Final Report

Hongye Xu

2019/5/5

1 Introduction

1.1 Background

I have always felt that the NBA has the best data storage in the sport filed. In the beginning, I wanted to analyze the performance of the players by scrapping the data from the official NBA.stat website. However, since the NBA.stat table is in javascript format, and the official has canceled all the existing official APIs, no possible R-based crawler method has been found after the effort. Therefore, I chose an alternative, which is the basketball-reference website. This report is based on two data sources on the basketball-reference. My goal is to predict the player's salary for next season based on player performance this season.

1.2 Glossary

Abbreviation	Explanation
Pos	Position
Age	Age of Player at the start of February 1st of that season
Tm	Team
G	Games
GS	Games Started
MP	Minutes Played Per Game
FG	Field Goals Per Game
FGA	Field Goal Attempts Per Game
FG%	Field Goal Percentage
3P	3-Point Field Goals Per Game
3PA	3-Point Field Goal Attempts Per Game
3P%	FG% on 3-Pt FGAs.
2P	2-Point Field Goals Per Game

2PA	2-Point Field Goal Attempts Per Game
eFG%	Effective Field Goal Percentage
FT	Free Throws Per Game
FTA	Free Throw Attempts Per Game
FT%	Free Throw Percentage
ORB	Offensive Rebounds Per Game
DRB	Defensive Rebounds Per Game
TRB	Total Rebounds Per Game
AST	Assists Per Game
STL	Steals Per Game
BLK	Blocks Per Game
TOV	Turnovers Per Game
PF	Personal Fouls Per Game
PTS	Points Per Game

Github Link

https://github.com/szxuhongye/NBA-Player-Salary-Predicton.git (https://github.com/szxuhongye/NBA-Player-Salary-Predicton.git)

2 Preparation

2.1 Required Packages

```r
library(rvest)
library(magrittr)
library(tibble)
library(dplyr)
library(stringr)
library(data.table)
library(corrplot)
library(GGally)
library(tidyverse)
library(PerformanceAnalytics)
library(plotly)
library(caret)
library(MASS)
```

2.2 Data Scraping and Cleaning

2.2.1 Players' Regular Season Data

```r
#NBA Season18-19 Players stat(Regular Season)
url<- "https://www.basketball-reference.com/leagues/NBA_2019_per_game.html"
player_season_v1 <- url %>% read_html()%>%
  html_nodes(xpath = '//*[@id="per_game_stats"]')%>%
  html_table()
```

```
# romove the rank column and row number
player_season_v2 <- player_season_v1[[1]]%>%dplyr::select(-Rk)
player_season_v3 <-player_season_v2[- grep("Player", player_season_v2$Player),]
# make some columns numeric
name <- c("Age","G","GS","MP","FG","FGA","FG%","3P","3PA","3P%","2P","2PA","2P%","eFG
%","FT","FTA","FT%","ORB","DRB","TRB","AST","STL","BLK","TOV","PF","PTS")
player_season_v3[name] <- sapply(player_season_v3[name],as.numeric)
rownames(player_season_v3) <- NULL

# There will be multiple rows of data for players transferred during the season. We o
nly keep the one with the largest number of games played(which is the stat of this pl
ayer throughout the season).
player_season_tidy <- player_season_v3 %>% group_by(Player)%>%
   mutate(rank = min_rank(desc(G))) %>%
   filter(rank == 1) %>%
   dplyr::select(-rank)
# No Scale data(for data visulization)
player_season_tidy <- player_season_tidy %>% filter(!is.na(`3P%`) & !is.na(`FT%`)& !i
s.na(`2P%`)) %>% as.data.frame(.)
head(player_season_tidy)
```

```
##          Player Pos Age  Tm  G GS   MP  FG  FGA   FG%  3P 3PA   3P%  2P
## 1 Alex Abrines  SG  25 OKC 31  2 19.0 1.8  5.1 0.357 1.3 4.1 0.323 0.5
## 2   Quincy Acy  PF  28 PHO 10  0 12.3 0.4  1.8 0.222 0.2 1.5 0.133 0.2
## 3 Jaylen Adams  PG  22 ATL 34  1 12.6 1.1  3.2 0.345 0.7 2.2 0.338 0.4
## 4 Steven Adams   C  25 OKC 80 80 33.4 6.0 10.1 0.595 0.0 0.0 0.000 6.0
## 5  Bam Adebayo   C  21 MIA 82 28 23.3 3.4  5.9 0.576 0.0 0.2 0.200 3.4
## 6    Deng Adel  SF  21 CLE 19  3 10.2 0.6  1.9 0.306 0.3 1.2 0.261 0.3
##    2PA   2P%  eFG%  FT FTA   FT% ORB DRB TRB AST STL BLK TOV  PF  PTS
## 1  1.0 0.500 0.487 0.4 0.4 0.923 0.2 1.4 1.5 0.6 0.5 0.2 0.5 1.7  5.3
## 2  0.3 0.667 0.278 0.7 1.0 0.700 0.3 2.2 2.5 0.8 0.1 0.4 0.4 2.4  1.7
## 3  1.1 0.361 0.459 0.2 0.3 0.778 0.3 1.4 1.8 1.9 0.4 0.1 0.8 1.3  3.2
## 4 10.1 0.596 0.595 1.8 3.7 0.500 4.9 4.6 9.5 1.6 1.5 1.0 1.7 2.6 13.9
## 5  5.7 0.588 0.579 2.0 2.8 0.735 2.0 5.3 7.3 2.2 0.9 0.8 1.5 2.5  8.9
## 6  0.7 0.385 0.389 0.2 0.2 1.000 0.2 0.8 1.0 0.3 0.1 0.2 0.3 0.7  1.7
```

