DATA ANALYSIS AND REGRESSION

REPORT

Introduction:

This project involves building a predictive model to estimate NBA player salaries for the 2017-2018 season. The prediction is based on player performance data from the 2016-2017 season, allowing us to assess how past performance influences future earnings.

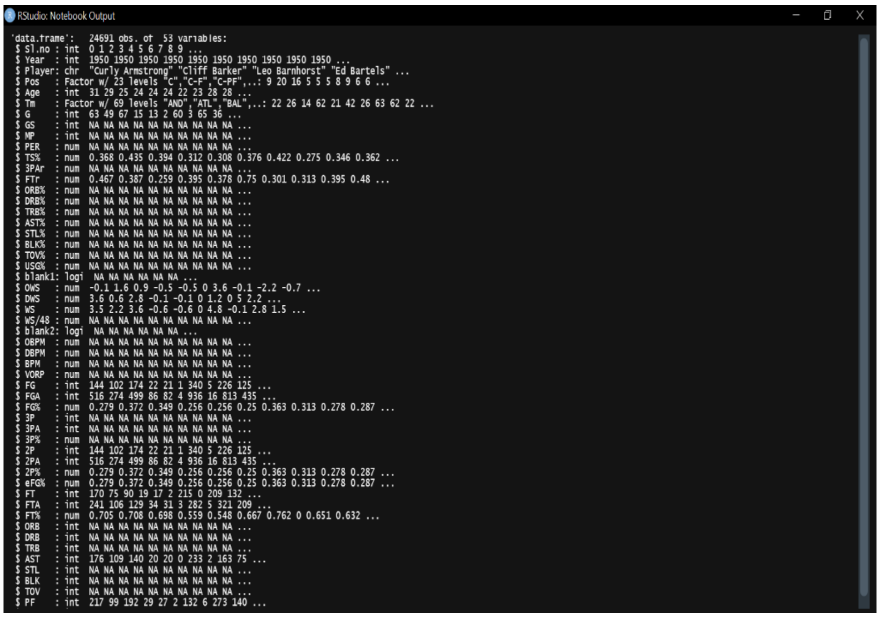
Objective: To develop a model that can predict the 2017-2018 season salaries of NBA players using their statistics from the previous season.

Understanding the relationship between player performance and salary can provide insights into the factors that NBA teams consider when compensating players. It highlights which performance metrics are most valued and how they contribute to a player's market worth, guiding both management decisions and player career strategies.

1.Data gathering and integration

Here I am using two datasets, one is the stats dataset and other is the Salary dataset.

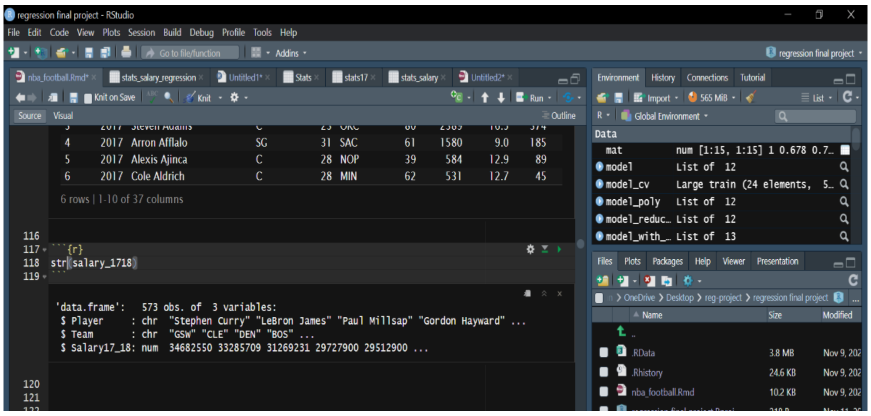
1.1 Stats:



```
'data.frame':	24691 obs. of  53 variables:
 $ Sl.no : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Year  : int  1950 1950 1950 1950 1950 1950 1950 1950 1950 1950 ...
 $ Player: chr  "Curly Armstrong" "Cliff Barker" "Leo Barnhorst" "Ed Bartels" ...
 $ Pos   : Factor w/ 23 levels "C","C-F","C-PF",..: 9 20 16 5 5 5 8 9 6 6 ...
 $ Age   : int  31 29 25 24 24 24 22 23 28 28 ...
 $ Tm    : Factor w/ 69 levels "AND","ATL","BAL",..: 22 26 14 62 21 42 26 63 62 22 ...
 $ G     : int  63 49 67 15 13 2 60 3 65 36 ...
 $ GS    : int  NA NA NA NA NA NA NA NA NA NA ...
 $ MP    : int  NA NA NA NA NA NA NA NA NA NA ...
 $ PER   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ TS%   : num  0.368 0.435 0.394 0.312 0.308 0.376 0.422 0.275 0.346 0.362 ...
 $ 3PAr  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ FTr   : num  0.467 0.387 0.259 0.395 0.378 0.75 0.301 0.313 0.395 0.48 ...
 $ ORB%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ DRB%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ TRB%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ AST%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ STL%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ BLK%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ TOV%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ USG%  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ blank1: logi  NA NA NA NA NA NA ...
 $ OWS   : num  -0.1 1.6 0.9 -0.5 -0.5 0 3.6 -0.1 -2.2 -0.7 ...
 $ DWS   : num  3.6 0.6 2.8 -0.1 -0.1 0 1.2 0 5 2.2 ...
 $ WS    : num  3.5 2.2 3.6 -0.6 -0.6 0 4.8 -0.1 2.8 1.5 ...
 $ WS/48 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ blank2: logi  NA NA NA NA NA NA ...
 $ OBPM  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ DBPM  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ BPM   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ VORP  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ FG    : int  144 102 174 22 21 1 340 5 226 125 ...
 $ FGA   : int  516 274 499 86 82 4 936 16 813 435 ...
 $ FG%   : num  0.279 0.372 0.349 0.256 0.256 0.25 0.363 0.313 0.278 0.287 ...
 $ 3P    : int  NA NA NA NA NA NA NA NA NA NA ...
 $ 3PA   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ 3P%   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ 2P    : int  144 102 174 22 21 1 340 5 226 125 ...
 $ 2PA   : int  516 274 499 86 82 4 936 16 813 435 ...
 $ 2P%   : num  0.279 0.372 0.349 0.256 0.256 0.25 0.363 0.313 0.278 0.287 ...
 $ eFG%  : num  0.279 0.372 0.349 0.256 0.256 0.25 0.363 0.313 0.278 0.287 ...
 $ FT    : int  170 75 90 19 17 2 215 0 209 132 ...
 $ FTA   : int  241 106 129 34 31 3 282 5 321 209 ...
 $ FT%   : num  0.705 0.708 0.698 0.559 0.548 0.667 0.762 0 0.651 0.632 ...
 $ ORB   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ DRB   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ TRB   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ AST   : int  176 109 140 20 20 0 233 2 163 75 ...
 $ STL   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ BLK   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ TOV   : int  NA NA NA NA NA NA NA NA NA NA ...
 $ PF    : int  217 99 192 29 27 2 132 6 273 140 ...
```

This dataset contains detailed performance metrics for NBA players from the year 1950 to 2017. It includes various statistics, such as points, assists, rebounds, and more, which are crucial for building features.

