R Project

Arjun M,Mahesh D,Ruthwik HM

16/10/2019

First we load the required dataset.

```
library(readr)
df <- read_csv('NBA_PLAYERS.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   TEAM = col_character(),
##   NAME = col_character(),
##   URL = col_character(),
##   POSITION = col_character(),
##   AGE = col_character(),
##   COLLEGE = col_character(),
##   SALARY = col_character(),
##   FGM_FGA = col_character(),
##   THM_THA = col_character(),
##   FTM_FTA = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

We can now view the dataset.

```
View(df)
```

To check the number of columns and type of data in each column we can use structure function.

```
str(df)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 550 obs. of  30 variables:
##  $ TEAM           : chr  "Boston Celtics" "Boston Celtics" "Boston Celtics" "Boston Celtic
s" ...
##  $ NAME           : chr  "Aron Baynes" "Justin Bibbs" "Jabari Bird" "Jaylen Brown" ...
##  $ EXPERIENCE     : num  6 0 1 2 1 2 8 11 7 0 ...
##  $ URL            : chr  "http://www.espn.com/nba/player/_/id/2968439" "http://www.espn.co
m/nba/player/_/id/3147500" "http://www.espn.com/nba/player/_/id/3064308" "http://www.espn.com/
nba/player/_/id/3917376" ...
##  $ POSITION       : chr  "SF" "G" "SG" "F" ...
##  $ AGE            : chr  "31" "22" "24" "21" ...
##  $ HT             : num  208 196 198 201 198 ...
##  $ WT             : num  117.7 99.5 89.6 99.5 92.8 ...
##  $ COLLEGE        : chr  "Washington State" "Virginia Tech" "California" "California" ...
##  $ SALARY         : chr  "5,193,600" "Not signed" "1,349,464" "5,169,960" ...
##  $ PPG_LAST_SEASON: num  6 0 3 14.5 1 1.4 2 12.9 24.4 0 ...
##  $ APG_LAST_SEASON: num  1.1 0 0.6 1.6 0 0.2 0 7.4 5.1 0 ...
##  $ RPG_LAST_SEASON: num  5.4 0 1.5 4.9 0.5 0.4 1 1.1 3.8 0 ...
##  $ PER_LAST_SEASON: num  12.09 0 12.18 13.69 -4.82 ...
##  $ PPG_CAREER     : num  5.4 0 3 10.4 1 1.6 15.6 14.2 22 0 ...
##  $ APG_CAREER     : num  0.7 0 0.6 1.2 0 0.2 3.4 8.6 5.5 0 ...
##  $ RGP_CAREER     : num  4.4 0 1.5 3.8 0.5 0.5 4.2 1.2 3.4 0 ...
##  $ GP             : num  376 0 13 148 2 47 517 718 441 0 ...
##  $ MPG            : num  15 0 8.8 23.6 1.5 5.8 31.3 33.2 33.9 0 ...
##  $ FGM_FGA        : chr  "2.2-4.3" "0" "1.2-2.0" "3.8-8.3" ...
##  $ FGP            : num  0.502 0 0.577 0.461 0.5 0.418 0.444 0.525 0.462 0 ...
##  $ THM_THA        : chr  "0.0-0.1" "0" "0.2-0.5" "1.1-3.0" ...
##  $ THP            : num  0.143 0 0.429 0.379 0 0.294 0.368 0.37 0.388 0 ...
##  $ FTM_FTA        : chr  "1.0-1.3" "0" "0.5-1.0" "1.6-2.4" ...
##  $ FTP            : num  0.802 0 0.462 0.658 0 0.71 0.82 0.75 0.875 0 ...
##  $ APG            : num  0.7 0 0.6 1.2 0 0.2 3.4 3.2 5.5 0 ...
##  $ BLKPG          : num  0.5 0 0.1 0.3 0 0 0.4 1.2 0.3 0 ...
##  $ STLPG          : num  0.2 0 0.2 0.7 0 0.1 1 0.8 1.3 0 ...
##  $ TOPG           : num  0.8 0 0.6 1.3 0.5 0.1 2 1.6 2.7 0 ...
##  $ PPG            : num  5.4 0 3 10.4 1 1.6 15.6 14.2 22 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   TEAM = col_character(),
##   ..   NAME = col_character(),
##   ..   EXPERIENCE = col_double(),
##   ..   URL = col_character(),
##   ..   POSITION = col_character(),
##   ..   AGE = col_character(),
##   ..   HT = col_double(),
##   ..   WT = col_double(),
##   ..   COLLEGE = col_character(),
##   ..   SALARY = col_character(),
##   ..   PPG_LAST_SEASON = col_double(),
##   ..   APG_LAST_SEASON = col_double(),
##   ..   RPG_LAST_SEASON = col_double(),
##   ..   PER_LAST_SEASON = col_double(),
##   ..   PPG_CAREER = col_double(),
##   ..   APG_CAREER = col_double(),
##   ..   RGP_CAREER = col_double(),
##   ..   GP = col_double(),
##   ..   MPG = col_double(),
##   ..   FGM_FGA = col_character(),
##   ..   FGP = col_double(),
##   ..   THM_THA = col_character(),
##   ..   THP = col_double(),
```

```
##   ..   FTM_FTA = col_character(),
##   ..   FTP = col_double(),
##   ..   APG = col_double(),
##   ..   BLKPG = col_double(),
##   ..   STLPG = col_double(),
##   ..   TOPG = col_double(),
##   ..   PPG = col_double()
##   .. )
```

We can check if any of the columns has any missing data or NA values with the following code.

```
unique_elements = lapply(df,unique)
lapply(lapply(unique_elements,is.na),sum)
```

```
## $TEAM
## [1] 0
##
## $NAME
## [1] 0
##
## $EXPERIENCE
## [1] 0
##
## $URL
## [1] 0
##
## $POSITION
## [1] 0
##
## $AGE
## [1] 0
##
## $HT
## [1] 0
##
## $WT
## [1] 0
##
## $COLLEGE
## [1] 0
##
## $SALARY
## [1] 0
##
## $PPG_LAST_SEASON
## [1] 1
##
## $APG_LAST_SEASON
## [1] 1
##
## $RPG_LAST_SEASON
## [1] 1
##
## $PER_LAST_SEASON
## [1] 1
##
## $PPG_CAREER
## [1] 0
##
## $APG_CAREER
## [1] 0
##
## $RGP_CAREER
## [1] 0
##
## $GP
## [1] 0
##
## $MPG
## [1] 0
##
## $FGM_FGA
## [1] 0
```

```
##
## $FGP
## [1] 0
##
## $THM_THA
## [1] 0
##
## $THP
## [1] 0
##
## $FTM_FTA
## [1] 0
##
## $FTP
## [1] 0
##
## $APG
## [1] 0
##
## $BLKPG
## [1] 0
##
## $STLPG
## [1] 0
##
## $TOPG
## [1] 0
##
## $PPG
## [1] 0
```

Now we see that 4 columns have missing data.These 4 columns are actually those containing data about last years statistics .Since in our analysis we are not making any comparisons based on time series we can drop these columns.