2.2.2 Scale

Considering that regression analysis is mainly used in this report, i try to scale some features.

```
#Here i use the Player column as the rowname since the scale function need the whole
matrix features to be numeric
scale <- player_season_tidy %>% dplyr::select(Player ,Age,MP,`2P%`,`3P%`,`FT%`,TRB:PT
S)
rownames(scale)<-scale[,1]
scale1 <- scale[,-1]%>% as.matrix(.)%>%
  scale(.) %>%
  as.data.frame(.)
head(scale1)
```

```
##                     Age         MP         2P%        3P%         FT%
## Alex Abrines -0.2348656 -0.1678429 -0.05111731  0.1079674  1.30497350
## Quincy Acy    0.4772268 -0.9471834  2.01348589 -1.5619660 -0.28148044
## Jaylen Adams -0.9469579 -0.9122876 -1.76955950  0.2398043  0.27342273
## Steven Adams -0.2348656  1.5071577  1.13572046 -2.7309194 -1.70430908
## Bam Adebayo  -1.1843221  0.3323309  1.03681731 -0.9730947 -0.03248542
## Deng Adel    -1.1843221 -1.1914543 -1.47285006 -0.4369582  1.85276253
##                     TRB         AST        STL         BLK        TOV
## Alex Abrines -0.9119870 -0.81732136 -0.3701968 -0.49747590 -0.7975707
## Quincy Acy   -0.4970106 -0.70641502 -1.3498261  0.02415998 -0.9238106
## Jaylen Adams -0.7874941 -0.09643014 -0.6151041 -0.75829384 -0.4188509
## Steven Adams  2.4078238 -0.26278966  2.0788766  1.58906762  0.7173087
## Bam Adebayo   1.4948758  0.06992937  0.6094326  1.06743174  0.4648288
## Deng Adel    -1.1194751 -0.98368087 -1.3498261 -0.49747590 -1.0500506
##                      PF         PTS
## Alex Abrines -0.1395071 -0.64209057
## Quincy Acy    0.7994827 -1.23613639
## Jaylen Adams -0.6760726 -0.98861730
## Steven Adams  1.0677655  0.77701887
## Bam Adebayo   0.9336241 -0.04804476
## Deng Adel    -1.4809210 -1.23613639
```

2.2.3 Players' Salaries

```
# Players' salaries from season 18-19
url1 <- "https://www.basketball-reference.com/contracts/players.html"
salaries <- url1 %>% read_html()%>%
  html_nodes(xpath = '//*[@id="player-contracts"]')%>%
  html_table()
salaries_v2 <- salaries[[1]]
colnames(salaries_v2) <- NULL
names(salaries_v2) <- as.character(unlist(salaries_v2[1,]))
salaries_v2 <- salaries_v2[-1,] %>%
  dplyr::select(-Rk)
rownames(salaries_v2) <- NULL
head(salaries_v2)
```

```
##                Player  Tm     2018-19     2019-20     2020-21     2021-22
## 1     Stephen Curry GSW $37,457,154 $40,231,758 $43,006,362 $45,780,966
## 2        Chris Paul HOU $35,654,150 $38,506,482 $41,358,814 $44,211,146
## 3 Russell Westbrook OKC $35,654,150 $38,178,000 $41,006,000 $43,848,000
## 4      LeBron James LAL $35,654,150 $37,436,858 $39,219,565 $41,002,273
## 5     Blake Griffin DET $32,088,932 $34,234,964 $36,595,996 $38,957,028
## 6    Gordon Hayward BOS $31,214,295 $32,700,690 $34,187,085
##       2022-23 2023-24 Signed Using   Guaranteed
## 1                      Bird Rights $166,476,240
## 2                                  $159,730,592
## 3 $46,662,000          Bird Rights $158,686,150
## 4                                  $113,310,573
## 5                      Bird Rights $102,919,892
## 6                        Cap space  $63,914,985
```

```
# Change 2018-19 salaries to be numeric data
salaries_v2$`2018-19` <- salaries_v2$`2018-19` %>%
  str_replace_all(.,"\\,","")%>%
  str_replace_all(., "\\$","")%>%
  as.numeric(.)
```

```
## Warning in function_list[[k]](value): NAs introduced by coercion
```

```
# Delete rows containing missing values (due to duplicate headers)
salaries_v2 <-salaries_v2[- grep("Player", player_season_v2$Player),]
salaries_v3 <-na.omit(salaries_v2)
#remove duplicated row(only keep the highest one)
salaries_v4 <- salaries_v3 %>% group_by(Player) %>%
  mutate(rank = min_rank(desc(`2018-19`))) %>%
   filter(rank == 1) %>%
   dplyr::select(-rank)
head(salaries_v4)
```

```
## # A tibble: 6 x 10
## # Groups:   Player [6]
##   Player Tm    `2018-19` `2019-20` `2020-21` `2021-22` `2022-23` `2023-24`
##   <chr>  <chr>     <dbl> <chr>     <chr>     <chr>     <chr>     <chr>
## 1 Steph… GSW    37457154 $40,231,… $43,006,… $45,780,… ""        ""
## 2 Chris… HOU    35654150 $38,506,… $41,358,… $44,211,… ""        ""
## 3 Russe… OKC    35654150 $38,178,… $41,006,… $43,848,… $46,662,… ""
## 4 LeBro… LAL    35654150 $37,436,… $39,219,… $41,002,… ""        ""
## 5 Blake… DET    32088932 $34,234,… $36,595,… $38,957,… ""        ""
## 6 Gordo… BOS    31214295 $32,700,… $34,187,… ""        ""        ""
## # ... with 2 more variables: `Signed Using` <chr>, Guaranteed <chr>
```

```
salaries_tidy <- salaries_v4 %>%
  dplyr::select(Player,Tm,`2018-19`)%>%
  as.data.frame(.)
# Following two players' salaries are not changed after transfer(So i delete them to
avoid duplication of one player after doing table merging)
salaries_tidy <- salaries_tidy[!(salaries_tidy$Player=="John Jenkins" & salaries_tidy
$Tm=="NYK"),]
salaries_tidy <- salaries_tidy[!(salaries_tidy$Player=="Emanuel Terry" & salaries_tid
y$Tm=="MIA"),]
salaries_tidy %>% mutate(duplicated(Player))%>%
  filter(`duplicated(Player)`== TRUE)
```

```
## [1] Player             Tm                  2018-19
## [4] duplicated(Player)
## <0 rows> (or 0-length row.names)
```

```
head(salaries_tidy)
```

```
##              Player  Tm  2018-19
## 1     Stephen Curry GSW 37457154
## 2        Chris Paul HOU 35654150
## 3 Russell Westbrook OKC 35654150
## 4      LeBron James LAL 35654150
## 5     Blake Griffin DET 32088932
## 6    Gordon Hayward BOS 31214295
```

