - salary  1   73.085 85.085
## - height  1   74.703 86.703
## - games   1   75.973 87.973
##
## Step:  AIC=79.57
## hitInd ~ games + height + weight + salary
##
##          Df Deviance    AIC
## - weight  1   71.103 79.103
## <none>        69.566 79.566
## - height  1   77.171 85.171
## - salary  1   78.544 86.544
## - games   1   80.600 88.600
##
## Step:  AIC=79.1
## hitInd ~ games + height + salary
##
##          Df Deviance    AIC
## <none>        71.103 79.103
## - height  1   77.270 83.270
## - salary  1   79.682 85.682
## - games   1   82.665 88.665
summary(hitPredict2)
```

```
##
## Call:
## glm(formula = hitInd ~ games + height + salary, family = "binomial",
##     data = myTeamsData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9743  -0.8011   0.0689   0.6964   2.0461
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.517e+01  1.165e+01   2.161  0.03070 *
## games        3.061e-02  1.073e-02   2.853  0.00433 **
## height      -3.670e-01  1.588e-01  -2.311  0.02081 *
## salary       1.643e-07  6.609e-08   2.486  0.01290 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 104.515  on 75  degrees of freedom
## Residual deviance:  71.103  on 72  degrees of freedom
## AIC: 79.103
##
## Number of Fisher Scoring iterations: 6
```

This model gives us the likelihood of a player having scored a hit based on some independent variable.

$$\log(_{p1-p})=(2.517e+01)+(3.061e-02)\times games-(03.670e-01)\times height+(1.643e-07)\times salarylog(p1-p)=(2.517e+01)+(3.061e-02)\times games-(03.670e-01)\times height+(1.643e-07)\times salary$$

```
summary(hitPredict2)
```

```
##
## Call:
## glm(formula = hitInd ~ games + height + salary, family = "binomial",
##     data = myTeamsData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.9743  -0.8011   0.0689   0.6964   2.0461
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.517e+01  1.165e+01   2.161  0.03070 *
## games        3.061e-02  1.073e-02   2.853  0.00433 **
## height      -3.670e-01  1.588e-01  -2.311  0.02081 *
## salary       1.643e-07  6.609e-08   2.486  0.01290 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 104.515  on 75  degrees of freedom
## Residual deviance:  71.103  on 72  degrees of freedom
## AIC: 79.103
##
## Number of Fisher Scoring iterations: 6
```

The model gives of coeffecients with significant p-values. It is the best model I think to predict the likelihood of having a hit. No model is perfect and there is always a model better than you model. So with same assumption we can say that there will be better models than this.

We cannot plot() graphs for glm's. But we can calculate odds and odds ratio.

```
coef(hitPredict2)
##   (Intercept)         games        height        salary
##  2.517263e+01  3.061482e-02 -3.670238e-01  1.643337e-07
exp(coef(hitPredict2))
##  (Intercept)        games       height       salary
## 8.557255e+10 1.031088e+00 6.927932e-01 1.000000e+00
```

We can predict the values for our data set and compare how many of the values are correctly predicted.

```
myTeamsData$hitPredict <- predict(hitPredict2, type = "response")
myTeamsData$hitPredict <- ifelse(myTeamsData$hitPredict > 0.50, 1, 0)
myTeamsData$hitPredict <- as.factor(myTeamsData$hitPredict)
```

We can use ggpredict() function to check some of the predicted values and the confidence intervals of the same variable.