```
df$PPG_LAST_SEASON = NULL
df$APG_LAST_SEASON = NULL
df$RPG_LAST_SEASON = NULL
df$PER_LAST_SEASON = NULL
```

Now we check if the age column has any non numeric data and replace it with the mean player age

```
unique(df$AGE)
```

```
##  [1] "31" "22" "24" "21" "28" "32" "26" "23" "29" "20" "25" "33" "27" "19"
## [15] "-"  "30" "34" "18" "36" "37" "35" "40" "38" "41"
```

```
df$AGE[df$AGE == '-'] = 0
df$AGE = sapply(df$AGE,as.numeric)
mean_age = mean(df$AGE)
df$AGE[df$AGE == 0] = round(mean_age)
```

Now we also see that the Salary column has a value which says “not signed”.This means that the particular player does not have a contract yet ,hence we replace his salary with 0.

```
unique(df$SALARY)
```

```
##   [1] "5,193,600"  "Not signed" "1,349,464"  "5,169,960"  "31,214,295"
##   [6] "28,928,709" "20,099,189" "5,375,000"  "1,378,242"  "3,050,390"
##  [11] "11,660,716" "6,700,800"  "838,464"    "2,667,600"  "2,034,120"
##  [16] "15,400,000" "18,500,000" "4,449,000"  "1,656,092"  "9,530,000"
##  [21] "13,764,045" "1,512,601"  "8,000,000"  "2,470,357"  "1,618,320"
##  [26] "1,702,800"  "1,632,240"  "1,942,422"  "7,019,698"  "5,000,000"
##  [31] "4,544,000"  "1,795,015"  "17,325,000" "6,500,000"  "18,622,514"
##  [36] "3,739,920"  "1,619,260"  "12,253,780" "4,294,480"  "4,155,720"
##  [41] "5,697,054"  "1,485,440"  "7,119,650"  "8,575,916"  "12,800,562"
##  [46] "10,464,092" "25,467,250" "8,339,880"  "1,740,000"  "1,600,520"
##  [51] "12,250,000" "2,526,840"  "1,703,649"  "6,434,520"  "10,000,000"
##  [56] "21,666,667" "23,114,067" "31,200,000" "8,333,333"  "1,826,300"
##  [61] "9,367,200"  "1,569,360"  "1,544,951"  "16,539,326" "8,653,847"
##  [66] "2,536,898"  "5,337,000"  "37,457,154" "30,000,000" "1,644,240"
##  [71] "17,469,565" "16,000,000" "8,307,692"  "18,988,725" "5,027,028"
##  [76] "12,000,000" "21,587,579" "3,375,360"  "13,565,218" "6,000,000"
##  [81] "14,800,000" "6,134,520"  "7,000,000"  "4,320,500"  "3,046,200"
##  [86] "6,300,000"  "1,349,383"  "7,461,960"  "3,500,000"  "1,000,000"
##  [91] "1,655,160"  "5,757,120"  "35,654,150" "1,689,840"  "1,487,694"
##  [96] "9,000,000"  "1,762,080"  "20,421,546" "15,000,000" "7,464,912"
## [101] "8,165,160"  "4,661,280"  "3,314,365"  "3,552,960"  "13,585,000"
## [106] "3,258,539"  "6,041,520"  "949,000"    "1,238,464"  "11,750,000"
## [111] "7,305,600"  "4,696,875"  "3,000,000"  "5,470,920"  "2,207,040"
## [116] "3,844,760"  "2,807,880"  "8,739,500"  "5,460,000"  "11,692,308"
## [121] "11,011,234" "11,286,516" "4,441,200"  "4,221,000"  "8,740,980"
## [126] "4,384,616"  "1,990,520"  "19,500,000" "14,357,750" "4,536,120"
## [131] "20,000,000" "3,263,294"  "2,494,346"  "2,280,600"  "12,500,000"
## [136] "2,760,095"  "19,000,000" "3,472,887"  "7,560,000"  "24,119,025"
## [141] "2,272,391"  "2,775,000"  "4,068,600"  "14,720,000" "1,952,760"
## [146] "2,500,000"  "25,434,263" "1,857,480"  "32,088,932" "17,043,478"
## [151] "3,940,402"  "3,275,280"  "10,002,681" "4,075,000"  "10,500,000"
## [156] "12,400,000" "1,911,960"  "7,945,000"  "2,407,560"  "7,333,333"
## [161] "21,000,000" "2,659,800"  "3,410,284"  "13,964,045" "24,157,303"
## [166] "1,641,000"  "9,607,500"  "2,481,000"  "11,327,466" "3,382,000"
## [171] "2,799,720"  "13,000,000" "10,607,143" "2,534,280"  "3,710,850"
## [176] "24,107,258" "1,230,000"  "6,560,640"  "22,897,200" "9,631,250"
## [181] "3,819,960"  "15,293,104" "3,206,160"  "1,621,415"  "13,500,375"
## [186] "35,650,150" "3,651,480"  "14,631,250" "7,969,537"  "8,641,000"
## [191] "30,521,115" "7,666,667"  "5,915,040"  "5,285,394"  "12,252,928"
## [196] "25,976,111" "2,205,000"  "8,808,685"  "1,567,707"  "22,347,015"
## [201] "6,153,846"  "2,487,000"  "27,739,975" "3,125,000"  "16,800,000"
## [206] "10,087,200" "11,571,429" "2,947,320"  "2,357,160"  "1,667,160"
## [211] "2,516,048"  "18,089,887" "1,634,640"  "2,299,080"  "7,200,000"
## [216] "2,250,960"  "4,350,000"  "13,766,421" "1,620,480"  "5,356,440"
## [221] "24,000,000" "17,000,000" "3,206,640"  "988,464"    "3,627,842"
## [226] "7,488,372"  "3,447,480"  "14,087,500" "13,528,090" "2,955,840"
## [231] "18,109,175" "6,270,000"  "14,651,700" "19,245,370" "12,537,527"
## [236] "11,550,000" "25,434,262" "3,448,926"  "7,250,000"  "4,865,040"
## [241] "1,050,000"  "21,590,909" "2,639,314"  "4,969,080"  "2,416,222"
## [246] "12,750,000" "2,749,080"  "15,944,154" "3,454,500"  "8,600,000"
## [251] "3,208,630"  "26,011,913" "12,650,000" "3,129,187"  "5,450,000"
## [256] "19,169,800" "11,830,358" "1,773,840"  "2,000,000"  "16,517,857"
## [261] "2,166,360"  "24,605,181" "1,874,640"  "3,364,249"  "29,230,769"
## [266] "3,499,800"  "12,917,808" "2,894,160"  "20,445,779" "15,170,787"
## [271] "14,000,000" "2,444,053"  "2,160,746"  "4,750,000"  "7,839,435"
## [276] "5,455,236"  "2,118,840"  "30,560,700" "1,757,429"  "5,451,600"
## [281] "15,500,000" "6,957,105"  "3,628,920"  "2,795,000"  "10,837,079"
## [286] "10,595,506" "27,977,689" "25,759,766" "11,111,111" "1,760,520"
## [291] "17,868,853" "2,074,320"  "1,679,520"  "11,536,515" "7,305,825"
```

```
## [296] "9,600,000"  "16,900,000" "23,241,573" "13,045,455" "3,111,480"
## [301] "2,150,000"  "14,975,000" "5,250,000"  "3,360,000"
```

```
df$SALARY[df$SALARY == "Not signed"] = "0"
df$SALARY = as.numeric(gsub(",", "", df$SALARY))
df$COLLEGE[grep("-",df$COLLEGE)] = "Others"
```

We will not be using the URL column as it has some external links.We can drop it.