Finally, there is no duplicate player data.

2.2.4 Merging Data

```
#non_scale data
stats_for_visualized <- merge(player_season_tidy, salaries_tidy, by.x = "Player", by.
y = "Player")
names(stats_for_visualized)[31] <- "salary18_19"
stats_for_visualization <- stats_for_visualized[-30]
head(stats_for_visualization)
```

```
##            Player Pos Age Tm.x  G GS   MP  FG  FGA   FG%  3P 3PA   3P%  2P
## 1    Aaron Gordon  PF  23  ORL 78 78 33.8 6.0 13.4 0.449 1.6 4.4 0.349 4.5
## 2   Aaron Holiday  PG  22  IND 50  0 12.9 2.1  5.2 0.401 0.9 2.5 0.339 1.2
## 3     Abdel Nader  SF  25  OKC 61  1 11.4 1.5  3.5 0.423 0.5 1.6 0.320 1.0
## 4      Al Horford   C  32  BOS 68 68 29.0 5.7 10.6 0.535 1.1 3.0 0.360 4.6
## 5 Al-Farouq Aminu  PF  28  POR 81 81 28.3 3.2  7.3 0.433 1.2 3.5 0.343 2.0
## 6      Alec Burks  SG  27  TOT 64 24 21.5 3.0  7.4 0.405 1.0 2.6 0.363 2.0
##   2PA   2P%  eFG%  FT FTA   FT% ORB DRB TRB AST STL BLK TOV  PF  PTS
## 1 9.0 0.499 0.507 2.4 3.2 0.731 1.7 5.7 7.4 3.7 0.7 0.7 2.1 2.2 16.0
## 2 2.7 0.459 0.483 0.8 1.0 0.820 0.1 1.2 1.3 1.7 0.4 0.3 0.8 1.4  5.9
## 3 1.9 0.513 0.498 0.4 0.6 0.750 0.2 1.7 1.9 0.3 0.3 0.2 0.4 1.1  4.0
## 4 7.6 0.604 0.586 1.1 1.4 0.821 1.8 5.0 6.7 4.2 0.9 1.3 1.5 1.9 13.6
## 5 3.9 0.514 0.514 1.9 2.1 0.867 1.4 6.1 7.5 1.3 0.8 0.4 0.9 1.8  9.4
## 6 4.8 0.428 0.469 1.8 2.2 0.823 0.5 3.2 3.7 2.0 0.6 0.3 1.0 1.4  8.8
##   salary18_19
## 1    21590909
## 2     1911960
## 3     1378242
## 4    28928710
## 5     6957105
## 6    11536515
```

```
#scale data
salaries1 <- salaries_tidy
rownames(salaries1) <- salaries1[,1]
salaries2 <- salaries1[,-1]
stats_scale <- merge(scale1, salaries2, by="row.names")
names(stats_scale)[15] <- "salary18_19"
stats_scale <- stats_scale[-14]
head(stats_scale)
```

```
##         Row.names        Age         MP        2P%        3P%         FT%
## 1    Aaron Gordon -0.7095938  1.5536855 -0.0634802 0.33648464 -0.06094200
## 2   Aaron Holiday -0.9469579 -0.8773917 -0.5579959 0.24859341  0.57221675
## 3     Abdel Nader -0.2348656 -1.0518710  0.1096003 0.08160007  0.07422672
## 4      Al Horford  1.4266833  0.9953519  1.2346236 0.43316500  0.57933089
## 5 Al-Farouq Aminu  0.4772268  0.9139283  0.1219632 0.28374990  0.90658148
## 6      Alec Burks  0.2398627  0.1229558 -0.9412456 0.45953237  0.59355918
##               TRB         AST        STL         BLK        TOV           PF
## 1  1.5363734745  0.90172692  0.1196179  0.80661380  1.2222684  0.531199926
## 2 -0.9949822302 -0.20733649 -0.6151041 -0.23665796 -0.4188509 -0.541931237
## 3 -0.7459964232 -0.98368087 -0.8600115 -0.49747590 -0.9238106 -0.944355424
## 4  1.2458900330  1.17899277  0.6094326  2.37152144  0.4648288  0.128775740
## 5  1.5778711090 -0.42914917  0.3645252  0.02415998 -0.2926109 -0.005365656
## 6  0.0009609979 -0.04097697 -0.1252894 -0.23665796 -0.1663710 -0.541931237
##             PTS salary18_19
## 1  1.12354560    21590909
## 2 -0.54308294     1911960
## 3 -0.85660712     1378242
## 4  0.72751506    28928710
## 5  0.03446161     6957105
## 6 -0.06454603    11536515
```

