Data_Programming_With_R_Final_Project

Siddhesh Bagwe - 21200534

2022-12-18

```
library(readr)
urlfile="https://raw.githubusercontent.com/adityarc19/IPL-analysis/main/data/2020.csv"
data=read_csv(url(urlfile),show_col_types = FALSE)
dim(data)
```

```
## [1] 150   8
```

```
str(data)
```

```
## spc_tbl_ [150 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Rank   : num [1:150] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Player : chr [1:150] "JJ Bumrah" "KL Rahul" "K Rabada" "Ishan Kishan" ...
##  $ Team   : chr [1:150] "Mumbai Indians" "Kings XI Punjab" "Delhi Capitals" "Mumbai Indians" ...
##  $ RAA    : num [1:150] 379 330 320 261 249 230 227 225 181 178 ...
##  $ Wins   : num [1:150] 1.281 1.113 1.082 0.881 0.842 ...
##  $ EFscore: num [1:150] 0.238 0.194 0.232 0.157 0.168 0.177 0.142 0.137 0.132 0.142 ...
##  $ Salary : chr [1:150] "$1,093,750" "$1,718,750" "$656,250" "$968,750" ...
##  $ Value  : chr [1:150] "$2,448,655" "$2,204,319" "$2,159,233" "$1,866,902" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Rank = col_double(),
##   ..   Player = col_character(),
##   ..   Team = col_character(),
##   ..   RAA = col_double(),
##   ..   Wins = col_double(),
##   ..   EFscore = col_double(),
##   ..   Salary = col_character(),
##   ..   Value = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Dataset Description

The Dataset gives the description of the player rankings in the Indian Premier League for the year 2020. The players are ranked with respect to RAA and wins for the data. The parameters are given below

1. Rank - Rank of the player
2. Player - Player Name
3. Team - Team for which the player plays
4. RAA - Runs Above Average(How well the player has performed with respect to average)

5. Wins - The importance of players in the wins.
6. EFscore - Score assigned to each player based on their performance.
7. Salary - Salary of the player each year
8. Value - Predicted value of the player based on performance

We will analyze the data to get an understanding of the player values and their performance.

Part 1: Analysis

```
summary(data)
```

```
##       Rank           Player             Team               RAA          
##  Min.   :  1.00   Length:150         Length:150         Min.   :-292.0000  
##  1st Qu.: 38.25   Class :character   Class :character   1st Qu.: -56.7500  
##  Median : 75.50   Mode  :character   Mode  :character   Median : -19.0000  
##  Mean   : 75.50                                         Mean   :   0.0133  
##  3rd Qu.:112.75                                         3rd Qu.:  61.2500  
##  Max.   :150.00                                         Max.   : 379.0000  
##       Wins             EFscore           Salary             Value          
##  Min.   :-0.9880   Min.   :0.00000   Length:150         Length:150         
##  1st Qu.:-0.1908   1st Qu.:0.01500   Class :character   Class :character   
##  Median :-0.0650   Median :0.05100   Mode  :character   Mode  :character   
##  Mean   : 0.0000   Mean   :0.06175                                         
##  3rd Qu.: 0.2060   3rd Qu.:0.09175                                         
##  Max.   : 1.2810   Max.   :0.23800
```

From the smmary of data we can see the min, max and the quantile data for each numerical parameters. We can see that the ranks go from 1-150 proving 150 coloumns. RAA data shows that the the player having the maximum RAA has 379 points while theplayer having lowest RAA is at -292 points with the mean at 0.0133. We also see the median at -19 which shows that there are more players with RAA below 0 than those above it.For the wins too we see the min at -0.9880 and max at 1.2810 with the mean ar 0.The EFscore is distributed from 0 to 0.238 with mean at 0.06175 and median at 0.05100.

```
sum(is.na(data))
```

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

```
data[rowSums(is.na(data))==1,]
```

```
## # A tibble: 1 x 8
##    Rank Player  Team             RAA   Wins EFscore Salary Value
##   <dbl> <chr>   <chr>          <dbl>  <dbl>   <dbl> <chr>  <chr>
## 1   140 DR Sams Delhi Capitals  -137 -0.464   0.006 <NA>   $-89,242
```

The dataset has 1 Null value which is the Salary for the player "DR Sams". We can replace the null value by the actual value of the player to remove the null value.

```
data$Salary[data$Player=='DR Sams']=data$Value[data$Player=='DR Sams']
sum(is.na(data))
```

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

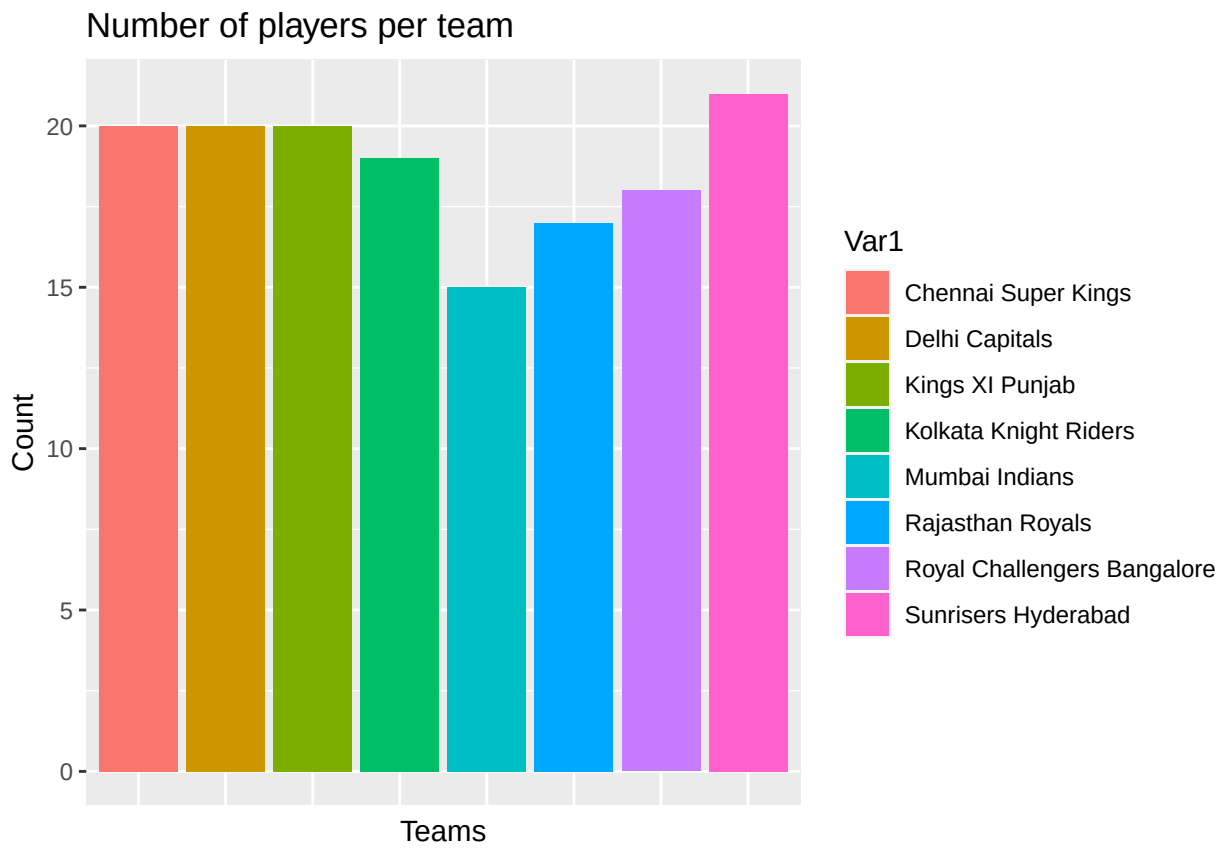
As we have replaced the null value with the actual value we can now see that our dataset has no null values left.