1.2 Salary_1718:



```
4   2017  Arron Afflalo   SG   31  SAC   61   1580    9.0   185
5   2017  Alexis Ajinca   C    28  NOP   39    584   12.9    89
6   2017  Cole Aldrich    C    28  MIN   62    531   12.7    45

6 rows | 1-10 of 37 columns
```

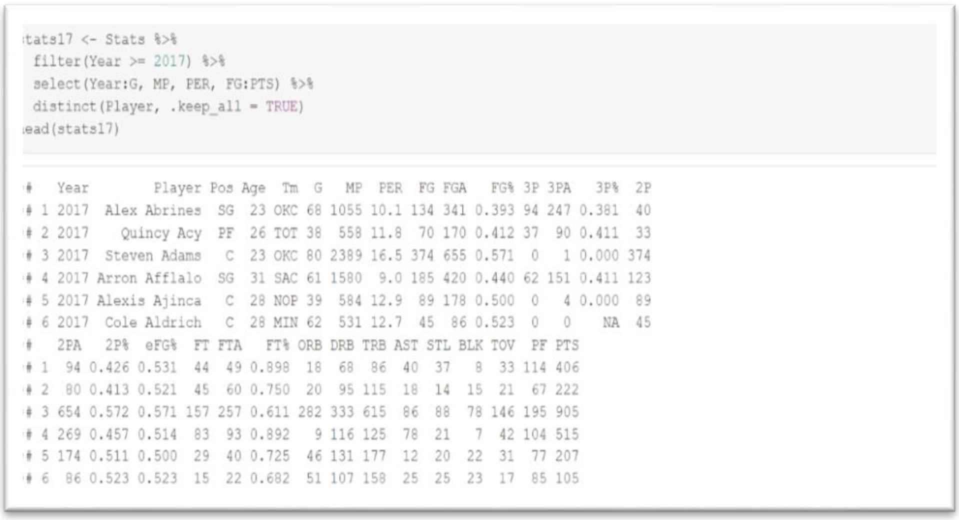
```{r}
str(salary_1718)
```

```
'data.frame':   573 obs. of  3 variables:
 $ Player     : chr  "Stephen Curry" "LeBron James" "Paul Millsap" "Gordon Hayward" ...
 $ Team       : chr  "GSW" "CLE" "DEN" "BOS" ...
 $ Salary17_18: num  34682550 33285709 31269231 29727900 29512900 ...
```

This includes the salaries of NBA players for the 2017-2018 season. This serves as the target variable for the regression model, providing the actual salaries that we aim to predict.

1.3 Filtering and Merging:

Since we are trying to predict the salaries of the players for the 17-18 Season on basis of their performance in 16-17 season, We do not require all data present in Stats dataset hence we will filter it out in accordance to our objective.



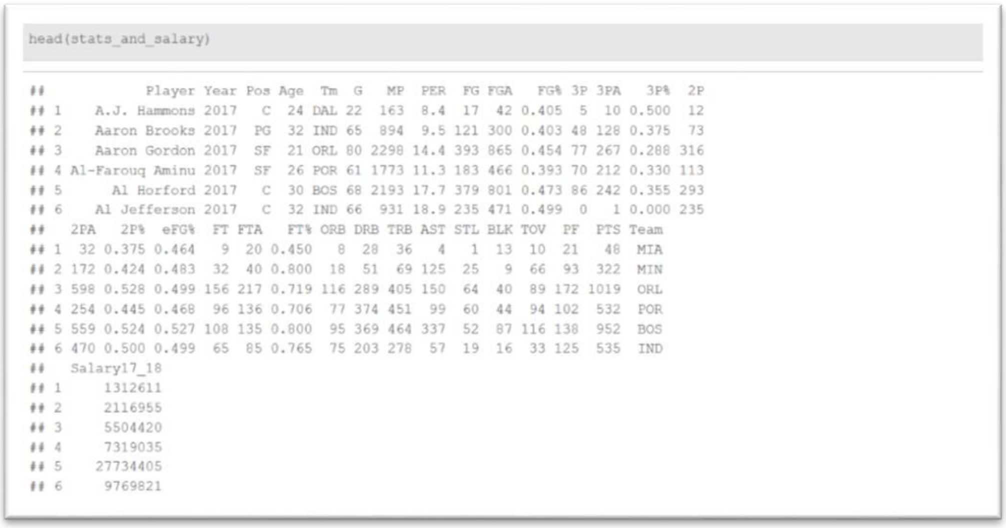
```
tats17 <- Stats %>%
  filter(Year >= 2017) %>%
  select(Year:G, MP, PER, FG:PTS) %>%
  distinct(Player, .keep_all = TRUE)
ead(stats17)
```

```
#   Year         Player Pos Age  Tm  G   MP  PER  FG FGA   FG% 3P 3PA   3P%  2P
# 1 2017   Alex Abrines  SG  23 OKC 68 1055 10.1 134 341 0.393 94 247 0.381  40
# 2 2017     Quincy Acy  PF  26 TOT 38  558 11.8  70 170 0.412 37  90 0.411  33
# 3 2017   Steven Adams   C  23 OKC 80 2389 16.5 374 655 0.571  0   1 0.000 374
# 4 2017 Arron Afflalo  SG  31 SAC 61 1580  9.0 185 420 0.440 62 151 0.411 123
# 5 2017 Alexis Ajinca   C  28 NOP 39  584 12.9  89 178 0.500  0   4 0.000  89
# 6 2017   Cole Aldrich   C  28 MIN 62  531 12.7  45  86 0.523  0   0    NA  45
#   2PA   2P%  eFG%  FT FTA   FT% ORB DRB TRB AST STL BLK TOV  PF PTS
# 1  94 0.426 0.531  44  49 0.898  18  68  86  40  37   8  33 114 406
# 2  80 0.413 0.521  45  60 0.750  20  95 115  18  14  15  21  67 222
# 3 654 0.572 0.571 157 257 0.611 282 333 615  86  88  78 146 195 905
# 4 269 0.457 0.514  83  93 0.892   9 116 125  78  21   7  42 104 515
# 5 174 0.511 0.500  29  40 0.725  46 131 177  12  20  22  31  77 207
# 6  86 0.523 0.523  15  22 0.682  51 107 158  25  25  23  17  85 105
```

Filtered data

The two datasets are then merged by matching rows with the same player names.

```r
stats_and_salary <- merge(stats17, salary_1718, by.x = "Player", by.y = "Player")
summary(stats_and_salary)
```



```
head(stats_and_salary)
```

```
##             Player Year Pos Age  Tm  G   MP  PER  FG FGA   FG% 3P 3PA   3P%  2P
## 1    A.J. Hammons 2017   C  24 DAL 22  163  8.4  17  42 0.405  5  10 0.500  12
## 2    Aaron Brooks 2017  PG  32 IND 65  894  9.5 121 300 0.403 48 128 0.375  73
## 3    Aaron Gordon 2017  SF  21 ORL 80 2298 14.4 393 865 0.454 77 267 0.288 316
## 4 Al-Farouq Aminu 2017  SF  26 POR 61 1773 11.3 183 466 0.393 70 212 0.330 113
## 5      Al Horford 2017   C  30 BOS 68 2193 17.7 379 801 0.473 86 242 0.355 293
## 6    Al Jefferson 2017   C  32 IND 66  931 18.9 235 471 0.499  0   1 0.000 235
##   2PA   2P%  eFG%  FT FTA   FT% ORB DRB TRB AST STL BLK TOV  PF  PTS Team
## 1  32 0.375 0.464   9  20 0.450   8  28  36   4   1  13  10  21   48  MIA
## 2 172 0.424 0.483  32  40 0.800  18  51  69 125  25   9  66  93  322  MIN
## 3 598 0.528 0.499 156 217 0.719 116 289 405 150  64  40  89 172 1019  ORL
## 4 254 0.445 0.468  96 136 0.706  77 374 451  99  60  44  94 102  532  POR
## 5 559 0.524 0.527 108 135 0.800  95 369 464 337  52  87 116 138  952  BOS
## 6 470 0.500 0.499  65  85 0.765  75 203 278  57  19  16  33 125  535  IND
##   Salary17_18
## 1     1312611
## 2     2116955
## 3     5504420
## 4     7319035
## 5    27734405
## 6     9769821
```