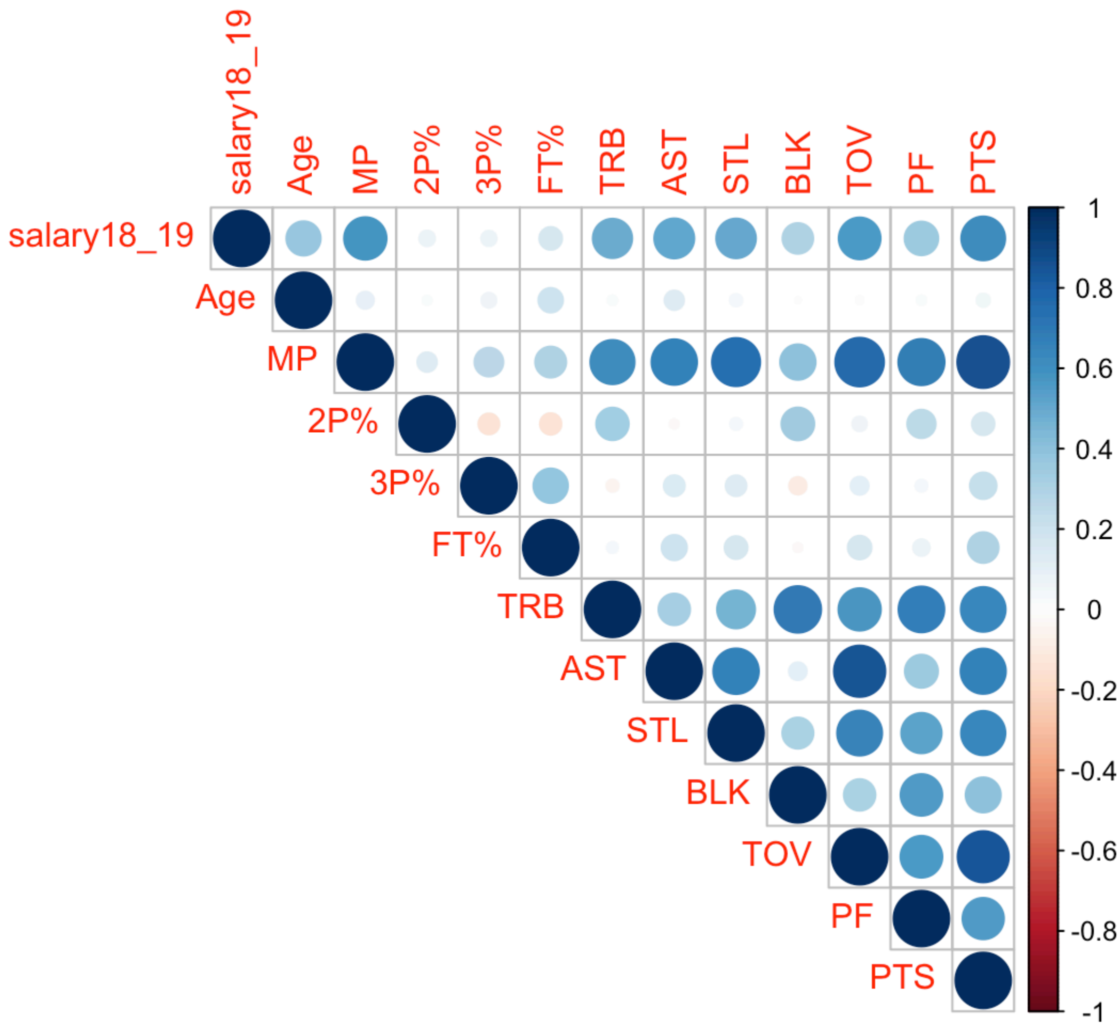
2.2.5 Save No_scale Data Into CSV

```
write.csv(stats_for_visualization,'18-19players_stat.csv')
```