```
# Shows predicted hitInd values for different no of games based on an adjus
ted height and salary
ggpredict(hitPredict2, terms = "games [all]")
## # Predicted probabilities of hitInd
##
## games | Predicted |       95% CI
## --------------------------------
##     2 |      0.25 | [0.11, 0.47]
##    17 |      0.34 | [0.19, 0.53]
##    26 |      0.40 | [0.26, 0.57]
##    33 |      0.46 | [0.31, 0.61]
##    50 |      0.59 | [0.43, 0.73]
##    72 |      0.74 | [0.53, 0.87]
##    93 |      0.84 | [0.60, 0.95]
##   160 |      0.98 | [0.75, 1.00]
##
## Adjusted for:
## * height =      74.00
## * salary = 4895186.89
# Shows predicted hitInd values for different heights based on an adjusted
no of games and salary
ggpredict(hitPredict2, terms = "height [all]")
## # Predicted probabilities of hitInd
##
## height | Predicted |       95% CI
## --------------------------------
##     68 |      0.89 | [0.53, 0.98]
##     69 |      0.85 | [0.52, 0.97]
##     71 |      0.73 | [0.47, 0.89]
##     72 |      0.65 | [0.44, 0.82]
##     73 |      0.56 | [0.40, 0.72]
##     75 |      0.38 | [0.24, 0.55]
##     76 |      0.30 | [0.15, 0.51]
##     79 |      0.12 | [0.03, 0.43]
##
## Adjusted for:
## *  games =      35.00
```

```
## * salary = 4895186.89
```

```
# Shows predicted hitInd values for different salaries based on an adjusted height and no of games
```

```
ggpredict(hitPredict2, terms = "salary [all]")
```

```
## # Predicted probabilities of hitInd
##
##  salary | Predicted |       95% CI
## ----------------------------------
##  507500 |      0.30 | [0.17, 0.47]
##  513850 |      0.30 | [0.18, 0.47]
##  519700 |      0.30 | [0.18, 0.47]
##   9e+05 |      0.32 | [0.19, 0.48]
## 1312000 |      0.33 | [0.20, 0.49]
##   4e+06 |      0.44 | [0.30, 0.58]
## 5825000 |      0.51 | [0.35, 0.67]
## 2.4e+07 |      0.95 | [0.58, 1.00]
##
## Adjusted for:
## *  games = 35.00
## * height = 74.00
```

Ref(Lüdecke, D.) We can see that the different predicted values lie well within the 95% confidence interval.

We can also compare values of hitInd from our data set with the predicted hitPredict values.

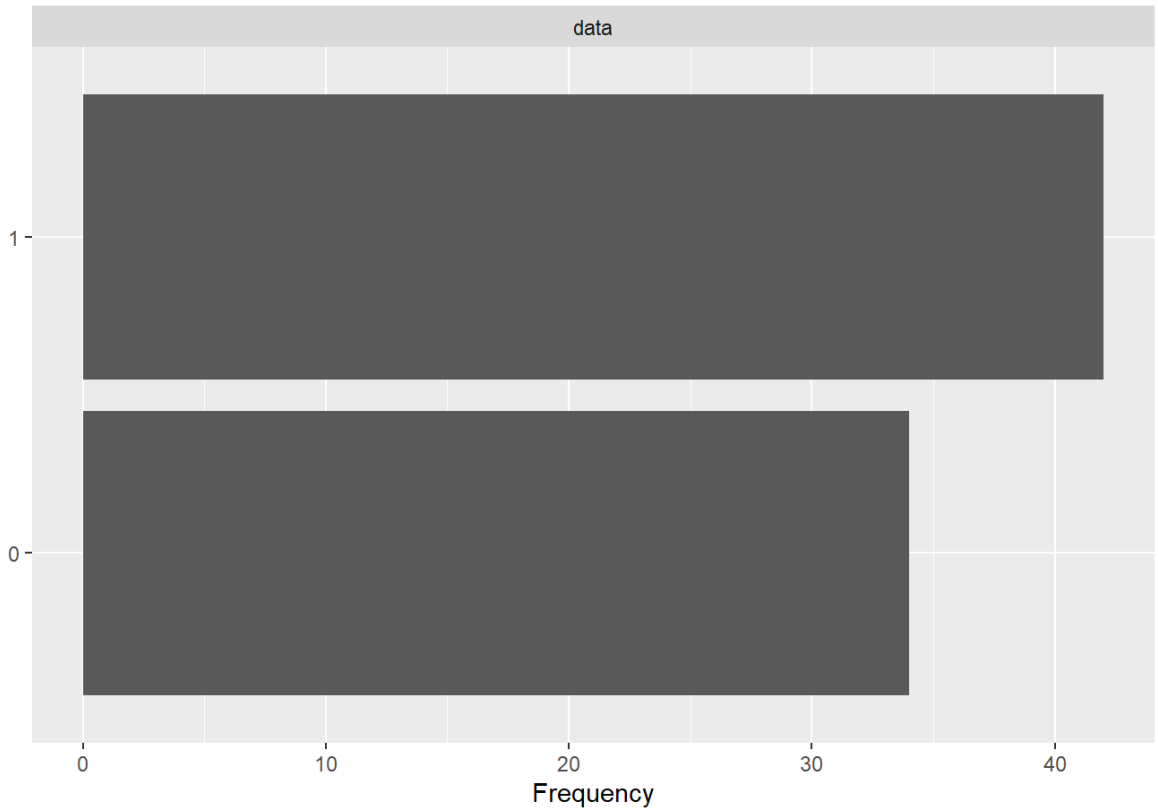
```
table(myTeamsData$hitInd)
```