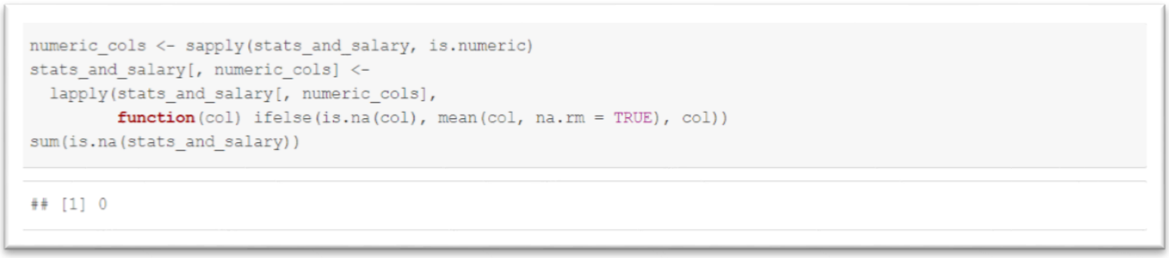
Merged data

2.Cleaning

Here we can remove the Team column from the stats_and_salary data as it is not required for analysis. This column indicates the team to which the player belonged in 17-18 season, Which is not required for us.

```r
library(dplyr)
stats_and_salary <- stats_and_salary %>% select(-Team)
```

Merged data has 32 Na values in total to be handled. We can replace these Na values by the means of their columns. I chose this approach because, since it is a sports data containing stats of each player, we cannot omit an entire player's data because of just few missing value



```r
numeric_cols <- sapply(stats_and_salary, is.numeric)
stats_and_salary[, numeric_cols] <-
  lapply(stats_and_salary[, numeric_cols],
         function(col) ifelse(is.na(col), mean(col, na.rm = TRUE), col))
sum(is.na(stats_and_salary))
```

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

3.Data Exploration

3.1 Per-Game statistics:

The original dataset includes whole-season statistics (e.g., total points, total assists) for each player. However, these cumulative totals can be misleading when comparing players who played different numbers of games.

Per-game statistics provide a more accurate representation of a player's average performance, making comparisons fairer.

Some players may not play the entire season due to injuries, resting, or other factors. As a result, their cumulative season totals could appear lower, even if they have strong performances when they are on the court.

By calculating per-game statistics (e.g., PPG - Points Per Game, MPG - Minutes Per Game), we normalize the data to reflect a player's contribution in each game they play, rather than across an entire season.

3.1 Creation of Per-Game Statistics

The dataset originally contained cumulative season totals (e.g., total points, total minutes played)

MPG (Minutes Per Game): MP/G

PPG (Points Per Game): PTS/G

APG (Assists Per Game): AST/G

RPG (Rebounds Per Game): TRB/G

TOPG (Turnovers Per Game): TOV/G

BPG (Blocks Per Game): BLK/G

SPG (Steals Per Game): STL/G

```r
stats_and_salary <- stats_and_salary %>%
  mutate(MPG = MP / G, PPG = PTS / G, APG = AST / G,
         RPG = TRB / G, TOPG = TOV / G, BPG = BLK / G,
         SPG = STL / G)
head(stats_and_salary)
```

```
##              Player Year Pos Age  Tm  G   MP  PER  FG FGA   FG% 3P 3PA   3P%  2P
## 1    A.J. Hammons 2017   C  24 DAL 22  163  8.4  17  42 0.405  5  10 0.500  12
## 2    Aaron Brooks 2017  PG  32 IND 65  894  9.5 121 300 0.403 48 128 0.375  73
## 3    Aaron Gordon 2017  SF  21 ORL 80 2298 14.4 393 865 0.454 77 267 0.288 316
## 4 Al-Farouq Aminu 2017  SF  26 POR 61 1773 11.3 183 466 0.393 70 212 0.330 113
## 5      Al Horford 2017   C  30 BOS 68 2193 17.7 379 801 0.473 86 242 0.355 293
## 6    Al Jefferson 2017   C  32 IND 66  931 18.9 235 471 0.499  0   1 0.000 235
##   2PA   2P%  eFG%  FT FTA   FT% ORB DRB TRB AST STL BLK TOV  PF  PTS
## 1  32 0.375 0.464   9  20 0.450   8  28  36   4   1  13  10  21   48
## 2 172 0.424 0.483  32  40 0.800  18  51  69 125  25   9  66  93  322
## 3 598 0.528 0.499 156 217 0.719 116 289 405 150  64  40  89 172 1019
## 4 254 0.445 0.468  96 136 0.706  77 374 451  99  60  44  94 102  532
## 5 559 0.524 0.527 108 135 0.800  95 369 464 337  52  87 116 138  952
## 6 470 0.500 0.499  65  85 0.765  75 203 278  57  19  16  33 125  535
##   Salary17_18       MPG       PPG       APG      RPG      TOPG       BPG
## 1     1312611  7.409091  2.181818 0.1818182 1.636364 0.4545455 0.5909091
## 2     2116955 13.753846  4.953846 1.9230769 1.061538 1.0153846 0.1384615
## 3     5504420 28.725000 12.737500 1.8750000 5.062500 1.1125000 0.5000000
## 4     7319035 29.065574  8.721311 1.6229508 7.393443 1.5409836 0.7213115
## 5    27734405 32.250000 14.000000 4.9558824 6.823529 1.7058824 1.2794118
## 6     9769821 14.106061  8.106061 0.8636364 4.212121 0.5000000 0.2424242
##          SPG
## 1 0.04545455
## 2 0.38461538
## 3 0.80000000
## 4 0.98360656
```

Now we can only keep our per-game stats for analysis by removing the cumulative stats of the player for entire season.

```
per_game_stats <- stats_and_salary %>% select(Player, Tm, MPG, PPG, APG, RPG, TOPG, BPG, SPG,  Salary17_18)
head(per_game_stats)
```

```
##            Player  Tm       MPG       PPG       APG      RPG      TOPG
## 1    A.J. Hammons DAL  7.409091  2.181818 0.1818182 1.636364 0.4545455
## 2    Aaron Brooks IND 13.753846  4.953846 1.9230769 1.061538 1.0153846
## 3    Aaron Gordon ORL 28.725000 12.737500 1.8750000 5.062500 1.1125000
## 4 Al-Farouq Aminu POR 29.065574  8.721311 1.6229508 7.393443 1.5409836
## 5      Al Horford BOS 32.250000 14.000000 4.9558824 6.823529 1.7058824
## 6    Al Jefferson IND 14.106061  8.106061 0.8636364 4.212121 0.5000000
##         BPG        SPG Salary17_18
## 1 0.5909091 0.04545455     1312611
## 2 0.1384615 0.38461538     2116955
## 3 0.5000000 0.80000000     5504420
## 4 0.7213115 0.98360656     7319035
## 5 1.2794118 0.76470588    27734405
## 6 0.2424242 0.28787879     9769821
```

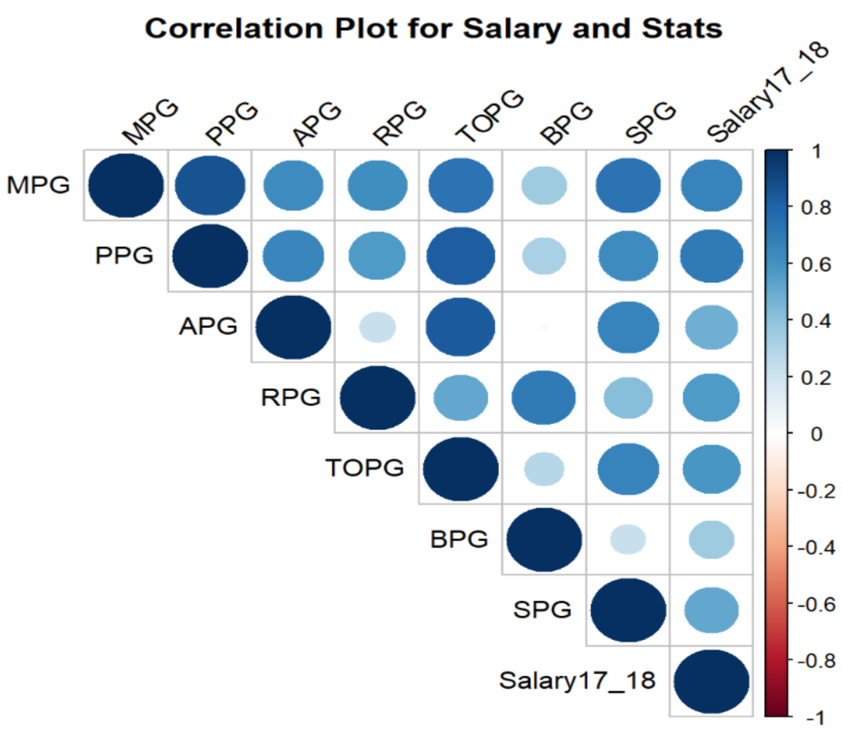
3.2 Correlation-1

```
library(dplyr)

numeric_columns <- per_game_stats %>%
  select(MPG, PPG, APG, RPG, TOPG, BPG, SPG, Salary17_18)

cor_matrix <- cor(numeric_columns, use = "complete.obs")

corrplot(cor_matrix,
         method = "circle",
         type = "upper",
         tl.col = "black",
         tl.srt = 45,
         title = "Correlation Plot for Salary and Stats",
         mar = c(0, 0, 1, 0))
```