```
x=table(data$Team)
x
```

```
##
##          Chennai Super Kings              Delhi Capitals
##                           20                          20
##              Kings XI Punjab       Kolkata Knight Riders
##                           20                          19
##               Mumbai Indians            Rajasthan Royals
##                           15                          17
## Royal Challengers Bangalore         Sunrisers Hyderabad
##                           18                          21
```

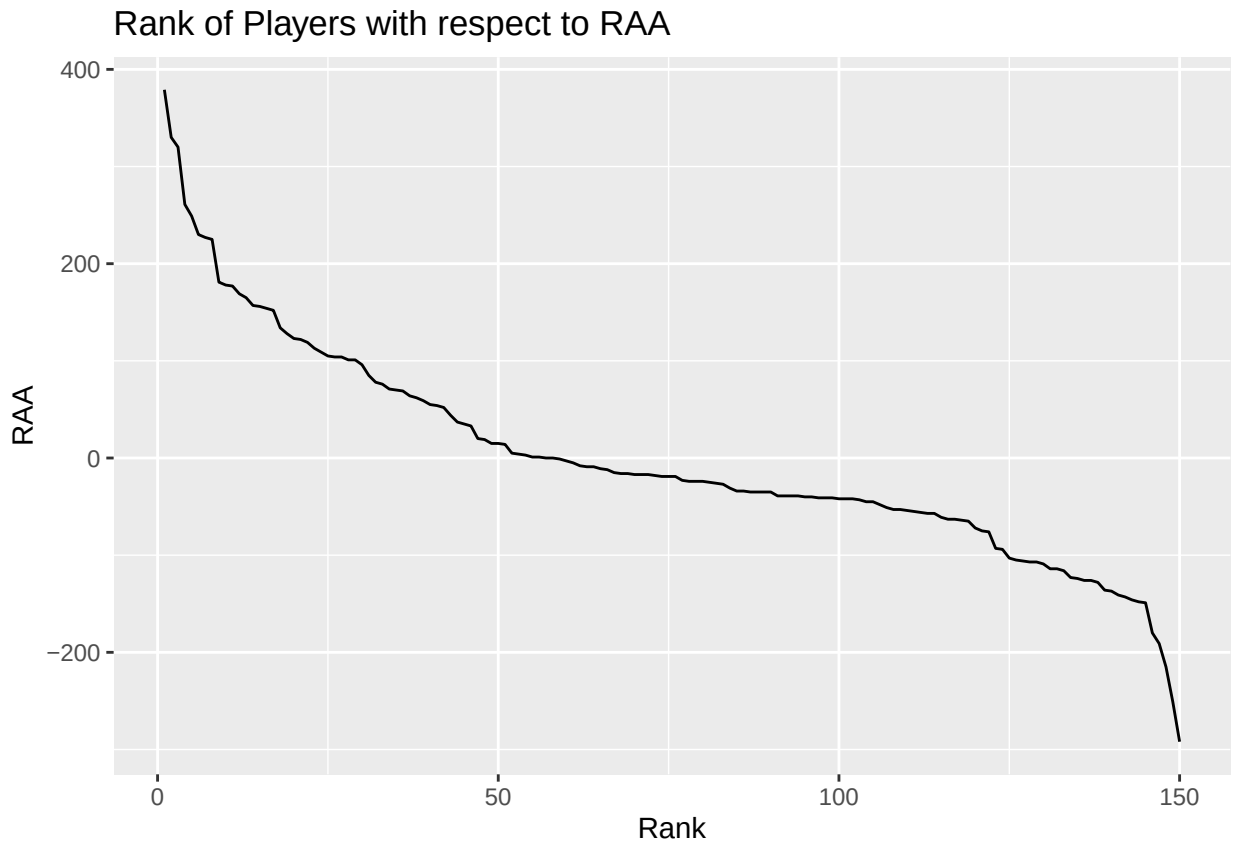
```
x=data.frame(x)
```

```
library(ggplot2)
ggplot(data=x,aes(x=Var1,y=Freq,fill=Var1)) +
  geom_bar(stat="identity") +
  labs(x = "Teams", y = "Count", title="Number of players per team") +
theme(axis.text.x=element_blank(),
      axis.ticks.x=element_blank())
```

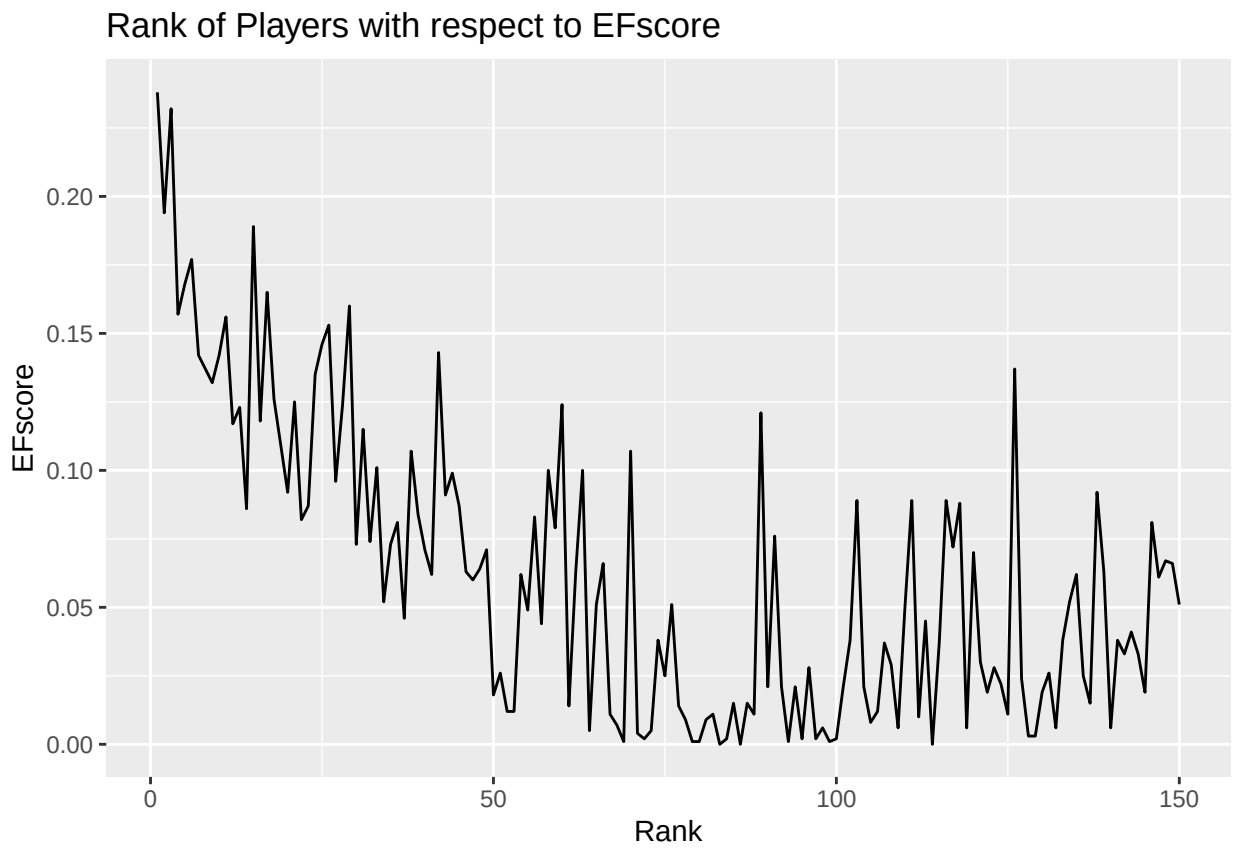


The data of number of top ranked players in the top 150 with respect to the teams is given below. We see Sunrisers Hyderabad having the most no. of players(21) in the top 150 with Mumbai Indians having the lowest count with 15.

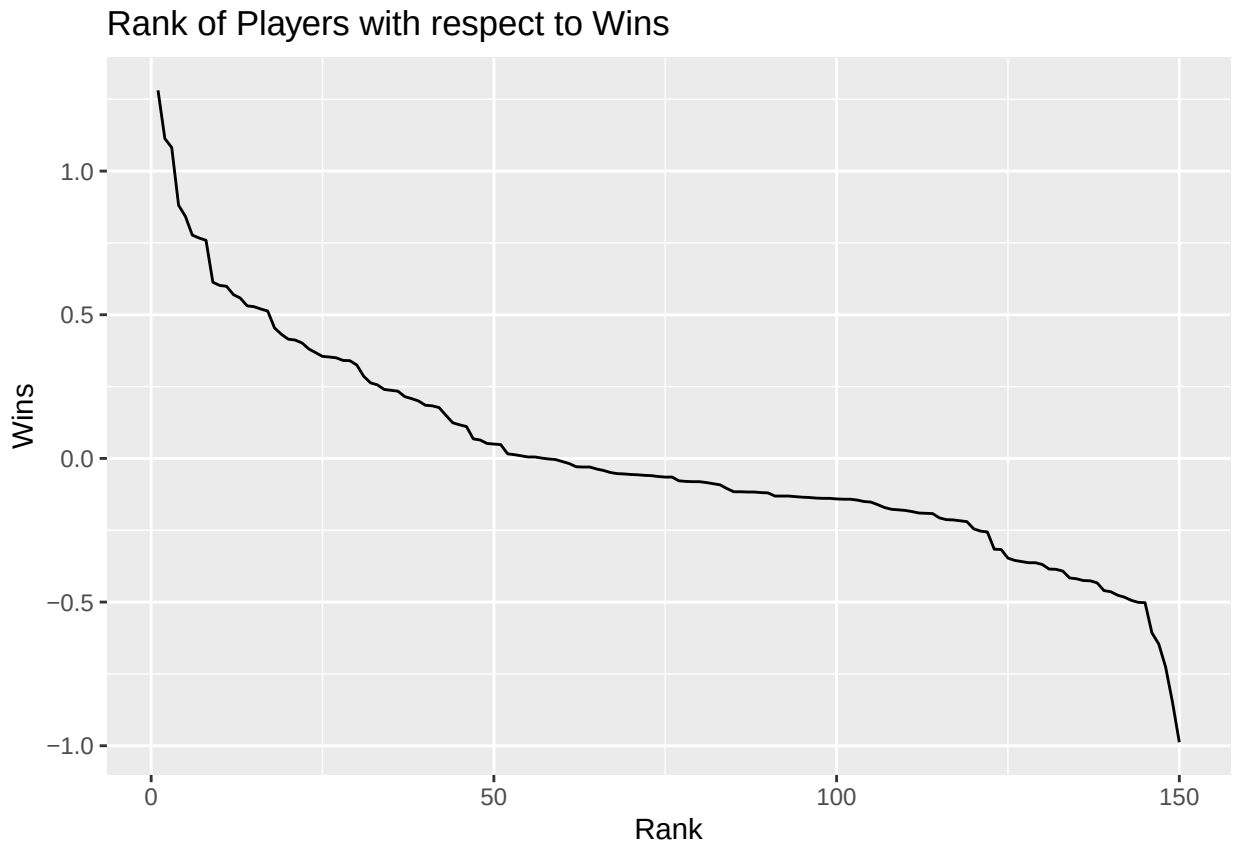
```
library(ggplot2)
ggplot(data=data,aes(x=Rank,y=RAA)) +
  geom_line()+labs(title = "Rank of Players with respect to RAA")
```