```
##
##  0  1
## 34 42
```

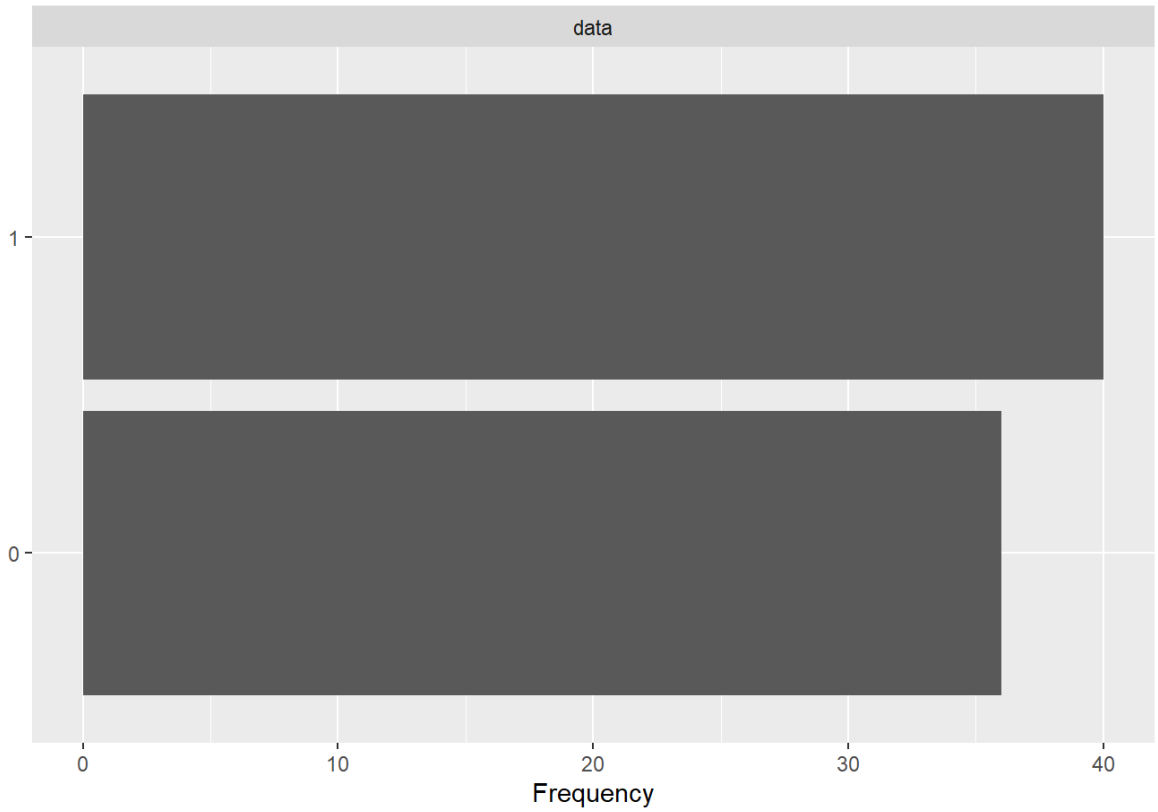
```
table(myTeamsData$hitPredict)
```