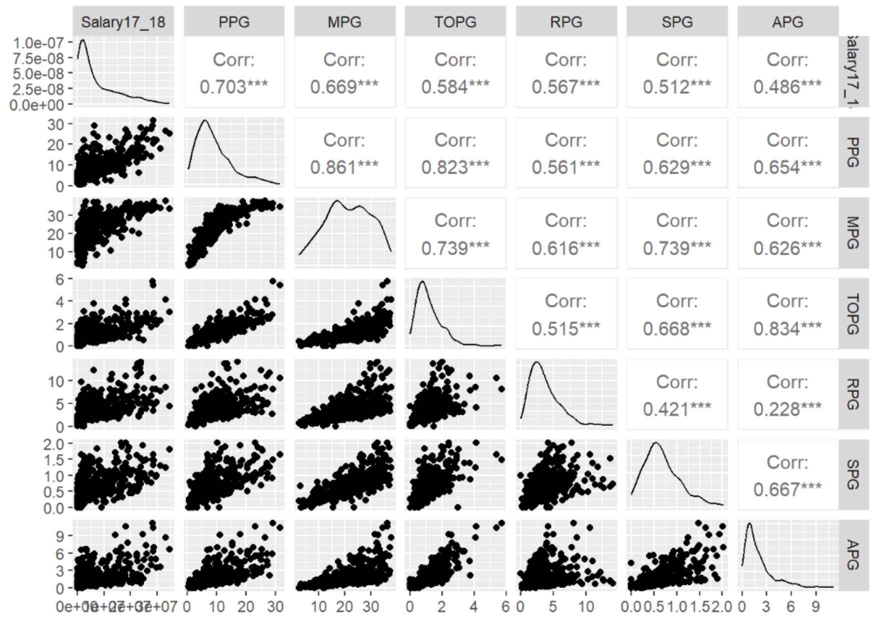
3.3 Correlation-2

```
library(dplyr)
library(GGally)
```

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

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
stats_salary_cor <-
  per_game_stats %>%
  select(Salary17_18, PPG, MPG, TOPG, RPG, SPG, APG)
ggpairs(stats_salary_cor)
```



By observing these two plots:

The Strength of correlation between the variables is ranked as follows: PPG>MPG>TOPG>RPG>PER>SPG>APG

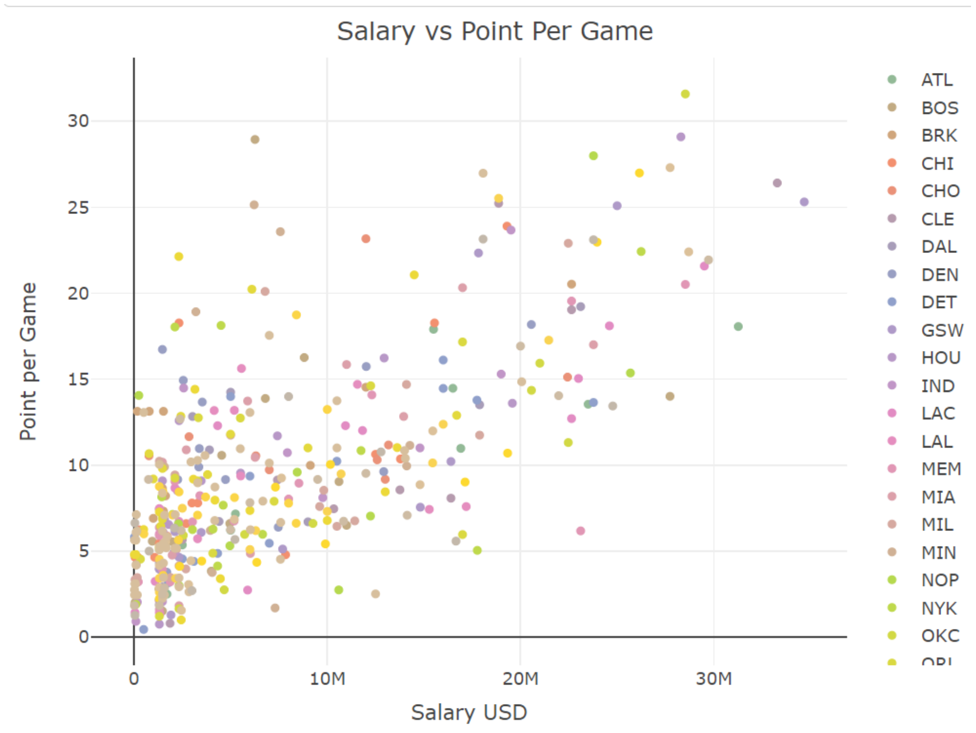
The number of turnovers (TOPG) made by players shows a positive correlation with their salary. We interpreted this as: “The more turnovers a player makes, the more involved they are in ball movements during the game. “

This could suggest that players who make turnovers might be more important to their team, possibly due to their aggressive playstyle.

We realize that this interpretation may not be fully accurate. For more precise analysis, we should consider additional data, such as how long each player holds the ball during games.

Let’s visualize a scatter plot between PPG and salary as they are highly correlated.

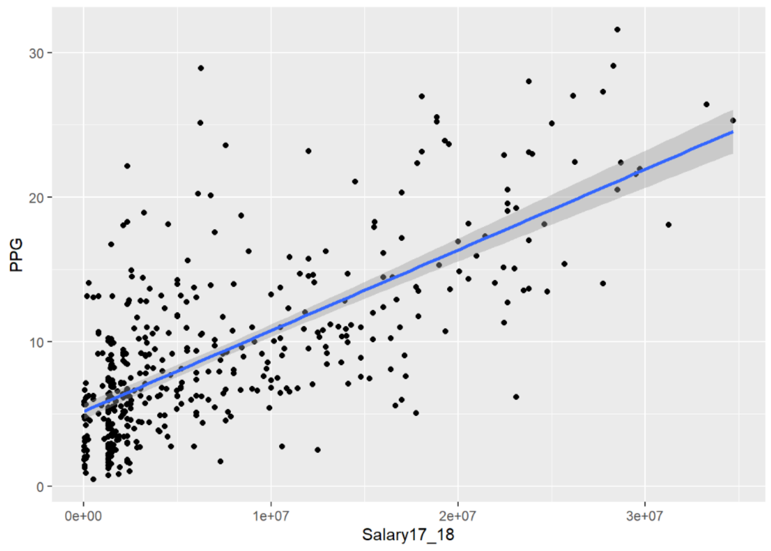
```
plot_ly(data = per_game_stats, x = ~Salary17_18, y = ~PPG, color = ~Team,
        hoverinfo = "text",
        text = ~paste("Player: ", Player,
                      "<br>Salary: ", format(Salary17_18, big.mark = ","),"$",
                      "<br>PPG: ", round(PPG, digits = 3),
                      "<br>Team: ", Team)) %>%
  layout(
    title = "Salary vs Point Per Game",
    xaxis = list(title = "Salary USD"),
    yaxis = list(title = "Point per Game")
  )
```



Point per Game VS Salary

If you hover on the points in this plot, you can check the player names. As you can tell the most paid player is Stephen Curry.

3.4 Linear regression:



You can see a general upward trend, which means that as players score more points per game, their salaries tend to be higher.

The blue line is a trend line that shows this overall pattern, and the shaded area around it represents the range where most data points are likely to fall, showing confidence in this trend.

In simple terms, players who score more points usually earn higher salaries.