```
df$URL = NULL
```

We see that the columns FGM_FGA(Field goals made vs Field goals attempted) has data as a string with yphens.We are interested to know the ration of these numbers in the column.This ratio is directly indicated in the FGP(Field goal percentage).Similarly THM_THA and FTM_FTA can be represented by THP and FTP.Now since we have columns with required ratios we can drop redundant columns.

```
df$FTM_FTA = NULL
df$FGM_FGA = NULL
df$THM_THA = NULL
```

Since the ppg ,apg are redundant with columns representing same statistics exist for career.

```
df$PPG = NULL
df$APG = NULL
```

Player with the maximum Salary(considering only the players who have revealed their salary to ESPN).

```
df$NAME[df$SALARY == max(df$SALARY)]
```

```
## [1] "Stephen Curry"
```

Calculating count of players based on the given grouping

Plot of distribution of experience in the league

```
library("plotly")
```

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

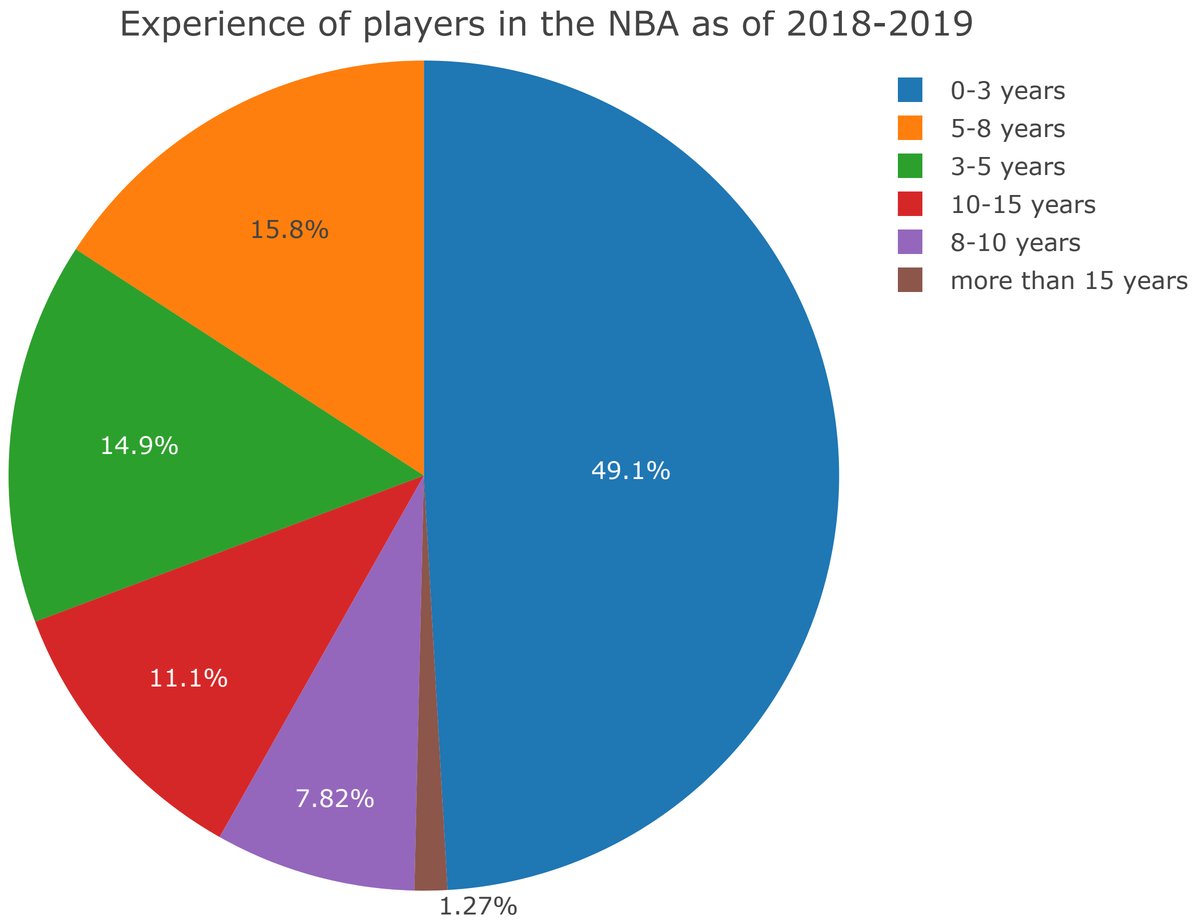
```
##
## Attaching package: 'plotly'
```

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

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot_ly(df[,3], labels = labels, values = ex, type = 'pie') %>%
        layout(title = 'Experience of players in the NBA as of 2018-2019',
        xaxis = list(showgrid = FALSE, zeroline = FALSE),
        yaxis = list(showgrid = FALSE, zeroline = FALSE))
```



From the above pie chart we can see that majority of the players are fairly young having very little experience playing in the league.We can also see that there are very few players who have been in the league for more than 15 years.