3 Correlation Check

3.1 Frist Check

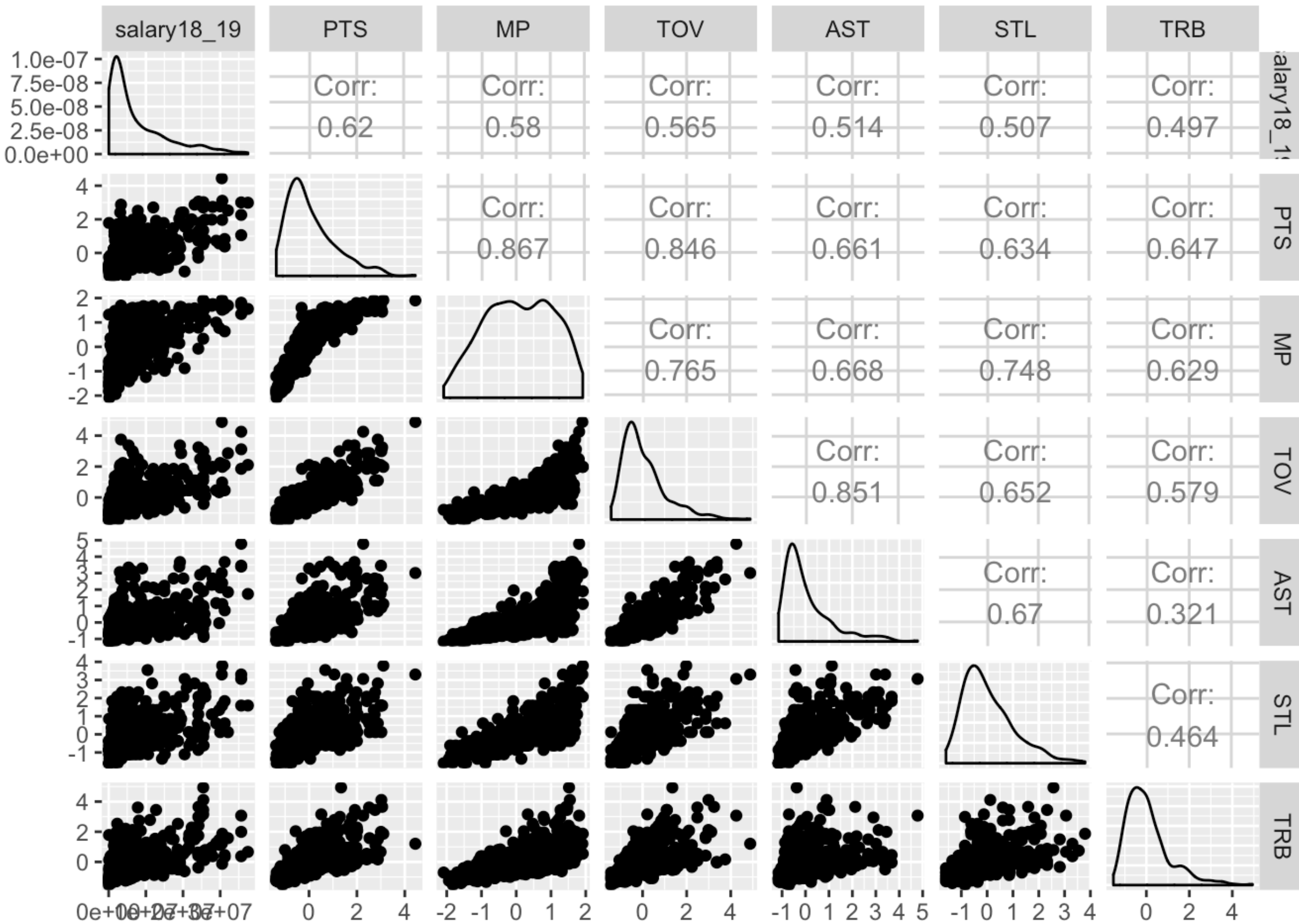
```
corrplot(cor(stats_scale %>%
               dplyr::select(salary18_19, Age:PTS, contains("%")),
             use = "complete.obs"),
         method = "circle",type = "upper")
```



The features that have strong correlation with salary are:PTS,TOV,STL,AST,TRB and MP. Besides, MP is strongly correlated with multiple features and may have multiple collinearities(This is in line with our common sense. The more time we play, the better the data will be). What I didn't expect was that the correlation between field goal and salary was not high, that is to say, the output of players influenced the salary of players more than efficiency.

3.2 Second Check

```
stats_salary_cor <-
  stats_scale %>%
  dplyr::select(salary18_19,PTS, MP, TOV, AST, STL, TRB)
ggpairs(stats_salary_cor)
```




```
cor(stats_salary_cor)[,"salary18_19"]
```

```
## salary18_19         PTS          MP         TOV         AST         STL
##   1.0000000   0.6198192   0.5803967   0.5645525   0.5142283   0.5066299
##         TRB
##   0.4972563
```

Correlation strength is: PTS > MP > TOV > AST > STL > TRB There's also one thing that surprises me: the number of players'turnivers is positively correlated with their salaries. I mean, generally speaking, assuming that a player's turnover rate is constant, the total number of turnovers will increase as his minutes played increases, and important players will have higher minutes played and higher salaries.