```
stats_salary_regression <-
  per_game_stats %>% select(Salary17_18, MPG:SPG)
lm(Salary17_18~., data=stats_salary_regression)
```

```
## 
## Call:
## lm(formula = Salary17_18 ~ ., data = stats_salary_regression)
## 
## Coefficients:
## (Intercept)          MPG          PPG          APG          RPG         TOPG  
##    -2792909        30565       686815      1059087       916087     -2709447  
##         BPG          SPG  
##      470136       631255  
```

Scoring points, assisting, and rebounding contribute the most to higher salaries. Turnovers (losing possession) have a large negative impact on salary.

Defensive stats like blocks and steals also contribute positively but to a lesser extent than scoring and playmaking skills.

4.Data Pre-processing

Earlier in Correlation analysis we saw that even turnovers have a bit of positive impact over a player’s salary and we related it to a player’s aggressive playstyle, if a player is involved more in ball movements, he tends to commit turn overs, so here we can a create a variable called as Aggressiveness.

If a player’s turnovers per game are equal to or above this average, they’re labelled as “Yes” for Aggressiveness (meaning they turn over the ball relatively often).

Apart from this we can also observe that MPG (minutes per game) also have a positive relation with the salary. Players who generally trusted by their coaches tend to play more on court and tend to score more which boosts their salaries, hence we can create another variable called Trusted.

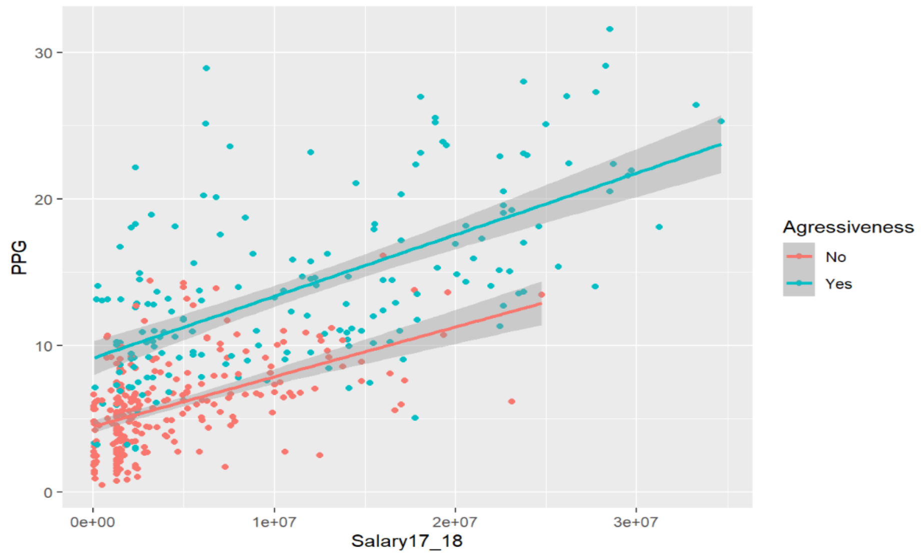
If a player’s minutes per game are equal to or above this average, they’re labelled as “Yes” for Trust (meaning the coach plays them often, indicating trust).

```
avg.minutes <- mean(per_game_stats$MPG)
avg.turnover <- mean(per_game_stats$TOPG)
per_game_stats$Trusted <- as.factor(ifelse(per_game_stats$MPG >= avg.minutes, "Yes", "No"))
per_game_stats$Agressiveness <- as.factor(ifelse(per_game_stats$TOPG >= avg.turnover, "Yes", "No"))
head(per_game_stats)
```

```
##            Player Team       MPG       PPG       APG      RPG      TOPG
## 1    A.J. Hammons  DAL  7.409091  2.181818 0.1818182 1.636364 0.4545455
## 2    Aaron Brooks  IND 13.753846  4.953846 1.9230769 1.061538 1.0153846
## 3    Aaron Gordon  ORL 28.725000 12.737500 1.8750000 5.062500 1.1125000
## 4 Al-Farouq Aminu  POR 29.065574  8.721311 1.6229508 7.393443 1.5409836
## 5      Al Horford  BOS 32.250000 14.000000 4.9558824 6.823529 1.7058824
## 6    Al Jefferson  IND 14.106061  8.106061 0.8636364 4.212121 0.5000000
##         BPG        SPG Salary17_18 Trusted Agressiveness
## 1 0.5909091 0.04545455     1312611      No            No
## 2 0.1384615 0.38461538     2116955      No            No
## 3 0.5000000 0.80000000     5504420     Yes            No
## 4 0.7213115 0.98360656     7319035     Yes           Yes
## 5 1.2794118 0.76470588    27734405     Yes           Yes
## 6 0.2424242 0.28787879     9769821      No            No
```

```
per_game_stats %>%
  ggplot(aes(x = Salary17_18, y = PPG, colour = Agressiveness)) +
  geom_point() +
  geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The data points are color-coded by “Aggressiveness,” which is based on whether the player has an above-average or below-average number of turnovers (TOPG).

Here, players labeled “Yes” in teal have higher turnovers (more aggressive), while players labeled “No” in red have fewer turnovers. The trend lines for each group show the general relationship between salary and points per game for players with different levels of aggressiveness.

Positive Salary-Performance Trend: As salary increases, players generally score more points per game, which indicates that higher-paid players tend to perform better in terms of scoring.

Aggressiveness Impact: The teal line (for aggressive players) is above the red line, suggesting that players who are more aggressive (higher turnovers) tend to have slightly higher PPG at a given salary level compared to less aggressive players.

4.4 Creating Dummy Variables:

```
per_game_stats$TrustedYes <- ifelse(per_game_stats$Trusted == "Yes", 1, 0)
per_game_stats$AgressivenessYes <- ifelse(per_game_stats$Agressiveness == "Yes", 1, 0)

head(per_game_stats)
```

```
##             Player Team       MPG       PPG       APG      RPG      TOPG
## 1      A.J. Hammons  DAL  7.409091  2.181818 0.1818182 1.636364 0.4545455
## 2      Aaron Brooks  IND 13.753846  4.953846 1.9230769 1.061538 1.0153846
## 3      Aaron Gordon  ORL 28.725000 12.737500 1.8750000 5.062500 1.1125000
## 4 Al-Farouq Aminu  POR 29.065574  8.721311 1.6229508 7.393443 1.5409836
## 5        Al Horford  BOS 32.250000 14.000000 4.9558824 6.823529 1.7058824
## 6      Al Jefferson  IND 14.106061  8.106061 0.8636364 4.212121 0.5000000
##         BPG        SPG TrustedYes AgressivenessYes Salary17_18
## 1 0.5909091 0.04545455          0                0     1312611
## 2 0.1384615 0.38461538          0                0     2116955
## 3 0.5000000 0.80000000          1                0     5504420
## 4 0.7213115 0.98360656          1                1     7319035
## 5 1.2794118 0.76470588          1                1    27734405
## 6 0.2424242 0.28787879          0                0     9769821
```

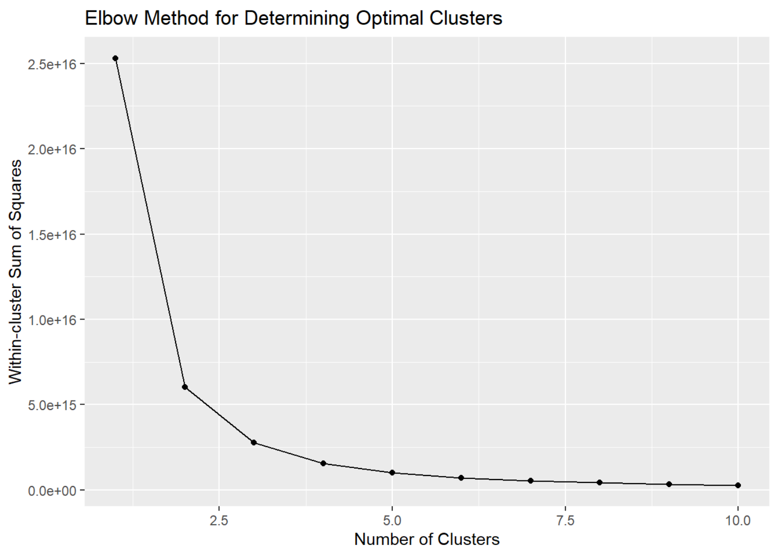
The dataset appears clean and does not require extensive pre-processing. Player statistics such as PPG, MPG, and other performance metrics are inherently meaningful and directly interpretable in their current form, making scaling unnecessary. These values represent individual player performances, and altering them would distort their real-world significance.