```
ggplot(data=data,aes(x=Rank,y=EFscore)) +
  geom_line()+labs(title = "Rank of Players with respect to EFscore")
```




```
ggplot(data=data,aes(x=Rank,y=Wins)) +
  geom_line()+labs(title = "Rank of Players with respect to Wins")
```

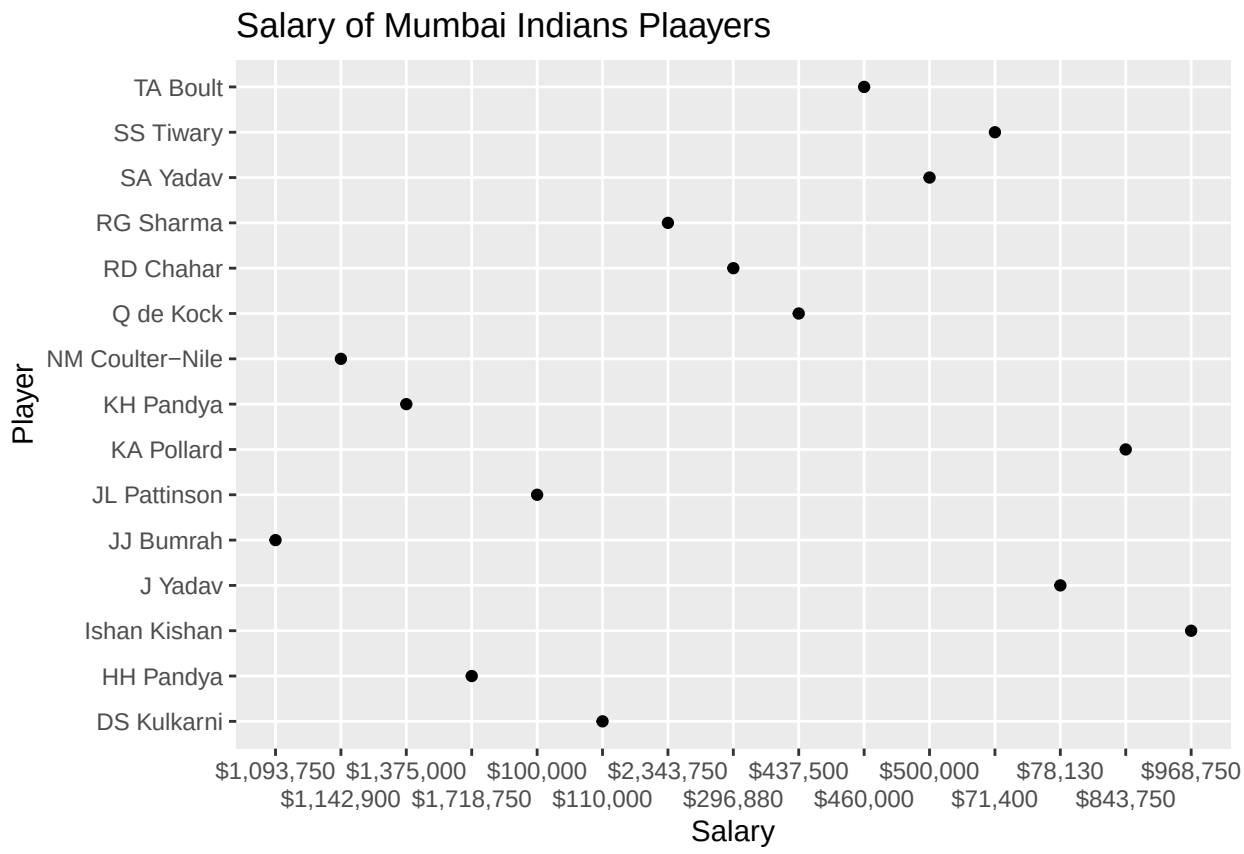


We see a negative correlation between the rank and RAA value. As RAA decreases the Rank of the player increases. The player at rank 1 has an RAA of approximately 400 while the player at rank 150 has RAA below -200.

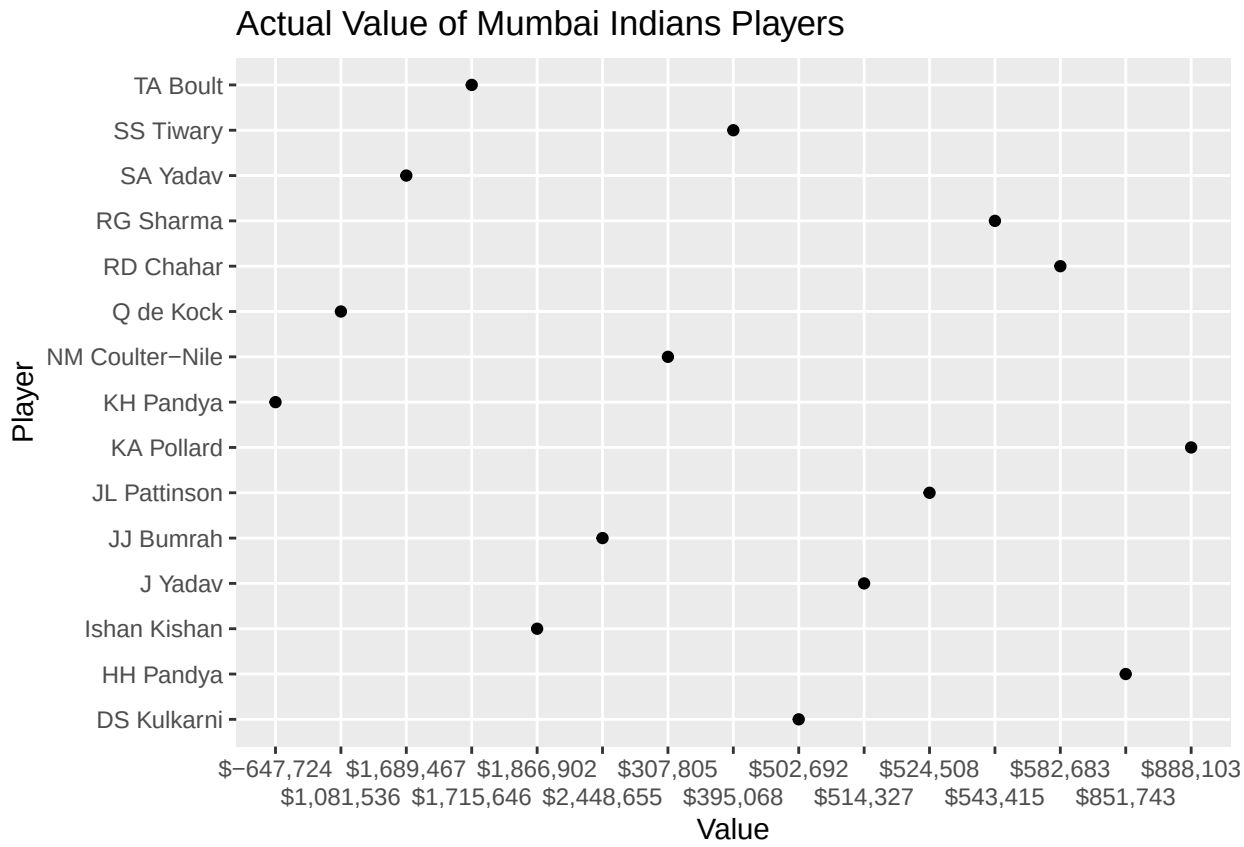
For the plot of Rank and Efscore the relationship is not linear as EFscore is assigned to the player with respect to RAA and wins combined. But we generally see that top ranked players have a higher EFscore. Some players have better EFscore but are ranked less as per the plot. This happens because their RAA score is less.

Player's performance also has a negative correlation with respect to the Rank. This means that top ranked players have more impactful performances which helped their teams to win.