```
##
##  0  1
## 36 40
```

```
plot_bar(myTeamsData$hitInd)
```




```
plot_bar(myTeamsData$hitPredict)
```



```
myTeamsData$hitInd
```

```
##  [1] 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0
1 0 0 1

## [39] 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0
1 0 1 1

## Levels: 0 1
```

```
myTeamsData$hitPredict
```

```
##  [1] 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 0 0 0
1 0 0 1

## [39] 0 1 1 1 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 0 0 0
0 0 1 1

## Levels: 0 1
```

We can look at the data and see that all the predicted values are not matching with the correct hitInd value but still more than enough value are same and correct and we can use this model to predict the value for hitInd value.

References

1. Baseball Reference. Yearly League Leaders and Records for Youngest. [online] Available at: https://www.baseball-reference.com/leaders/Youngest_leagues.shtml
2. Baseball Reference. Yearly League Leaders and Records for Oldest. [online] Available at: https://www.baseball-reference.com/leaders/Oldest_leagues.shtml
3. Baseball Reference. 2015 MLB Batting Leaders. [online] Available at: https://www.baseball-reference.com/leagues/MLB/2015-batting-leaders.shtml
4. Baseball Reference. Minimum Salary. [online] Available at: https://www.baseball-reference.com/bullpen/Minimum_salary#:~:text=The%20minimum%20salary%20is%20the,raised%20by%2050%25%20to%20%24300%2C000
5. Gains, Cork. Business Insider. [online] Available at: https://www.businessinsider.com/highest-paid-mlb-players-2015-5?r=US&IR=T
6. Hayes, Adam. Heteroskedasticity. [online] Available at: https://www.investopedia.com/terms/h/heteroskedasticity.asp
7. Jackson, Matt. The 2015 MLB All-Skinny Team. [online] Available at: https://www.beyondtheboxscore.com/2016/1/6/10712314/2015-mlb-all-skinny-team-new-years-resolution
8. Kristan, Greg. Shortest MLB Players. [online] Available at: https://thestadiumreviews.com/resources/shortest-mlb-players/
9. Daniel, Lüdecke. Marginal Effects of Regression Models. [online] Available at: https://cran.csiro.au/web/packages/ggeffects/vignettes/ggeffects.html
10. Moritz, Steffen. ImputeTS [online] Available at: https://www.rdocumentation.org/packages/imputeTS/versions/3.1
11. Papp, Charlie. Longest Career in MLB History. [online] Available at: https://www.sportscasting.com/here-are-the-6-players-with-the-longest-careers-in-mlb-history/

12. Rdocumentation. Na.mean( ). [online] Available at: https://www.rdocumentation.org/packages/imputeTS/versions/3.1/topics/na.mean
13. Schork, Jaochim. Drop Levels in R. [online] Available at: https://statisticsglobe.com/droplevels-r-example/
14. Shepperd, Martin. Modern Data Book.
15. Superbaseball2020. Heaviest Baseball Players of All Time. [online] Available at: https://superbaseball2020blog.wordpress.com/2013/02/15/heaviest-baseball-player-of-all-time/
16. Wikipedia. 2015 MLB Season. [online] Available at: https://en.wikipedia.org/wiki/2015_Major_League_Baseball_season