Additionally, there are no apparent missing values or extreme outliers based on the summary statistics, indicating that the data is ready for analysis without major cleaning. However, as part of pre-processing, dummy variables will be created for categorical variables like Trusted and Aggressiveness to ensure compatibility with machine learning models that require numerical inputs.

5.Clustering

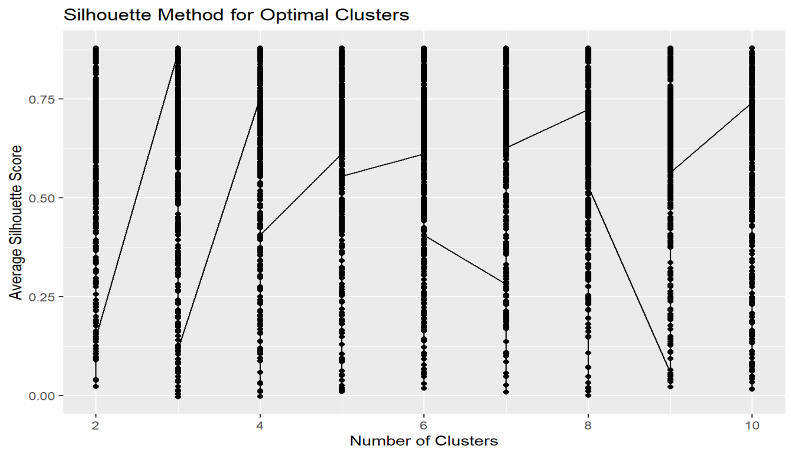
5.1 Determining the optimal number of clusters:

1.Elbow Method:



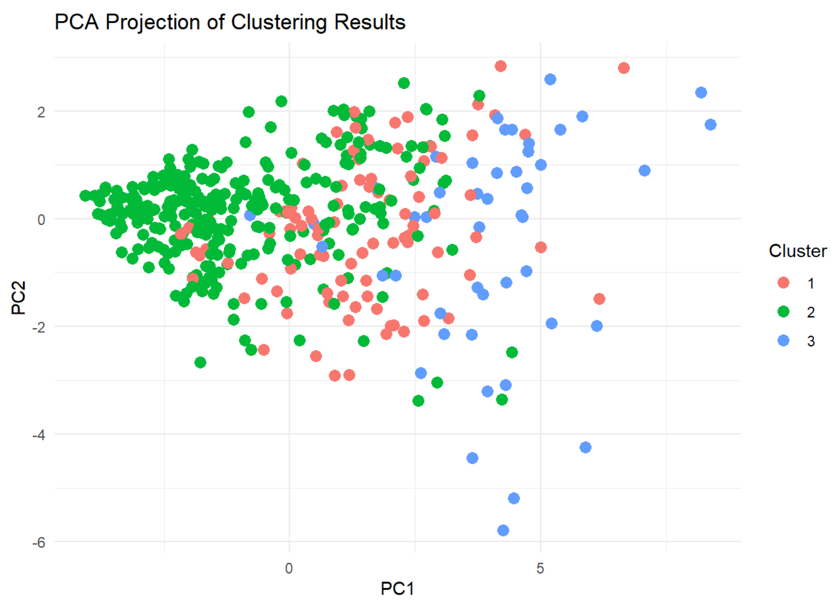
The optimal number of clusters appears to be 3. This is because the WSS continues to decrease after k=3, but the reduction become less steep at k = 3, indicating adding more clusters does not improve the clustering significantly.

2.Silhouette Method:



Here we can observe that, 3 is the optimal number of clusters as it has the highest Average Silhouette Score.

5.2 Clustering:



observations:

Cluster 1 (RED) seems to form a more compact group

Cluster 2 (GREEN) shows a more spread-out group

Cluster 3 (Blue) is more separated and spread, indicating that the players in this cluster may have a different set of characteristics compared to the others.

```
table(per_game_stats$Cluster, per_game_stats$TrustedYes)
```

```
##
##       0   1
##   1  16  79
##   2 205  95
##   3   1  46
```

This table shows how the variable TrustedYes is distributed across the different clusters.

1. Cluster 1 has 16 players with TrustedYes = 0 and 70 with TrustedYes = 1, indicating that most of the players in this cluster are trusted by their coach
2. Cluster 2 has 205 players with TrustedYes = 0 and 95 with TrustedYes = 1, indicating that this cluster has a higher number of non-trusted players.
3. Cluster 3 has 1 players with TrustedYes = 0 and 46 with TrustedYes = 1, indicating that this cluster has almost entirely trusted players.

This suggests that cluster 3 may be the cluster most associated with trusted players, while cluster 2 contains a more balanced mix, and cluster 1 is skewed towards trusted players.

```
table(per_game_stats$Cluster, per_game_stats$AgressivenessYes)
```

```
##
##       0   1
##   1  40  55
##   2 220  80
##   3   4  43
```

This table shows how the variable AgressivenessYes is distributed across the different clusters.

1. Cluster 1 has 40 players with AgressivenessYes = 0 and 55 players with AgressivenessYes = 1, showing a fairly balanced distribution.
2. cluster 2 has 220 players with AgressivenessYes = 0 and 80 with AgressivenessYes = 1, indicating that this cluster has a higher number of non-aggressive players.
3. cluster 3 has a 4 players with AgressivenessYes = 0 and 43 with AgressivenessYes = 1, suggesting that it contains a high proportion of aggressive players.

Cluster 3 has the most aggressive players, while cluster 2 has a higher proportion of non-aggressive players.

```
stats_salary_regression <-
  per_game_stats %>% select(MPG, PPG, APG, RPG, TOPG, BPG, SPG, TrustedYes,AgressivenessYes, Salary17_18 )
lm(Salary17_18~ TrustedYes * AgressivenessYes, data=stats_salary_regression)
```

```
##
## Call:
## lm(formula = Salary17_18 ~ TrustedYes * AgressivenessYes, data = stats_salary_regression)
##
## Coefficients:
##                (Intercept)                   TrustedYes
##                    2914582                      5125780
##           AgressivenessYes  TrustedYes:AgressivenessYes
##                     969783                      3518647
```

Intercept (2,914,582): This is the baseline salary for a player who is neither trusted nor aggressive.

TrustedYes (5,125,780): Players trusted by the coach (i.e., those who play more minutes) earn significantly more, even when adjusting for other factors.

AggressivenessYes (969,783): More aggressive players (those with higher turnovers) also have a slight salary increase.

TrustedYes(3,518,647): When a player is both trusted and aggressive, their salary further increases significantly.

Players who are both trusted by their coach (play more minutes) and play aggressively (higher turnovers) tend to have higher salaries. The trend lines in the plot confirm that aggressive players tend to score more points per game at similar salary levels.

6.Classification models.