```
y=data[data$Team=='Mumbai Indians',]
ggplot(data=y,aes(x=Salary,y=Player)) +  scale_x_discrete(guide = guide_axis(n.dodge = 2))+
  geom_point()+labs(title = "Salary of Mumbai Indians Plaayers")
```




```
ggplot(data=y,aes(x=Value,y=Player)) +  scale_x_discrete(guide = guide_axis(n.dodge = 2))+
  geom_point()+labs(title = "Actual Value of Mumbai Indians Players")
```



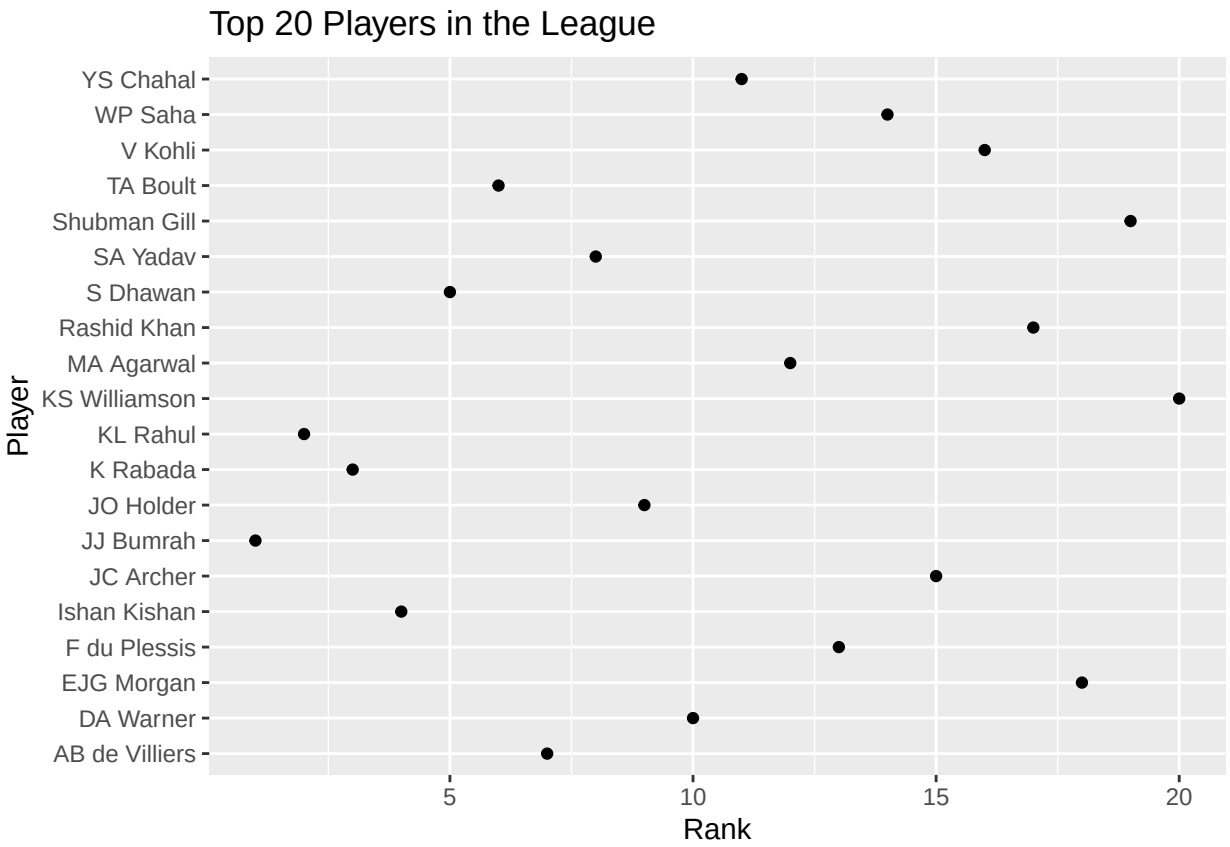
The above plots show the salary that the team Mumbai Indians pay to the players versus their actual value. Some players are paid more than their value while some are underpaid. For example(Jasprit Bumrah who is the top ranked player earns $1,093,750 while his value is more than double $2,448,655). We get the analysis of the wages according to the performance so that they are paid accordingly in the next year. Similarly we can check for different teams to get the analysis.