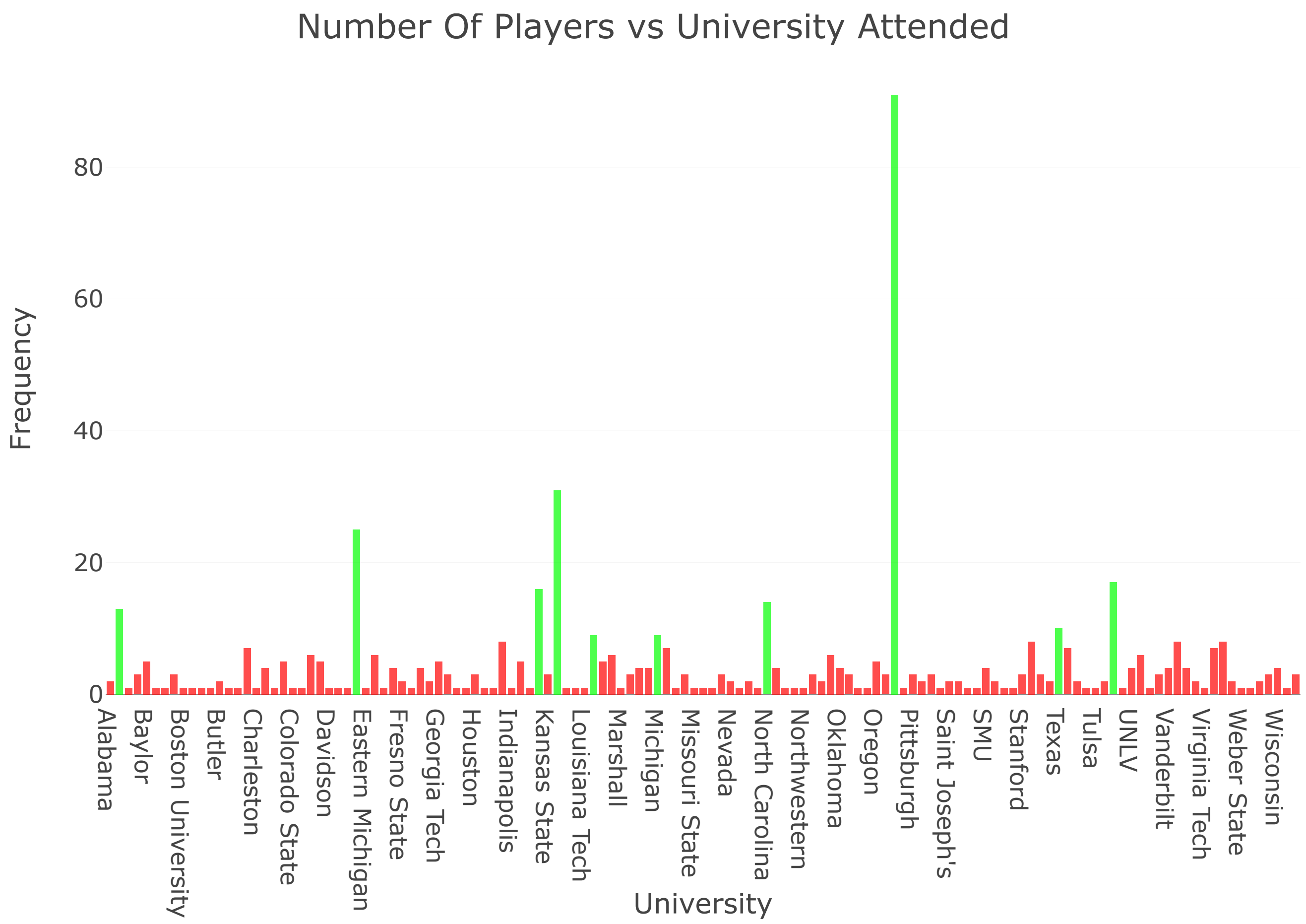
Calculating the number of players in the league based on college they attended.

```
c = sort(df$COLLEGE)
c = as.data.frame(table(c))
colnames(c) = c("College","Frequency")
#c = c[ -c(1),]
is_applicable = vector()
for (i in 1:length(c$Frequency)) {
  if ( c$Frequency[i] > 8)
    is_applicable[i] = TRUE
  else
    is_applicable[i] = FALSE
}
colors = vector(mode = "character",length = 30)
for (i in 1:length(c$Frequency)) {
  if ( is_applicable[i])
    colors[i] = "rgba(0,255,0,0.7)"
  else
    colors[i] = "rgba(255,0,0,0.7)"
}
```

Plot of number of players vs university attended

```
p = plot_ly(x = ~c$College,y = c$Frequency,marker = list(color = colors))
p = layout(p,title = "Number Of Players vs University Attended",xaxis = list(title = "Universi
ty",type = "category"),yaxis = list(title = "Frequency"))
p
```

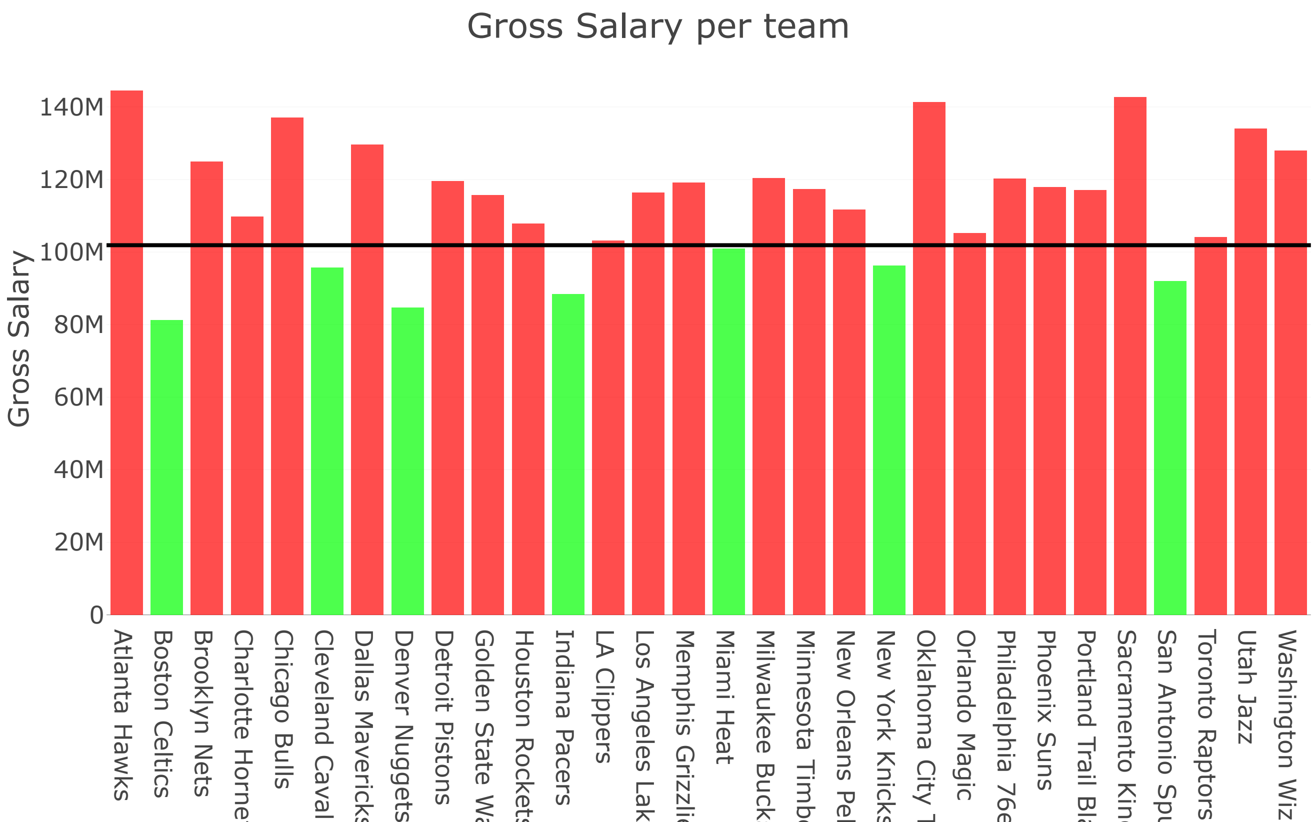
```
## No trace type specified:
##   Based on info supplied, a 'bar' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#bar
```



From the above plot,we can see that most of the players in the may not have attended college in the USA.The University of Kentucky has the most NBA players among the universities situated in the USA.