4 Data Visualization

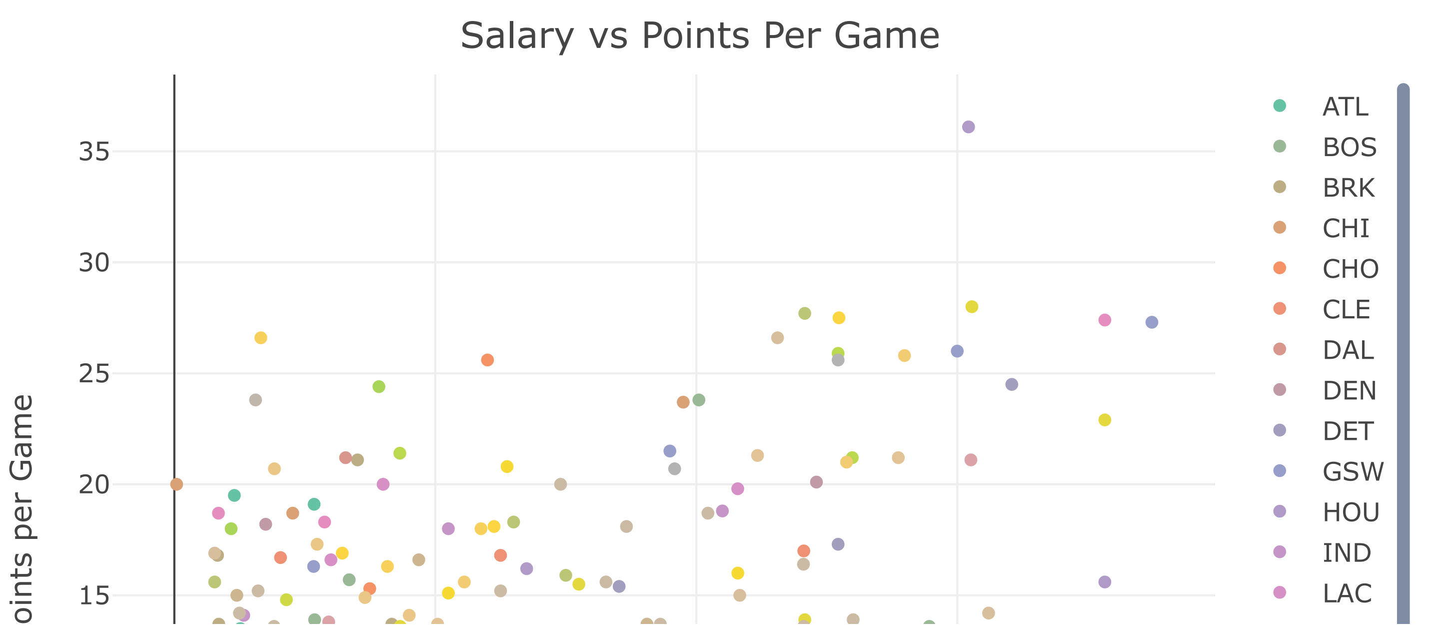
4.1 Interactive Plot

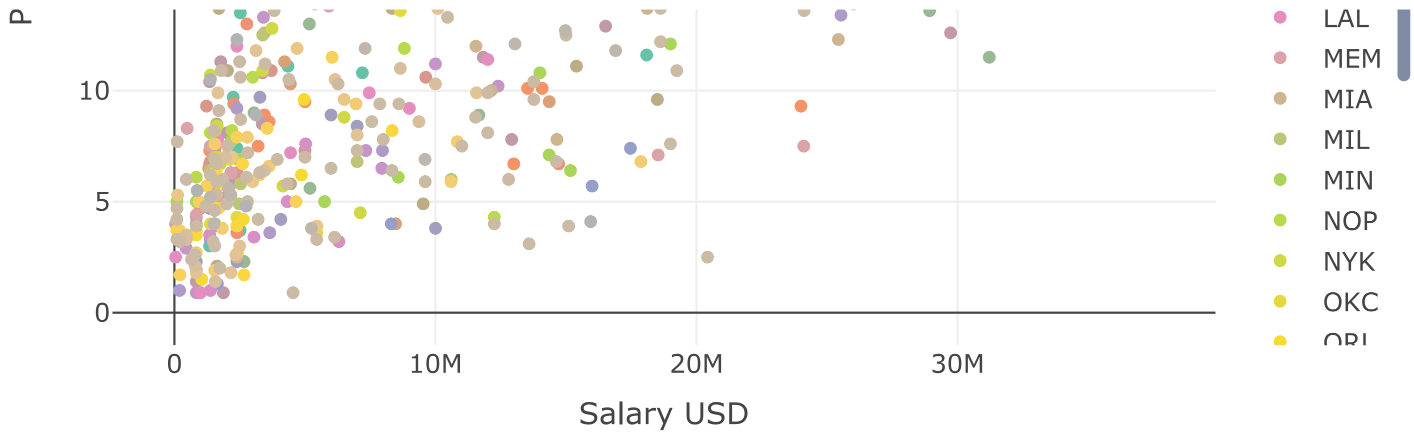
```
names(stats_for_visualization )[4] <- "Team"
plot_ly(data = stats_for_visualization , x = ~salary18_19, y = ~PTS, color = ~Team,
        hoverinfo = "text",
        text = ~paste("Player: ", Player,
                      "<br>Salary: ", format(salary18_19, big.mark = ","),"$",
                      "<br>PTS: ", round(PTS, digits = 3),
                      "<br>Team: ", Team)) %>%
  layout(
    title = "Salary vs Points Per Game",
    xaxis = list(title = "Salary USD"),
    yaxis = list(title = "Points per Game")
  )
```