```
z=data[1:20,]
table(z$Team)
```

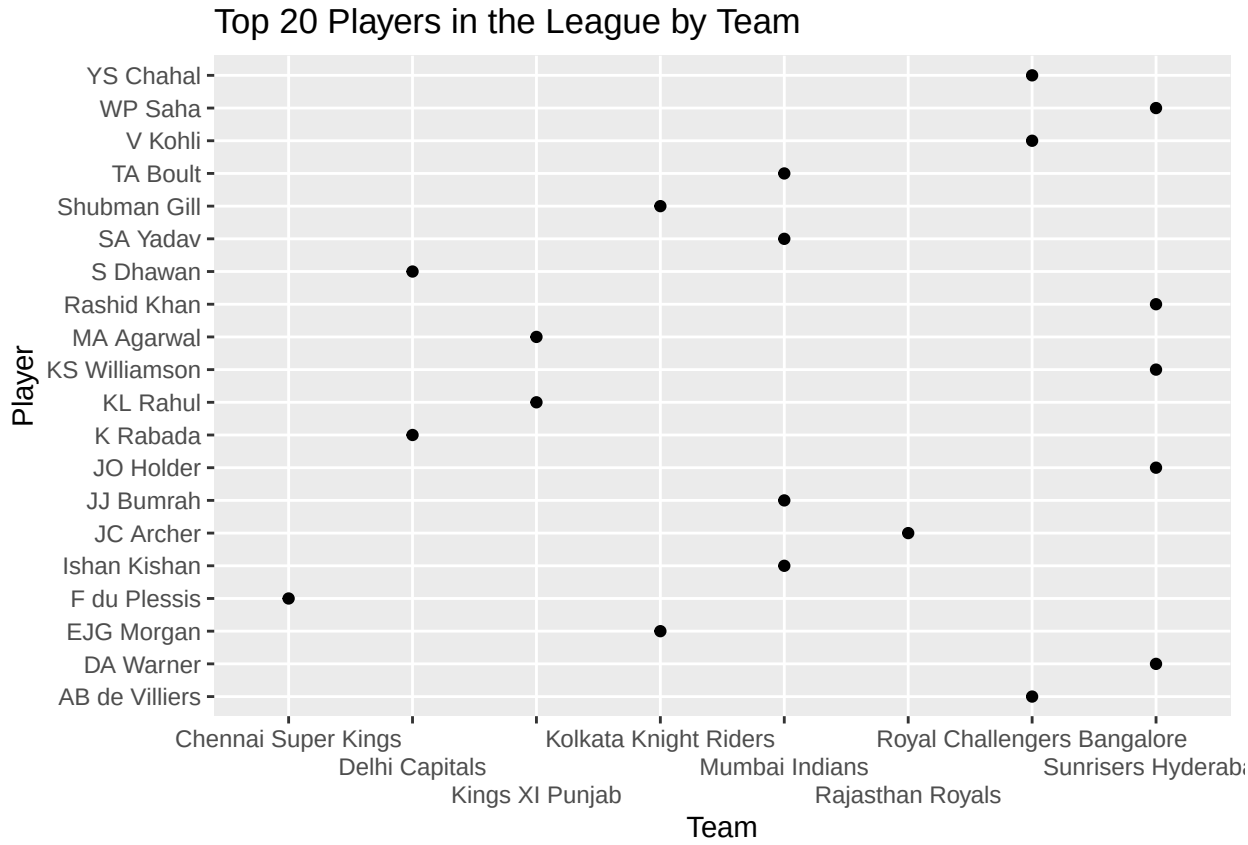
```
##
##         Chennai Super Kings              Delhi Capitals
##                           1                           2
##             Kings XI Punjab       Kolkata Knight Riders
##                           2                           2
##              Mumbai Indians           Rajasthan Royals
##                           4                           1
## Royal Challengers Bangalore         Sunrisers Hyderabad
##                           3                           5
```

Considering the top 20 players of the year we see that Sunrisers Hyderabad has the maximum players in top 20 with 5 followed by Mumbai Indians. Rajasthan Royals and Chennai Super Kings have the lowest with just 1 player in top 20.

```
library(ggplot2)
ggplot(data=z,aes(x=Rank,y=Player)) +
  geom_point()+labs(title = "Top 20 Players in the League")
```




```
ggplot(data=z,aes(x=Team,y=Player)) +
  geom_point()+scale_x_discrete(guide=guide_axis(n.dodge=3))+labs(title = "Top 20 Players in the League
```



The plot shows the top 20 players in the league by their Rank and which team they belong to. We can check this with the table above.

Observations of EDA

1. There was one null value in the data.
2. The parameters RAA and Wins are linearly correlated to the Rank.
3. Data can be used to check if the players are under or over-paid.
4. The number of players per team is found.

Part 2: R Package

For this part we will use the "caret" package in R.

The caret package (short for Classification And Regression Training) contains functions to streamline the model training process for complex regression and classification problems. The package utilizes a number of R packages but tries not to load them all at package start-up (by removing formal package dependencies, the package startup time can be greatly decreased). The package "suggests" field includes 32 packages.

Some of the features of caret include:

1. Pre-Processing: Where data is pre-processed
2. Data splitting: Splitting the training data into two similar categorical data sets is done.
3. Training Model: caret provides many packages for machine learning algorithms. Resampling for model tuning The model can be tuned using repeated k-fold, k-fold, etc. Also, the parameter can be tuned using 'tuneLength.'
4. Making Predictions: caret has the function predict which gives us the prediction of the model.

We will use this package to work on regression for our model.We will predict the rank of the player based on "RAA","Wins" & "EFscore".

First we need to install the caret package. We can do this using the following command install.packages("caret")

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

```
## Loading required package: lattice
```

Once the package is loaded we will use it work on our model.

Step 1: Pre-Processing:

In this step we will preprocess the data and standardize it to get the proper data for further use. For this we will use the function "preProcess".

```
preprocess = preProcess(data[,4:6], method=c("center", "scale"))
data1 = predict(preprocess, data[,4:6])
Rank=data$Rank
data1=cbind(Rank,data1)
summary(data1)
```

```
##       Rank             RAA                Wins              EFscore
##  Min.   :  1.00   Min.   :-2.6366   Min.   :-2.6401   Min.   :-1.1520
##  1st Qu.: 38.25   1st Qu.:-0.5125   1st Qu.:-0.5097   1st Qu.:-0.8722
##  Median : 75.50   Median :-0.1717   Median :-0.1737   Median :-0.2005
##  Mean   : 75.50   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:112.75   3rd Qu.: 0.5529   3rd Qu.: 0.5505   3rd Qu.: 0.5598
##  Max.   :150.00   Max.   : 3.4219   Max.   : 3.4231   Max.   : 3.2884
```

Here you can see that we have created the function preprocess to standardize our x data (making the mean 0). As we are predicting Rank the x data becomes the coloumns "RAA","Wins" & "EFscore". We first standardize this data and add the Rank to it to create dataset data1. From the summary we can see that the data is now standardized and we will now use this dataset in the next steps.

Step 2. Data splitting:

In this step we split our dataset into train and test sets. To do this the function "createDataPartition" is used.

```
set.seed(102)
train = createDataPartition(
  y = data1$Rank,
  p = .80,
  list = FALSE)
training = data1[ train,]
testing  = data1[-train,]
nrow(training)
```

```
## [1] 122
```

```
nrow(testing)
```

```
## [1] 28
```

We have first created the train function to define our split parameters. We choose Value as the parameter to be predicted and the train size as 80% which means test size will be 20%. Further we use this function on our dataset to get training and testing data. We can see the dimensions of the training and testing data to prove that the data is split in the ratio 80:20.

Step 3: Training the Model.

We now use the training data to fitthe model. The model is built in caret using the "train" function.

```
model= train(training[,2:4], training$Rank, method = "lm" )
summary(model)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35.86 -12.93  -1.56  11.97  58.39
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   75.998      1.436  52.931   <2e-16 ***
## RAA         -644.397    579.019  -1.113    0.268
## Wins         601.654    578.929   1.039    0.301
## EFscore        3.709      2.043   1.815    0.072 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.82 on 118 degrees of freedom
## Multiple R-squared:  0.8705, Adjusted R-squared:  0.8672
## F-statistic: 264.4 on 3 and 118 DF,  p-value: < 2.2e-16
```

Using the train function we have built the model for the data. Though this is an example of linear regression we can also carry out other regression and classification models using caret. The final step of the model would be making predictions and testing the accuracy. From the model summary we see the R-squared value as 0.8705. We will check this using predictions.

Step 4 : Making Predictions

In the final step we will make the predictions using the "predict" function.

```
predictions = predict(model, testing)
SSE <- sum((testing$Rank - predictions)^2)
SST <- sum((testing$Rank - mean(testing$Rank))^2)
r2 <- 1 - SSE / SST
r2
```

```
## [1] 0.8962345
```

We see that the value of R-squared using the prediction (0.896) is pretty close to the model summary (0.8705).

This is how we can use the Caret function to create a model. We can create different regression and classification models using Caret. There are over 230 models included in the package including various tree-based models, neural nets, deep learning and much more. We have explained some functions of the caret function through our model. Similarly we can use different models in Caret

Part 3: Functions/Programming

In this part we will use S3 class to provide analysis for the data. As wee have seen the parameters Rank, Player, Team, RAA, Wins and Value are the most important ones for the analysis of the dataset so we will create a s3 class with this parameters.