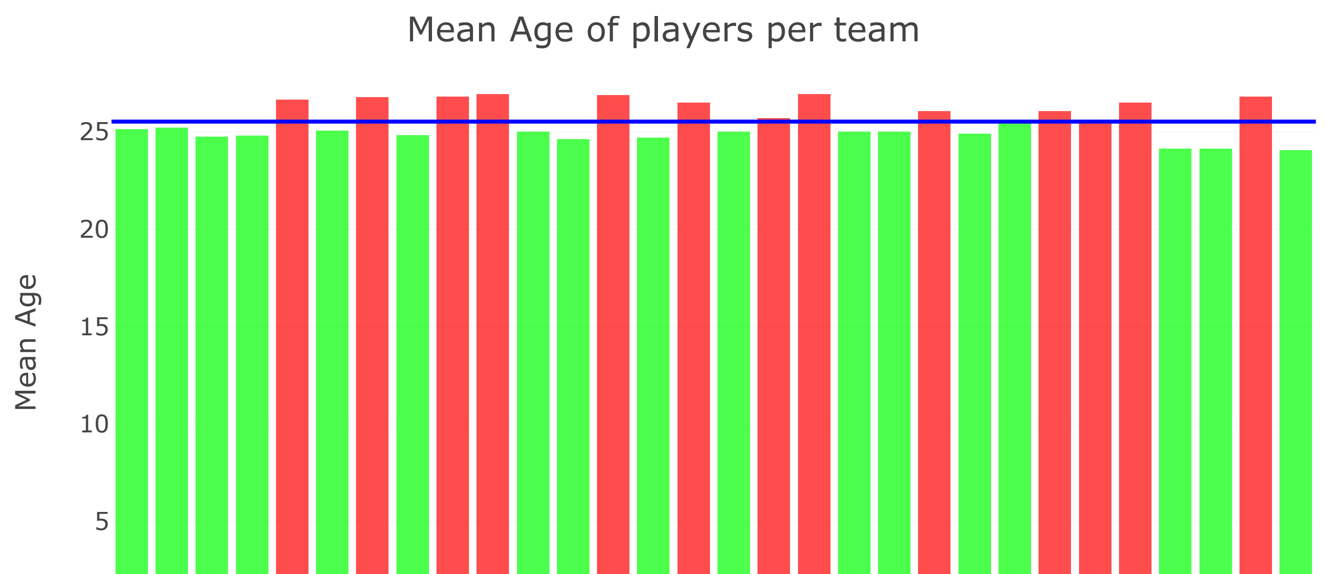
Calculating and plotting gross salary per team in the NBA for the 2018-19 season

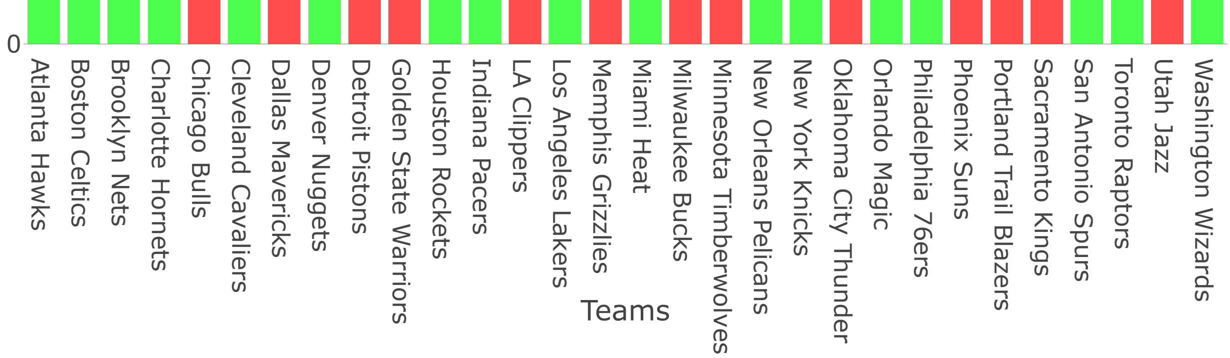
```r
team_gross_salary = tapply(df$SALARY, df$TEAM, sum)
teams = unique(df$TEAM)
sal_cap = 101900000
sals = data.frame(teams,team_gross_salary)
colnames(sals) = c("Team","Gross Salary")
is_applicable = vector()
for (i in 1:length(team_gross_salary)) {
  if ( team_gross_salary[i] > sal_cap)
    is_applicable[i] = TRUE
  else
    is_applicable[i] = FALSE
}
colors = vector(mode = "character",length = 30)
for (i in 1:length(team_gross_salary)) {
  if ( is_applicable[i])
    colors[i] = "rgba(255,0,0,0.7)"
  else
    colors[i] = "rgba(0,255,0,0.7)"
}
hline <- function(y,color = "black") {
  list(
    type = "line",
    name = "NBA 2018-19 Salary Cap",
    x0 = 0,
    x1 = 1,
    xref = "paper",
    y0 = y,
    y1 = y,
    line = list(color = color)
  )
}
p <- plot_ly(sals,x = ~teams,y = ~team_gross_salary,type = "bar" ,marker = list(color = color
s))
p = layout(p,title = "Gross Salary per team",xaxis = list(title = "Teams"),yaxis = list(title
= "Gross Salary"))
p <- layout(p,shapes = list(hline(sal_cap)))
p
```



From the above plot,we can see that majority of the teams have exceeded the NBA salary cap and do not have much cap space to sign new players.

```
nba_mean_age = mean(df$AGE)
team_mean_age = tapply(df$AGE, df$TEAM, mean)
ages = data.frame(teams,team_mean_age)
colnames(sals) = c("Team","Team Mean Age")
is_applicable = vector()
for (i in 1:length(team_mean_age)) {
  if ( team_mean_age[i] > nba_mean_age)
    is_applicable[i] = TRUE
  else
    is_applicable[i] = FALSE
}
colors = vector(mode = "character",length = 30)
for (i in 1:length(team_mean_age)) {
  if ( is_applicable[i])
    colors[i] = "rgba(255,0,0,0.7)"
  else
    colors[i] = "rgba(0,255,0,0.7)"
}
hline <- function(y,color = "blue") {
  list(
    type = "line",
    title = "NBA 2018-19 Mean Age",
    name = "NBA 2018-19 Mean Age",
    x0 = 0,
    x1 = 1,
    xref = "paper",
    y0 = y,
    y1 = y,
    line = list(color = color)
  )
}
p <- plot_ly(ages,x = ~teams,y = ~team_mean_age,type = "bar" ,marker = list(color = colors))
p = layout(p,title = "Mean Age of players per team",xaxis = list(title = "Teams"),yaxis = list
(title = "Mean Age"))
p <- layout(p,shapes = list(hline(nba_mean_age)))
p
```





From the above plot,we can see that the mean player age of majority of the teams is less than the league's mean player age.