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

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

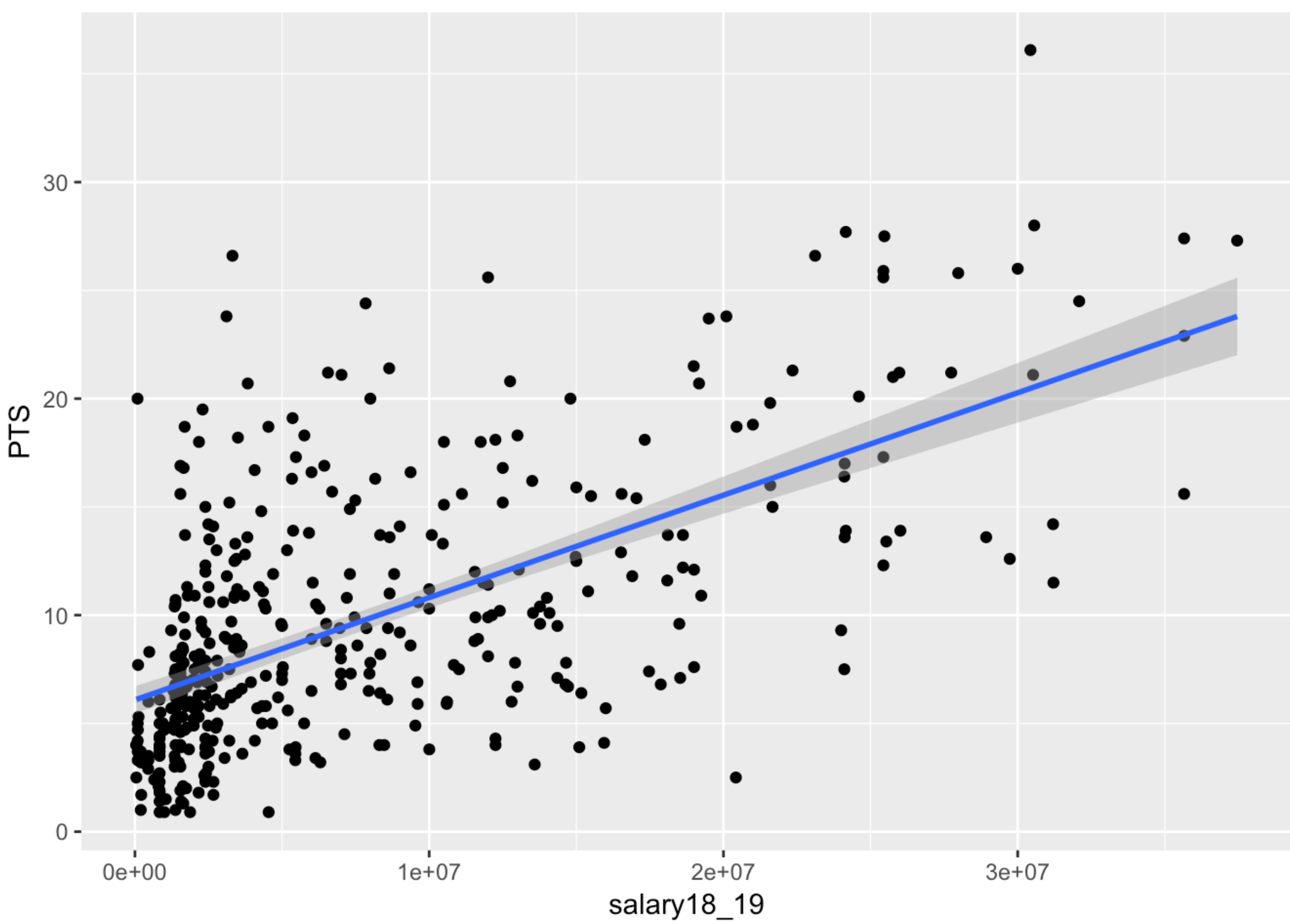
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for p
alette Set2 is 8
## Returning the palette you asked for with that many colors

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for p
alette Set2 is 8
## Returning the palette you asked for with that many colors
```





4.2 Scatter Plot With Regression Line

```
stats_for_visualization %>%
  ggplot(aes(x = salary18_19, y = PTS)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Under the simple linear model, we can understand that the fitted curve represents the average level of the league, and the player below the curve performs worse than the expected performance corresponding to the salary. We can check their name by hovering on the points in the interactive plot. It includes a lot of All-Star players, such as Chris Paul, Kyle Lowry, Al Horford, Gordon Haywood(The Celtics are unlucky) and etc. However, it only considers the scoring feature, and does not fully reflect the players'influence on the field.