```
a= list(Rank=data$Rank[1:10], Player=data$Player[1:10],
        Team=data$Team[1:10], RAA=data$RAA[1:10], Wins=data$Wins[1:10], Value=data$Value[1:10])
class(a)='players'
a
```

```
## $Rank
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $Player
##  [1] "JJ Bumrah"      "KL Rahul"       "K Rabada"       "Ishan Kishan"
##  [5] "S Dhawan"       "TA Boult"       "AB de Villiers" "SA Yadav"
##  [9] "JO Holder"      "DA Warner"
##
## $Team
##  [1] "Mumbai Indians"              "Kings XI Punjab"
##  [3] "Delhi Capitals"              "Mumbai Indians"
##  [5] "Delhi Capitals"              "Mumbai Indians"
##  [7] "Royal Challengers Bangalore" "Mumbai Indians"
##  [9] "Sunrisers Hyderabad"         "Sunrisers Hyderabad"
##
## $RAA
##  [1] 379 330 320 261 249 230 227 225 181 178
##
## $Wins
##  [1] 1.281 1.113 1.082 0.881 0.842 0.777 0.767 0.759 0.613 0.602
##
## $Value
##  [1] "$2,448,655" "$2,204,319" "$2,159,233" "$1,866,902" "$1,810,181"
##  [6] "$1,715,646" "$1,701,102" "$1,689,467" "$1,477,128" "$1,461,129"
##
## attr(,"class")
## [1] "players"
```

We can see that when we print a s3 class it gives out the output as a vector for all the values. We will define a print function to give the output a better look.

```
print.players <- function(x) {
cat('Rank :', x$Rank,'\n')
cat('Player :', x$Player,'\n')
cat('Team :', x$Team,'\n')
cat('RAA :', x$RAA,'\n')
cat('Wins :',x$Wins,'\n')
cat('Value :', x$Value,'.')
}
print(a)
```

```
## Rank : 1 2 3 4 5 6 7 8 9 10
```

```
## Player : JJ Bumrah KL Rahul K Rabada Ishan Kishan S Dhawan TA Boult AB de Villiers SA Yadav JO Holde
## Team : Mumbai Indians Kings XI Punjab Delhi Capitals Mumbai Indians Delhi Capitals Mumbai Indians Ro
## RAA : 379 330 320 261 249 230 227 225 181 178
## Wins : 1.281 1.113 1.082 0.881 0.842 0.777 0.767 0.759 0.613 0.602
## Value : $2,448,655 $2,204,319 $2,159,233 $1,866,902 $1,810,181 $1,715,646 $1,701,102 $1,689,467 $1,4
```

We see that the print function gives a neater look to the class.

For analysis of the function we will create 3 summary functions for the class to give different analysis. First will give us the numerical analysis of the data. The second gives us the player Rank and the third will give us number of players per team.

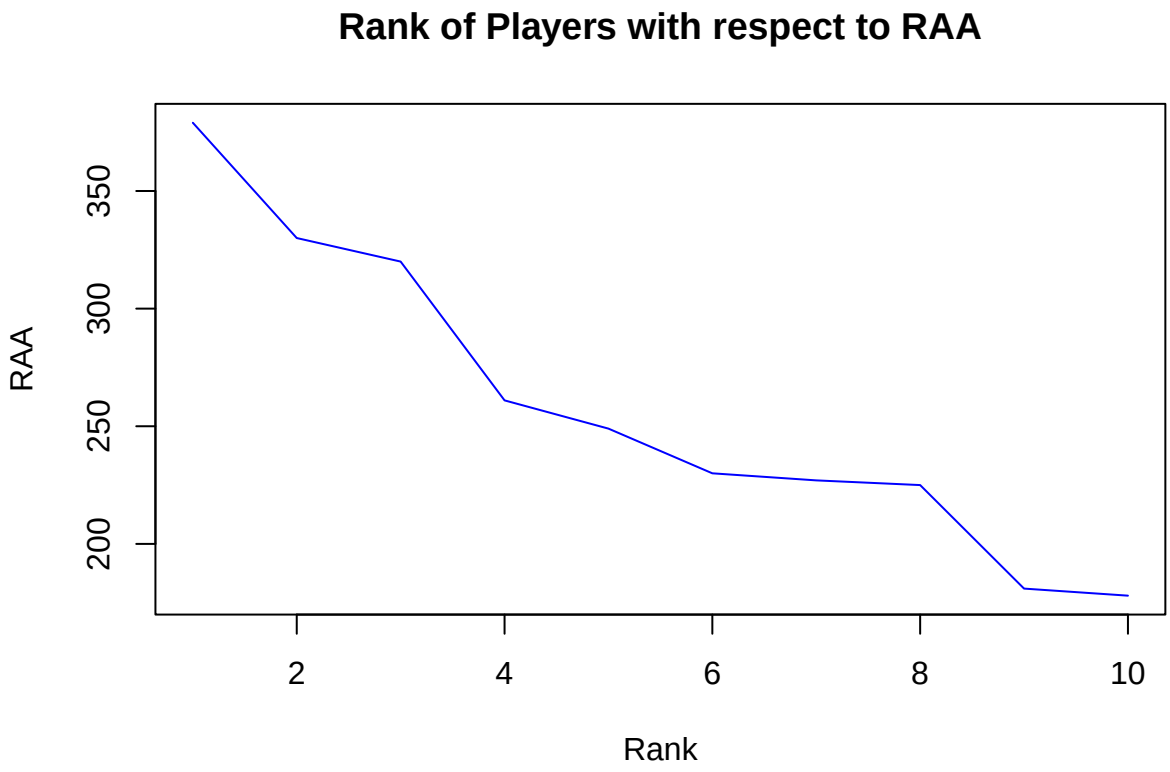
```
summary.players <- function(x){
  cat('Numerical Summary of Data\n\n')
  cat('Rank\n')
  cat('Min :', min(x$Rank), 'Max :', max(x$Rank),'Mean :',mean(x$Rank), 'Median :',median(x$Rank),'\n')
  cat('RAA\n')
  cat('Min :', min(x$RAA), 'Max :', max(x$RAA),'Mean :',mean(x$RAA), 'Median :',median(x$RAA),'\n')
  cat('Wins\n')
  cat('Min :', min(x$Wins), 'Max :', max(x$Wins),'Mean :',mean(x$Wins), 'Median :',median(x$Wins),'\n')
  cat('\n')
  cat('Number of Players per Team\n')
  print(table(x$Team))
}