Calculating top earner from each team

```
#install.packages("dplyr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

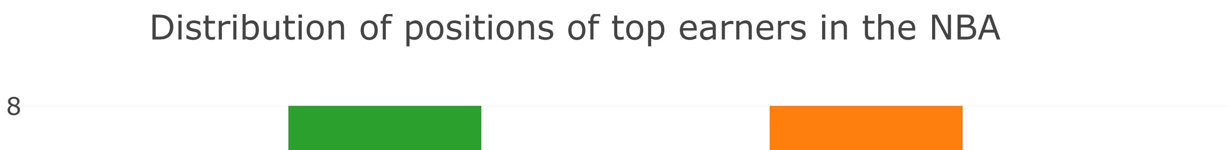
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

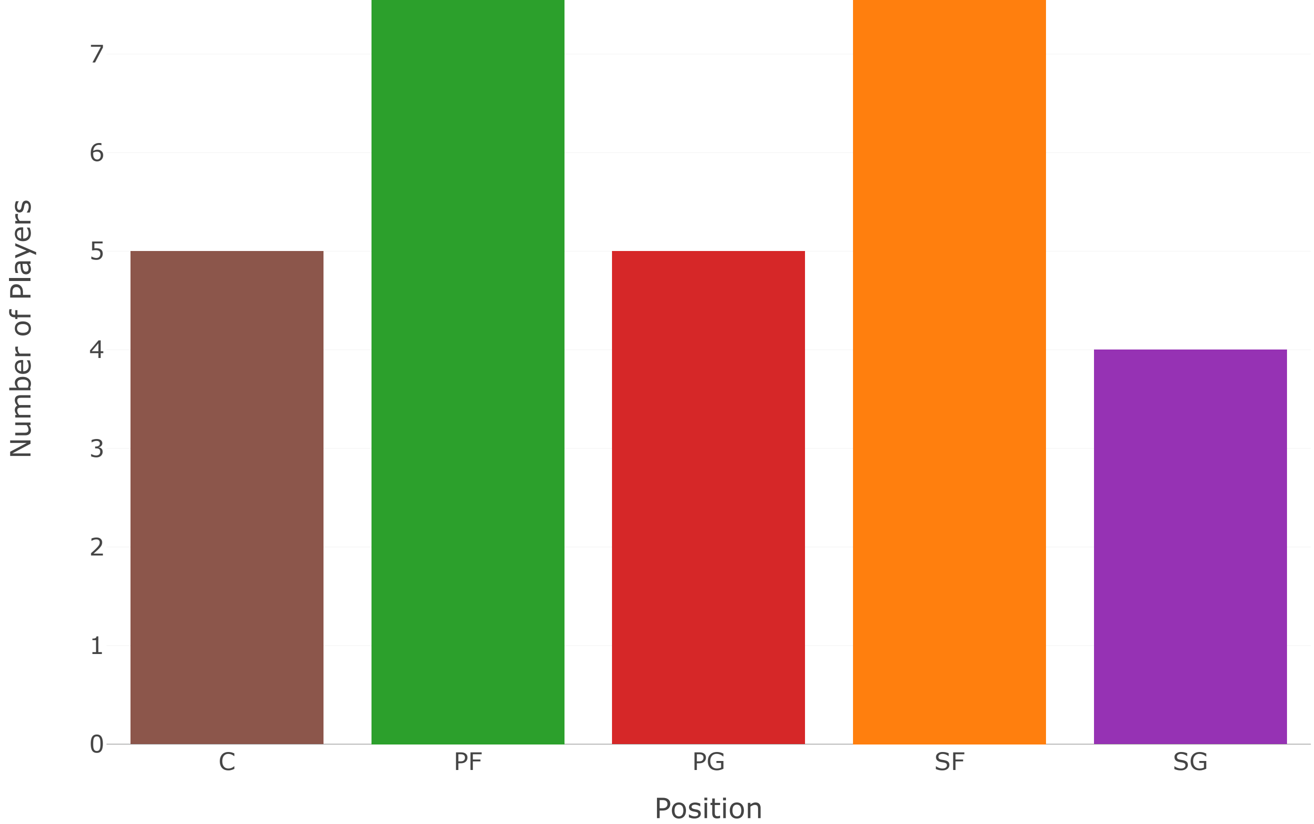
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
max_sal = function(group){
  group[which.max(group)]
}
top_earners = group_by(df)
top_earners = top_earners[order(top_earners$TEAM),]
sals = as.data.frame(tapply(top_earners$SALARY,top_earners$TEAM,max_sal))
colnames(sals) = c("MaxSalary")
top_earners = top_earners %>% filter(SALARY %in% sals$MaxSalary)
top_earners = top_earners[-c(8,17,18,24),]
pos = unique(top_earners$POSITION)
counts = count(top_earners,top_earners$POSITION)
colnames(counts) = c("Position","No of Players")
colors = c("rgb(150, 50, 180)","rgb(255, 127, 14)","rgb(44, 160, 44)","rgb(214, 39, 40)","rgb
(140, 86, 75)")
```

Plotting top earner vs position played

```
p = plot_ly(counts,x = ~pos,y = ~counts$`No of Players`,type = "bar",marker = list(color = col
ors ))
p = layout(p,title = "Distribution of positions of top earners in the NBA",xaxis = list(title
= "Position"),yaxis = list(title = "Number of Players"))
p
```





We can see that the league has quite a few high earning players who are Power Forwards and Small Forwards.

To check the correlation between the columns we have to drop the non numeric columns.

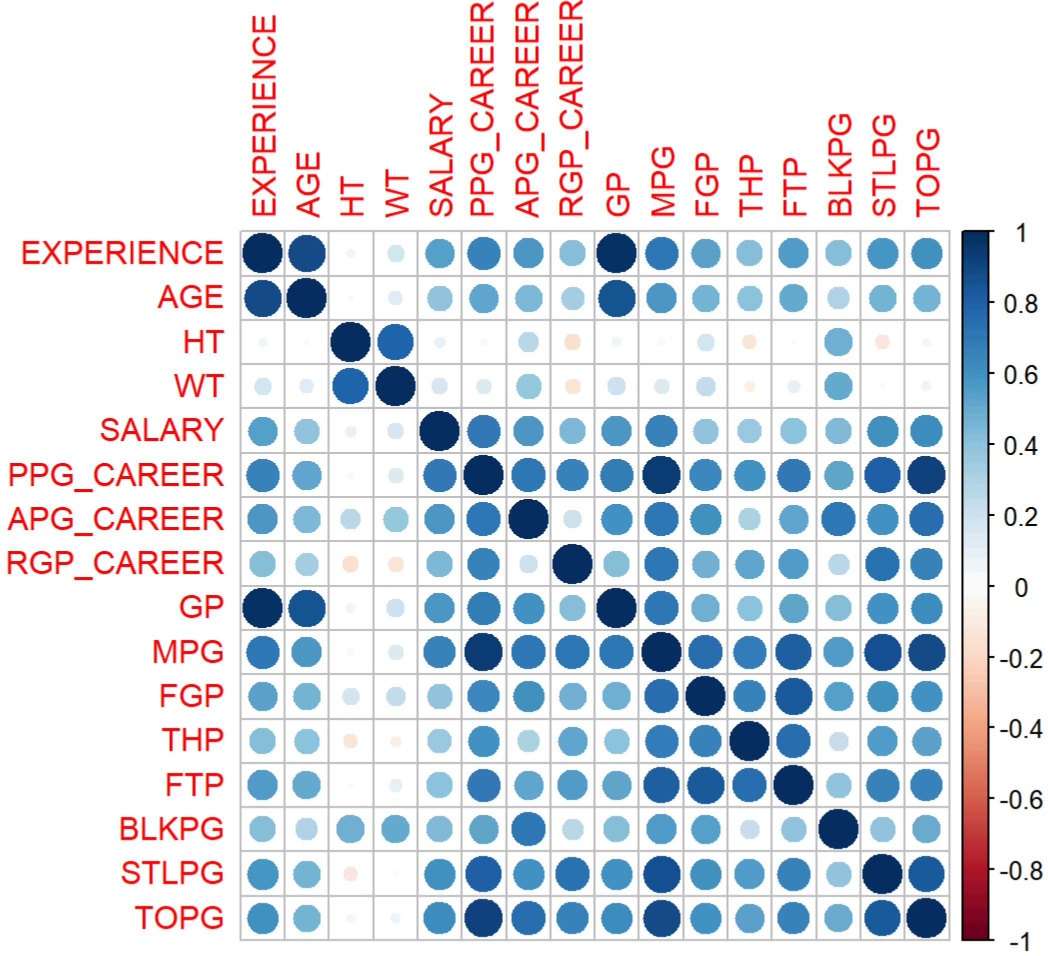
```
a = df
a$TEAM = NULL
a$NAME = NULL
a$COLLEGE = NULL
a$POSITION = NULL
```