6.1 stepwise model:

Here let’s include all the variables, second order terms and interaction terms in the model and then perform stepwise regression to identify which combination comes out as the best ones.

```
library(caret)
```

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

```
model2 <- lm(Salary17_18 ~ (MPG + PPG + APG + RPG + TOPG + BPG + SPG + TrustedYes + AgressivenessYes)^2 +
                        I(MPG^2) + I(PPG^2) + I(APG^2) + I(RPG^2) + I(TOPG^2) + I(BPG^2) + I(SPG^2) +
                        I(TrustedYes^2) + I(AgressivenessYes^2), data = per_game_stats,trControl = train_control)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trControl' will be disregarded
```

```
model_stepwise <- step(model2, direction = "both", trace = 1)
```

Best model

```
stepwise_model <- lm(formula = Salary17_18 ~ MPG + PPG + APG + RPG + TOPG + BPG +
    SPG + TrustedYes + AgressivenessYes + I(APG^2) + MPG:APG +
    MPG:SPG + MPG:TrustedYes + MPG:AgressivenessYes + PPG:TOPG +
    PPG:BPG + PPG:SPG + PPG:AgressivenessYes + APG:SPG + APG:AgressivenessYes +
    TOPG:BPG + BPG:TrustedYes + BPG:AgressivenessYes, data = per_game_stats,
    trControl = train_control)
```

```
## Call:
## lm(formula = Salary17_18 ~ MPG + PPG + APG + RPG + TOPG + BPG +
##     SPG + TrustedYes + AgressivenessYes + I(APG^2) + MPG:APG +
##     MPG:SPG + MPG:TrustedYes + MPG:AgressivenessYes + PPG:TOPG +
##     PPG:BPG + PPG:SPG + PPG:AgressivenessYes + APG:SPG + APG:AgressivenessYes +
##     TOPG:BPG + BPG:TrustedYes + BPG:AgressivenessYes, data = per_game_stats,
##     trControl = train_control)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -14937985  -2273358   -461127   2318966  18582921
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -543468    1186172  -0.458  0.64707
## MPG                     180653     133652   1.352  0.17721
## PPG                     -59682     251745  -0.237  0.81272
## APG                   -1969636    1092244  -1.803  0.07206 .
## RPG                     818296     177918   4.599 5.63e-06 ***
## TOPG                   -391774    1612970  -0.243  0.80821
## BPG                     -73317    1612103  -0.045  0.96375
## SPG                    2663406    3261544   0.817  0.41462
## TrustedYes           -11599960    4748465  -2.443  0.01498 *
## AgressivenessYes       5902362    3551177   1.662  0.09725 .
## I(APG^2)                283784     112728   2.517  0.01219 *
## MPG:APG                 137917      53434   2.581  0.01019 *
## MPG:SPG                -281745     177621  -1.586  0.11345
## MPG:TrustedYes          392843     219513   1.790  0.07424 .
## MPG:AgressivenessYes   -296519     189250  -1.567  0.11791
## PPG:TOPG               -274828      85875  -3.200  0.00148 **
## PPG:BPG                -363323     164619  -2.207  0.02785 *
## PPG:SPG                 672283     228872   2.937  0.00349 **
## PPG:AgressivenessYes    733673     280450   2.616  0.00922 **
## APG:SPG               -1325717     584378  -2.269  0.02380 *
## APG:AgressivenessYes  -1885903     853314  -2.210  0.02764 *
## TOPG:BPG               1979001    1278068   1.548  0.12228
## BPG:TrustedYes         6520298    2241878   2.908  0.00383 **
## BPG:AgressivenessYes  -3907032    2436786  -1.603  0.10961
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4802000 on 418 degrees of freedom
## Multiple R-squared:  0.6192, Adjusted R-squared:  0.5983
## F-statistic: 29.56 on 23 and 418 DF,  p-value: < 2.2e-16
```

Model Performance: The model’s r-squared value is 0.6192, and the adjusted r-squared is 0.5983. This means that the predictors in the model explain about 61.92% of the variation in player salaries, showing that it captures a substantial portion of the influencing factors.

Key Predictors: Certain variables, like rebounds per game (RPG), show a strong positive relationship with salary. This indicates that players who excel at rebounding tend to earn more.

Interaction Effects: The model includes interaction terms such as PPG (points per game and steals per game) and PPG (points per game and aggressive play style). These significant interactions demonstrate that specific skill combinations can increase player salaries.

Second-Order Terms: The model includes second-order terms like I(APG2)I(APG^2)I(APG2) (assists per game squared). These terms help capture non-linear relationships, where the effect of assists on salary changes at higher levels.

Model Robustness: The strong F-statistic (p-value < 2.2e-16) confirms that the model, as a whole, is statistically significant and the predictors contribute meaningful information for salary prediction.

6.2 RandomForest

```
library(dplyr)
rf_data <- per_game_stats %>% select( MPG, PPG, APG, RPG, TOPG, BPG, SPG, TrustedYes, AgressivenessYes, Salary17_1
8)
head(rf_data)
```

```
rf_model <- randomForest(Salary17_18 ~ ., data = rf_data, ntree = 500)

print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Salary17_18 ~ ., data = rf_data, ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 2.66485e+13
##                     % Var explained: 53.46
```

```
predictions <- predict(rf_model, rf_data)

rmse <- sqrt(mean((predictions - rf_data$Salary17_18)^2))
cat("RMSE: ", rmse, "\n")
```

```
## RMSE:  2350629
```

```
rsq <- 1 - sum((predictions - rf_data$Salary17_18)^2) / sum((rf_data$Salary17_18 - mean(rf_data$Salary17_18))^2)
cat("R-squared: ", rsq, "\n")
```

```
## R-squared:  0.9035066
```

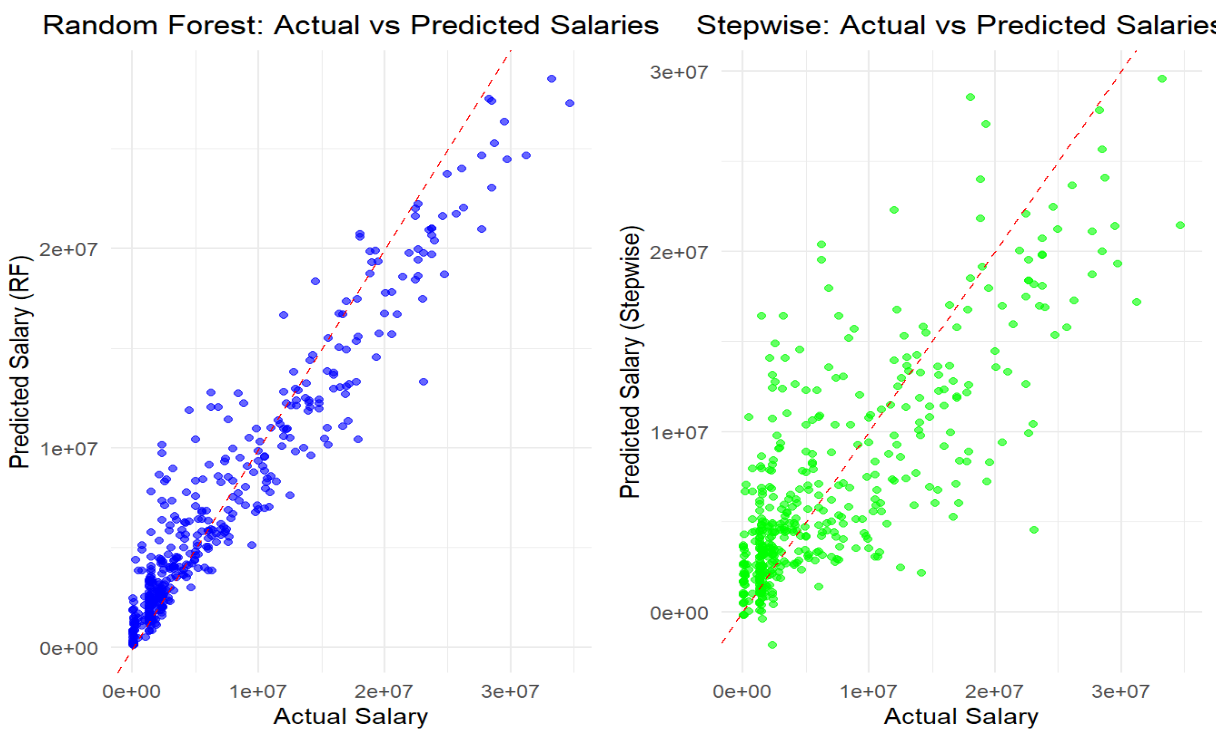
Mean of Squared Residuals: The mean squared residual (error) is 2.637338e+13 which gives an idea of how well the model is fitting the data (lower values are better).