summary(a)
```

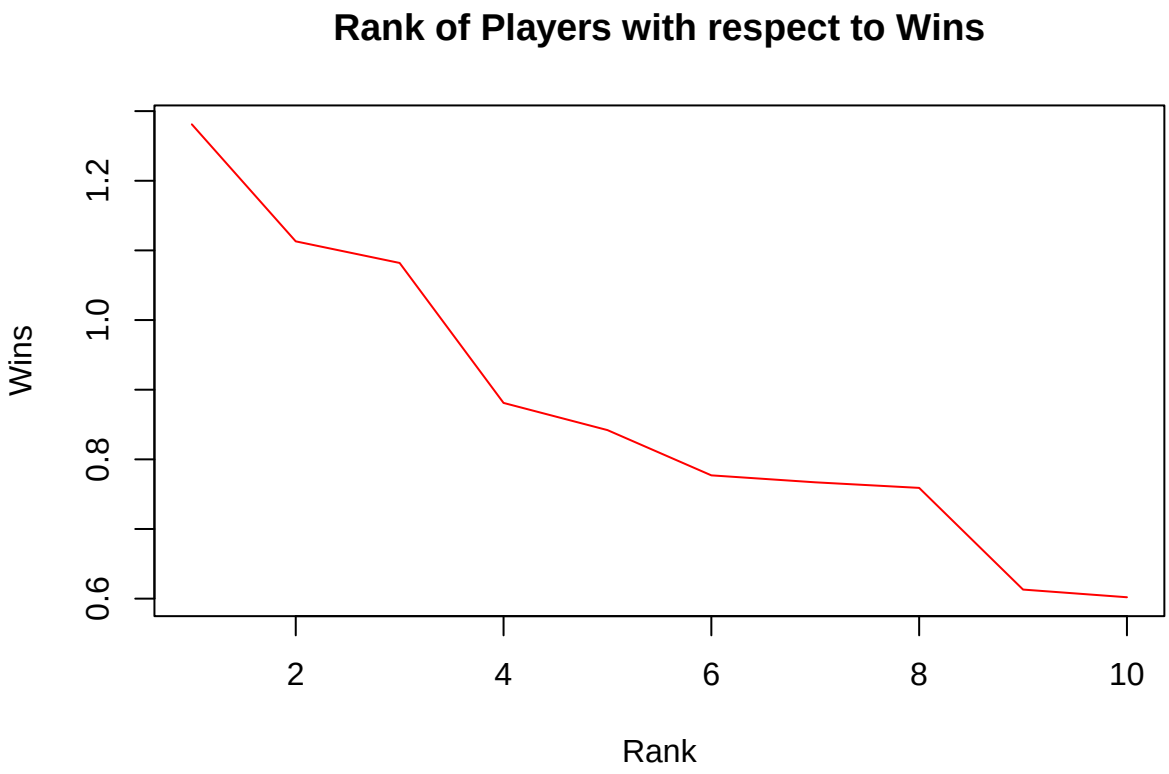
```
## Numerical Summary of Data
##
## Rank
## Min : 1 Max : 10 Mean : 5.5 Median : 5.5
## RAA
## Min : 178 Max : 379 Mean : 258 Median : 239.5
## Wins
## Min : 0.602 Max : 1.281 Mean : 0.8717 Median : 0.8095
##
## Number of Players per Team
##
##              Delhi Capitals             Kings XI Punjab
##                           2                           1
##              Mumbai Indians Royal Challengers Bangalore
##                           4                           1
##         Sunrisers Hyderabad
##                           2
```

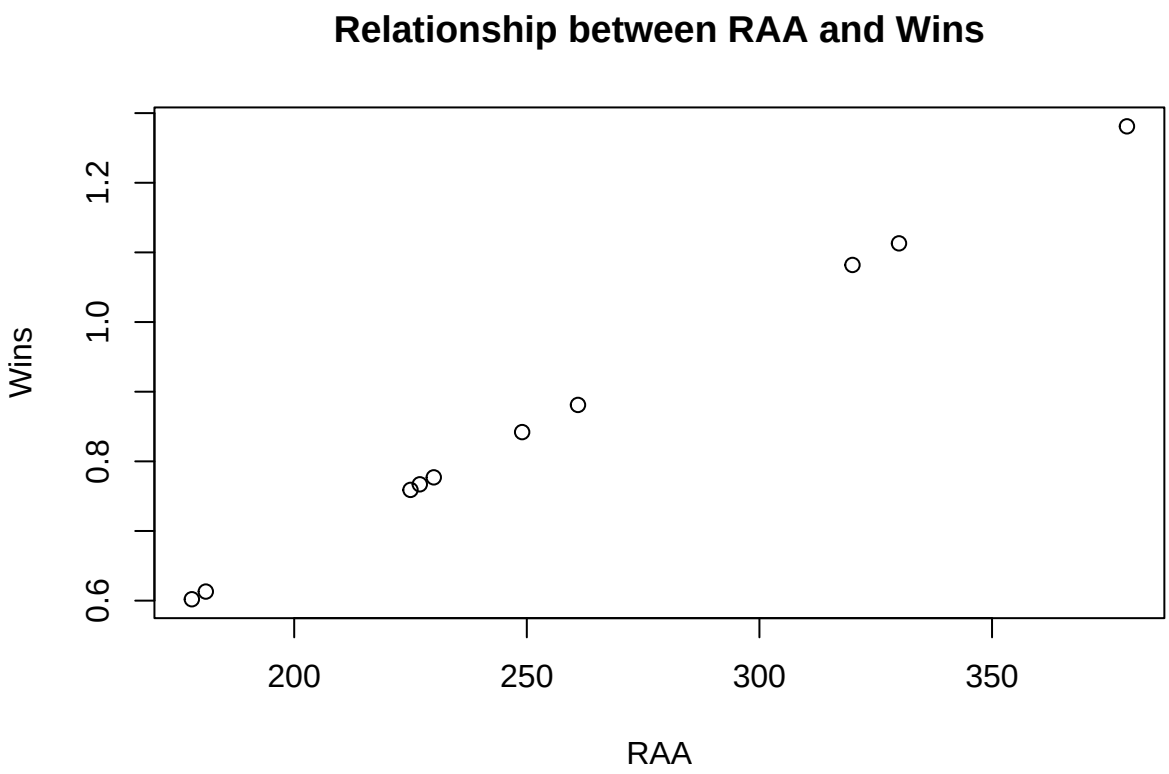
Here we have created a Summary function for class players which gives the numerical analysis of the data as well as the number of players belonging to each team.

```
plot.players<- function(x){
  plot(x$Rank,x$RAA,main="Rank of Players with respect to RAA",
ylab="RAA",xlab="Rank",type="l",col="blue")
  plot(x$Rank,x$Wins,main="Rank of Players with respect to Wins",
ylab="Wins",xlab="Rank",type="l",col="red")
```