Constructing a correlation plot and a correlation matrix to check the and visualize correlation.

```
#install.packages("corrplot")
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
Matrix = cor(a)
corrplot(Matrix,method = "circle")
```



To check How salary depends on other columns we check the columns having a correlation of more than 0.6.

```
Matrix[5,] > 0.6
```

```
## EXPERIENCE        AGE         HT         WT     SALARY PPG_CAREER
##      FALSE      FALSE      FALSE      FALSE       TRUE       TRUE
## APG_CAREER RGP_CAREER         GP        MPG        FGP        THP
##      FALSE      FALSE      FALSE       TRUE      FALSE      FALSE
##        FTP      BLKPG      STLPG       TOPG
##      FALSE      FALSE       TRUE       TRUE
```

Splitting the data into 70% training and 30% test data.

```
train <- df[1:440,]
test <- df[440:550,]
```

We see that the columns PPG_career , MPG, STLPG,TOPG AFFECT THE sALARY.

```
model <- lm(SALARY~ PPG_CAREER+MPG+STLPG+TOPG,data = train)
summary(model)
```

```
##
## Call:
## lm(formula = SALARY ~ PPG_CAREER + MPG + STLPG + TOPG, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -19950526  -2948146     76553   1853379  18140458
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -76554     461693  -0.166  0.86838
## PPG_CAREER   1133988     143784   7.887 2.53e-14 ***
## MPG           -92158      73342  -1.257  0.20959
## STLPG        3492079    1276520   2.736  0.00648 **
## TOPG        -2212234     869204  -2.545  0.01127 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5374000 on 435 degrees of freedom
## Multiple R-squared:  0.501,  Adjusted R-squared:  0.4964
## F-statistic: 109.2 on 4 and 435 DF,  p-value: < 2.2e-16
```

Having constructed a linear model with the following variables affecting the salary attribute, we see that the r-squared is not very high indicating the model is not the best we can arrive at(correlation does not mean or indicate causation). Howevever intuitively we see that the number of games played by a player has to affect the salary he receives.

```
model <- lm(SALARY~ PPG_CAREER+MPG+STLPG+TOPG+GP,data = train)
summary(model)
```

```
##
## Call:
## lm(formula = SALARY ~ PPG_CAREER + MPG + STLPG + TOPG + GP, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -21629597  -2574002    -88211   1707129  17525424
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    88211     455000   0.194  0.84637
## PPG_CAREER   1027170     143478   7.159 3.49e-12 ***
## MPG          -159617      73808  -2.163  0.03112 *
## STLPG        3601952    1253519   2.873  0.00426 **
## TOPG        -1833623     858191  -2.137  0.03319 *
## GP              5103       1226   4.161 3.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5276000 on 434 degrees of freedom
## Multiple R-squared:  0.5202, Adjusted R-squared:  0.5146
## F-statistic:  94.1 on 5 and 434 DF,  p-value: < 2.2e-16
```

We now see that adding the GP as one of the factors for the salary attribute,increases the r-squared indicating that the model is a better fit .Also we see that p-value indicated here is very very low.This means that the p-value is statistically significant at a confidence level of 99% also.This means we can reject the null hypothesis that the given attributes do not affect the salary of the player.Basically we can assume that there is a correlation between the salary and the above fields.

Calculating the correlation accuracy for the model

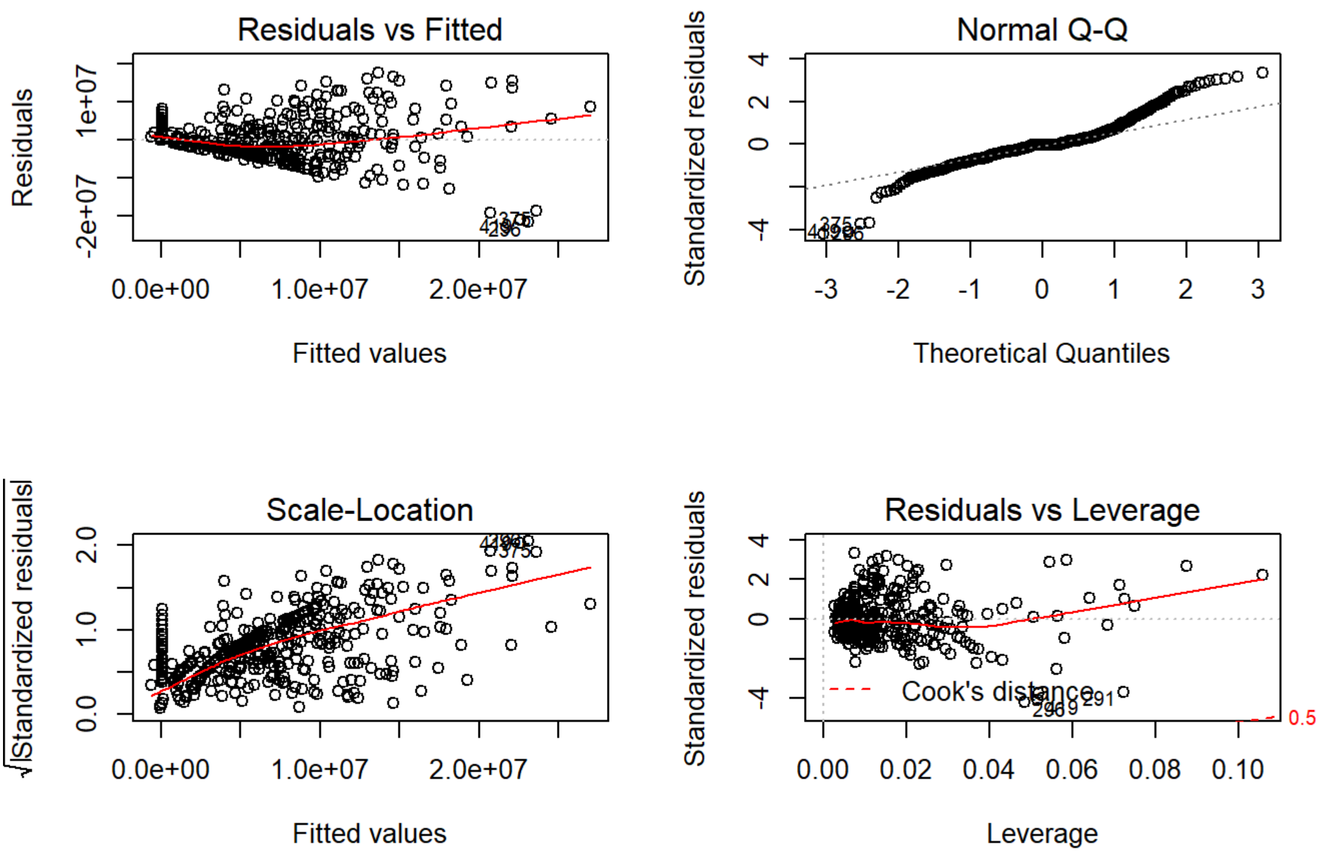
```
predicted1 <- predict(model,test)
act_pred1 <- data.frame(cbind(actuals = test$SALARY,predict = predicted1))
cor_acc <- cor(act_pred1)
print(paste0("Correlation accuracy=",cor_acc[1,2]))
```

```
## [1] "Correlation accuracy=0.706123516422658"
```

We see that the correlation accuracy is 70.6% which is is not very good but reasonable.