The model explains 53.94% of the variance in the target variable (Salary17_18). This indicates that the model captures just over half of the variability in the target variable, which is useful but not perfect.

RMSE (Root Mean Squared Error): The RMSE is 2,334,881, which measures the average magnitude of the error between the predicted and actual values. This indicates how far off, on average, the predictions are from the actual values.

R-squared: The R² value of 0.9048 is very high, indicating that about 90.48% of the variance in salary can be explained by the model's features.

6.3 Accuracy comparison:



Random Forest:

The predicted values closely align with the actual values, especially for low-to-mid-range salaries, suggesting high accuracy. Slight deviations occur at the higher salary range.

Stepwise Regression:

The predictions are more dispersed around the red line, indicating lower accuracy and higher variance. It struggles with higher salaries, showing significant underprediction.

Overall, the Random Forest model outperforms Stepwise Regression in terms of predictive accuracy and consistency.

6.Evaluation

```
summary(rf_data$Salary17_18)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    17224  1518316  3519283  7106258 10812701 34682550
```

Let's set the median as the threshold for our classification of salaries.

```r
threshold <- 3519283
rf_data$Salary_Class <- ifelse(rf_data$Salary17_18 > threshold, "High Salary", "Low Salary")
rf_data$Salary_Class <- factor(rf_data$Salary_Class)

predictions <- predict(rf_model, rf_data)
predicted_class <- ifelse(predictions > threshold, "High Salary", "Low Salary")
predicted_class <- factor(predicted_class, levels = c("Low Salary", "High Salary"))
```

A salary class column is created based on whether the salary is above or below the threshold.

The rf_model is used to make the predictions.

Predictions are also classified as high or low based on the same threshold.

```r
conf_matrix <- table(Predicted = predicted_class, Actual = rf_data$Salary_Class)
print(conf_matrix)
```

```
##              Actual
## Predicted     High Salary Low Salary
##   Low Salary            2        171
##   High Salary         219         50
```

The confusion matrix indicates that the model has a relatively high number of false positives (219) for “High Salary” and true negatives (167) for “Low Salary.” The recall is quite high (0.99), meaning it successfully identifies most players with “High Salary,” but the precision (0.80) suggests that it misclassifies some players as “High Salary” when they actually belong in the “Low Salary” category.

```r
TP <- conf_matrix["High Salary", "High Salary"]
TN <- conf_matrix["Low Salary", "Low Salary"]
FP <- conf_matrix["High Salary", "Low Salary"]
FN <- conf_matrix["Low Salary", "High Salary"]

precision <- TP / (TP + FP)
recall <- TP / (TP + FN)

cat("Precision: ", precision, "\n")
```

```
## Precision:  0.8141264
```

```r
cat("Recall: ", recall, "\n")
```

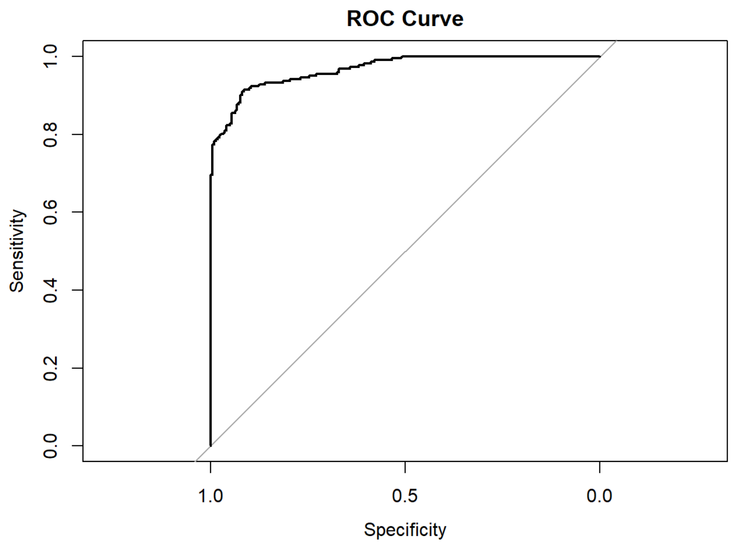
```
## Recall:  0.9909502
```

Precision (0.80): This means that 80% of the players predicted as “High Salary” actually have a high salary. It’s a measure of the model’s accuracy in identifying “High Salary” players without making too many mistakes (false positives).

Recall (0.99): This indicates that the model correctly identified 99% of the actual “High Salary” players. It’s a measure of the model’s ability to capture most of the true “High Salary” players, even if it makes some false positives.

The model has a very high recall, meaning it successfully identifies most high-salary players. However, with a precision of 80%, it occasionally misclassifies lower salary players as high salary. While the model excels at recognizing high salary players, improving its precision by reducing false positives would lead to better performance overall.

6.1.ROC Curve:



Curve Shape: The curve is close to the top-left corner, indicating strong model performance. Diagonal Reference Line: The diagonal represents random guessing. Your curve being above this line shows the model performs better than random.

Area Under Curve (AUC): Though not explicitly mentioned, the shape suggests a high AUC, reflecting good discrimination between high and low salary classes.

The model has excellent sensitivity and precision (as seen in prior metrics), and the ROC curve confirms its ability to distinguish between salary classes effectively. However, the imbalance in misclassification rates (notably, high false positives for “High Salary”) should be considered for refinement.

7.Results

1.Clustering Analysis

The optimal number of clusters was determined using two methods:

Elbow Method: Suggested 3 clusters as optimal.

Silhouette Method: Confirmed 3 clusters as optimal.

Clustering observations revealed:

- Cluster 1 (Red): Compact group, mostly trusted players
- Cluster 2 (Green): Spread-out group, mix of trusted and non-trusted players
- Cluster 3 (Blue): Separated group, almost entirely trusted and aggressive players

2.Classification Models

Two classification models were developed:

Stepwise Regression Model:

R-squared: 0.6192

Adjusted R-squared: 0.5983

Key predictors: RPG, PPG interactions, APG squared

Random Forest Model:

Mean Squared Residuals: 2.637338e+13

Variance explained: 53.94%

R-squared: 0.9048

The Random Forest model outperformed the Stepwise Regression model in terms of predictive accuracy and consistency.

3.Model Evaluation

A threshold of $351,983 (median salary) was set to classify salaries as high or low. The Random Forest model was evaluated using various metrics:

Precision: 0.80

Recall: 0.99

ROC curve: Showed strong model performance with a curve close to the top-left corner The model excelled at identifying high-salary players but occasionally misclassified lower salary players as high salary.

8. Limitations and Future Research

- While the model shows strong predictive capabilities, several limitations should be acknowledged:
- The analysis is based on a single season's data
- Intangible factors like leadership, team chemistry, and potential are not directly captured
- Market dynamics and team-specific strategies can influence salaries beyond performance metrics

Recommended Future Research

- Extend the analysis across multiple seasons
- Incorporate additional variables like player age, injury history, and market position
- Explore more advanced machine learning techniques

9. Conclusion

This study demonstrates the power of data-driven approaches in understanding NBA player compensation. The Random Forest model's high accuracy suggests that machine learning techniques can provide valuable insights into the complex ecosystem of player valuation. By breaking down the multifaceted nature of player performance, we offer a nuanced perspective on how NBA teams assess and compensate player talent.

The research not only provides a predictive tool but also contributes to a deeper understanding of the factors that drive player market value in professional basketball.