```
  plot(x$RAA,x$Wins,main="Relationship between RAA and Wins",
ylab="Wins",xlab="RAA")
}
plot(a)
```




Rank of Players with respect to Wins
Wins
0.6
0.8
1.0
1.2
2
4
6
8
10
Rank



The plot function here gives us three plots

1. Rank of Players with respect to RAA.
2. Rank of Players with respect to Wins.
3. Relationship between RAA and Wins

We will use the print, summary and plot functions on other lists with the same class. We will use players from rank 21-30 for this.

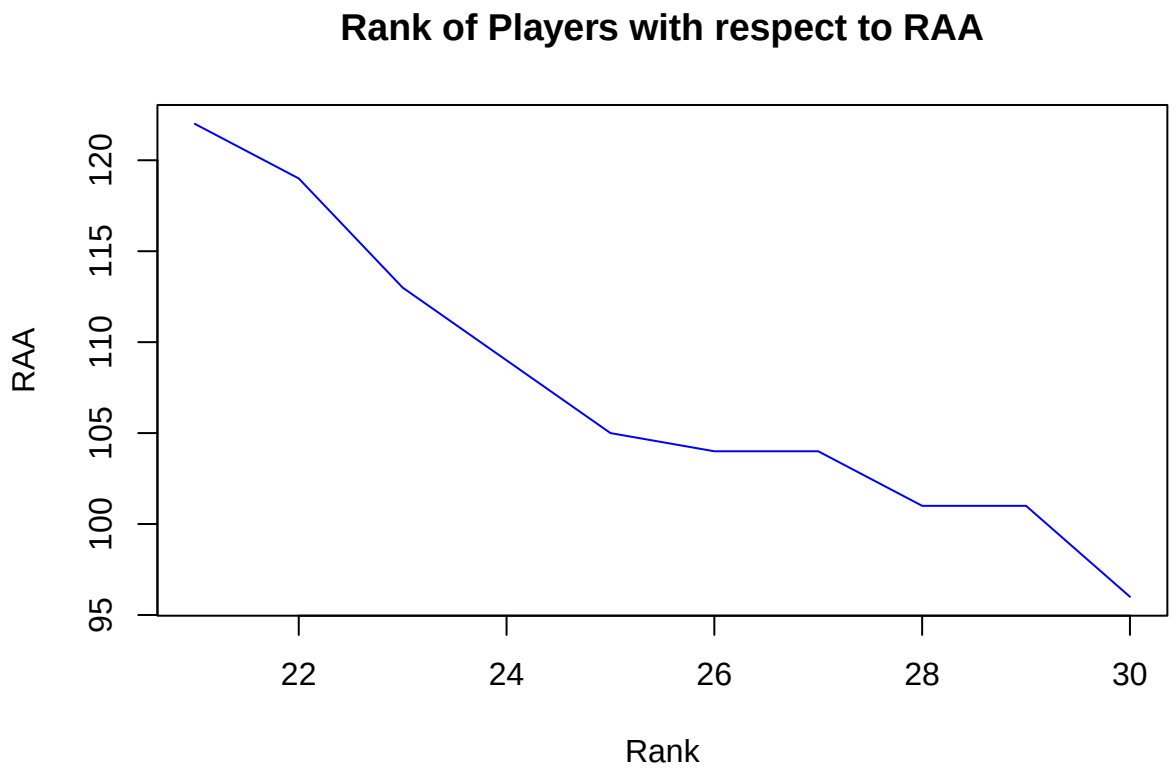
```
b= list(Rank=data$Rank[21:30], Player=data$Player[21:30],
        Team=data$Team[21:30], RAA=data$RAA[21:30], Wins=data$Wins[21:30], Value=data$Value[21:30])
class(b)='players'
print(b)
```

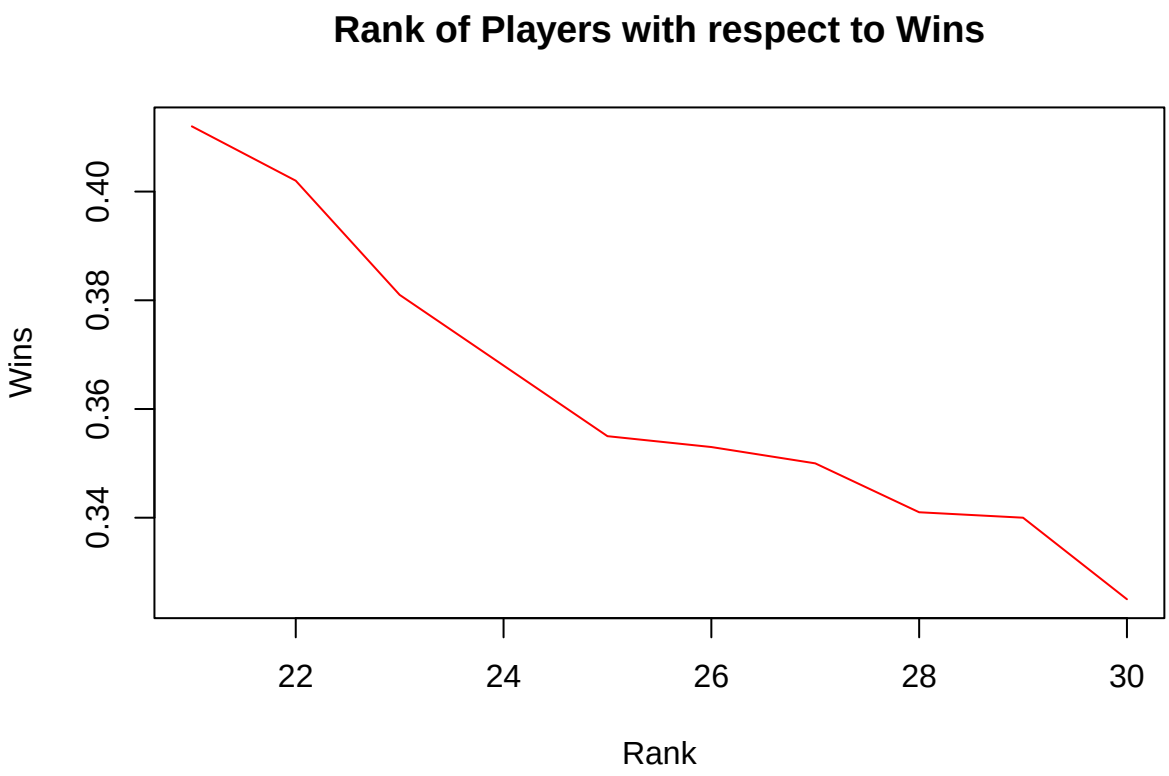
```
## Rank : 21 22 23 24 25 26 27 28 29 30
## Player : SS Iyer RD Gaikwad CH Gayle CV Varun Mohammed Shami MP Stoinis AT Rayudu Q de Kock A Nortje
## Team : Delhi Capitals Chennai Super Kings Kings XI Punjab Kolkata Knight Riders Kings XI Punjab Delhi
## RAA : 122 119 113 109 105 104 104 101 101 96
## Wins : 0.412 0.402 0.381 0.368 0.355 0.353 0.35 0.341 0.34 0.325
## Value : $1,184,797 $1,170,253 $1,139,711 $1,120,804 $1,101,897 $1,098,988 $1,094,625 $1,081,536 $1,08
```

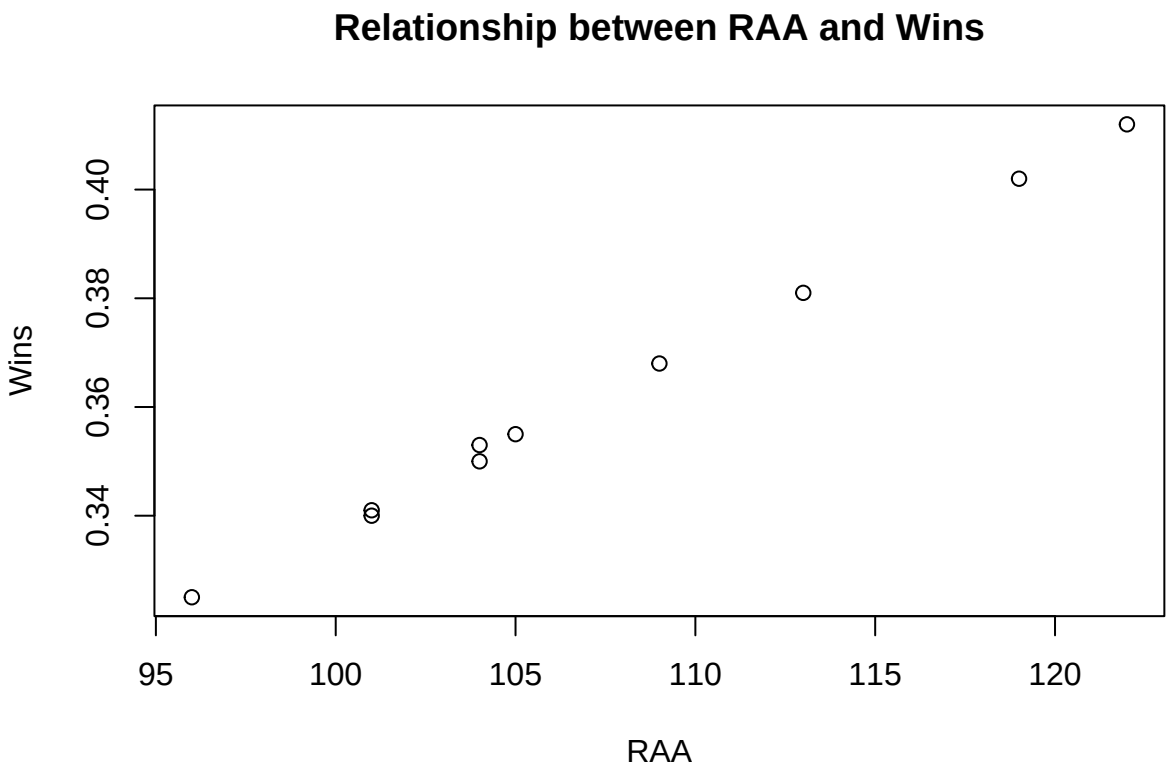
```
summary(b)
```

```
## Numerical Summary of Data
##
## Rank
## Min : 21 Max : 30 Mean : 25.5 Median : 25.5
## RAA
## Min : 96 Max : 122 Mean : 107.4 Median : 104.5
## Wins
## Min : 0.325 Max : 0.412 Mean : 0.3627 Median : 0.354
##
## Number of Players per Team
##
##   Chennai Super Kings        Delhi Capitals       Kings XI Punjab
##                     3                     3                     2
## Kolkata Knight Riders        Mumbai Indians
##                     1                     1
```

```
plot(b)
```




Rank of Players with respect to Wins
Wins
0.34
0.36
0.38
0.40
22
24
26
28
30
Rank



So we can use the Functions for any S3 class of this data. This will help us understanding the data and carrying out the analysis for the data of any other year.