Plotting the residuals vs Fitted values and also the normal Q-Q plot to check the variance ,linear relationship and the normality of residuals.

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



We see that in the residuals plot the line at 0 is not linear exactly showing there does not completely exist a linear relationship for the linear regression model we have made.However considerinng most part of it as linear we observe heteroscedasticity as there is unequal variance on both sides of the line.The Q-Q plot actually shows a reasonable fit showing the residuals distribution to be almost normal. Hence we can conclude, that the model we have coctructed is not a very good estimator of the players' salary as a linear model is not sufficient in this case.

Constructing another model to predict the games played by a player in his career based on his age and experience.

```
model2 <- lm(GP~EXPERIENCE+AGE,data = train)
summary(model)
```

```
##
## Call:
## lm(formula = SALARY ~ PPG_CAREER + MPG + STLPG + TOPG + GP, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -21629597  -2574002    -88211   1707129  17525424
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    88211     455000   0.194  0.84637
## PPG_CAREER   1027170     143478   7.159 3.49e-12 ***
## MPG          -159617      73808  -2.163  0.03112 *
## STLPG        3601952    1253519   2.873  0.00426 **
## TOPG        -1833623     858191  -2.137  0.03319 *
## GP              5103       1226   4.161 3.83e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5276000 on 434 degrees of freedom
## Multiple R-squared:  0.5202, Adjusted R-squared:  0.5146
## F-statistic:  94.1 on 5 and 434 DF,  p-value: < 2.2e-16
```

We now see that the r-squaredis very high (.971) indicating that the model is a very good fit .Also we see that p-value indicated here is very very low.This means that the p-value is statistically significant at a confidence level of 99% also.This means we can reject the null hypothesis that the given attributes do not affect the games played by the player.Basically we can assume that there is a strong correlation between the games played and the above fields.

Calculating the correlation accuracy for the model

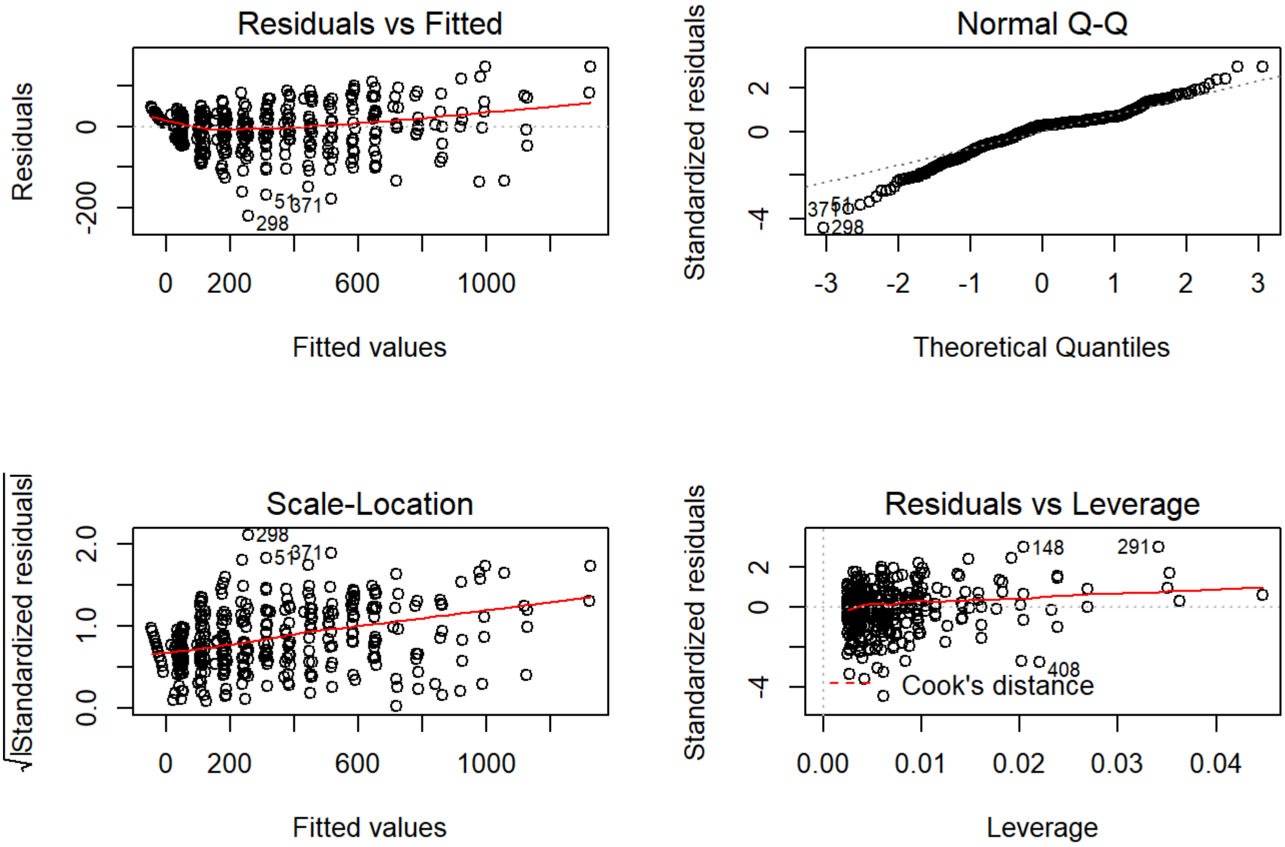
```
predicted2 <- predict(model2,test)
act_pred2 <- data.frame(cbind(actuals = test$GP,predict = predicted2))
cor_acc2 <- cor(act_pred2)
print(paste0("Correlation accuracy=",cor_acc2[1,2]))
```

```
## [1] "Correlation accuracy=0.980395501874178"
```

The correlation accuracy is approx 98% which indicates the model is a very good fit.

Plotting the residuals vs Fitted values and also the normal Q-Q plot to check the variance ,linear relationship and the normality of residuals for the second model.

```
par(mfrow = c(2, 2))
plot(model2)
```



Now we see that residuals vs fitted values is slightly better showing a homoscedastic relationship and the Q-Q plot shows almost a normal distribution.

In conclusion the second model constructed to predict the Games played is a better fit and a decent model with high accuracy.