5 Multiple Regression

```
regression <- stats_scale %>% dplyr::select(salary18_19,PTS, MP, TOV, AST, STL, TRB)
  lm(salary18_19~ PTS + MP + TOV + AST + STL + TRB, data =regression)
```

```
##
## Call:
## lm(formula = salary18_19 ~ PTS + MP + TOV + AST + STL + TRB,
##     data = regression)
##
## Coefficients:
## (Intercept)          PTS           MP          TOV          AST
##     7114340      3626578      -546186     -2022044      2571555
##         STL          TRB
##      833962      1888579
```

From here, we can see that points per game is the most significant feature of positive impact, while turnovers per game is the most significant feature of negative impact. However, simple multiple regression also has some problems, that is, there are multiple collinearities.

5.1 Player 's Importance And Incautiousness

Here we make two definitions that a player is "important" if his minutes played is above average and is "incautious" if his turnover per game is above average.

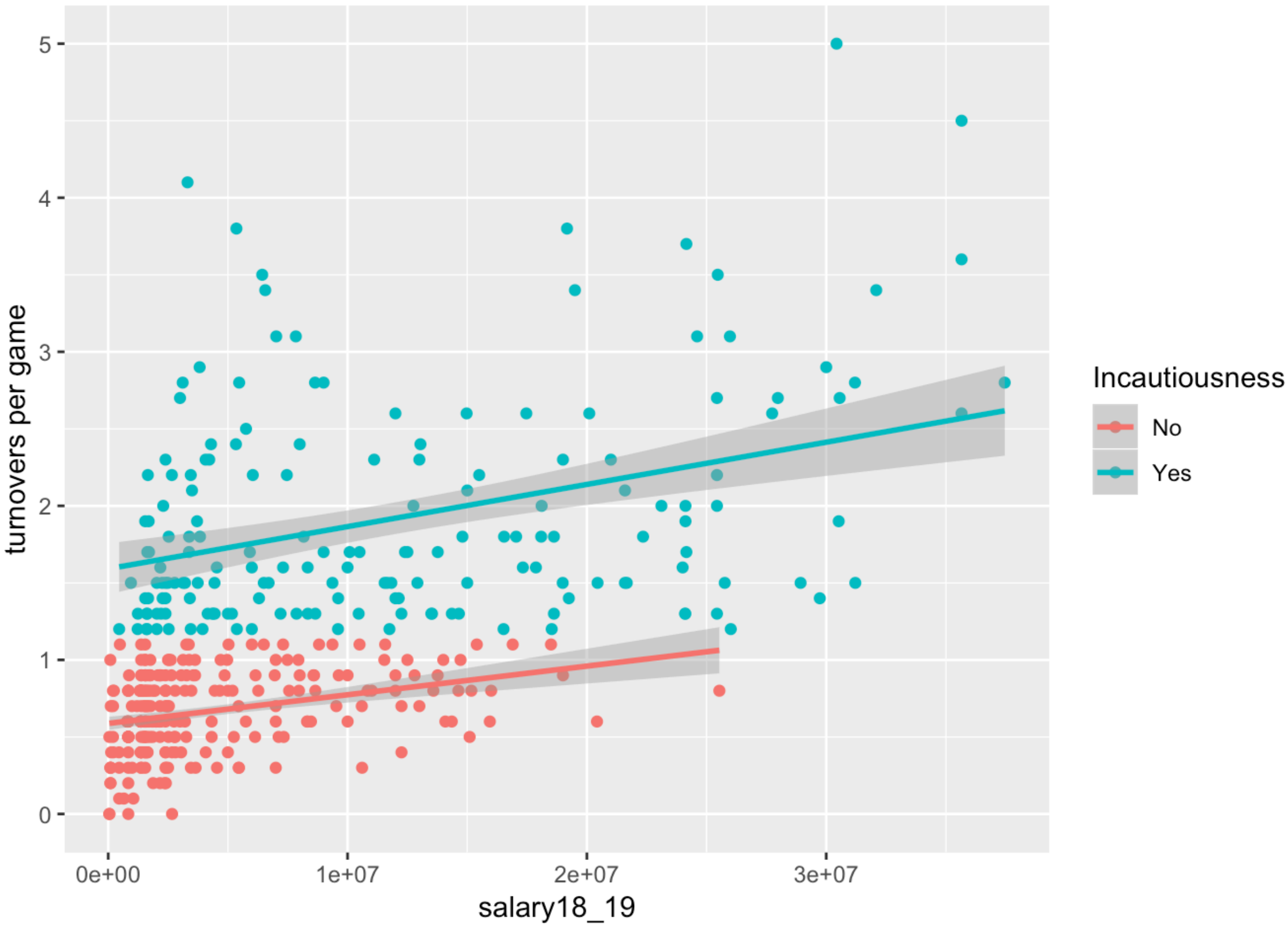
```
avg.minutes <- mean(regression$MP)
avg.turnover <- mean(regression$TOV)
regression$Importance<- as.factor(ifelse(regression$MP >= avg.minutes, "Yes", "No"))
regression$Incautiousness <- as.factor(ifelse(regression$TOV >= avg.turnover, "Yes",
"No"))
head(regression)
```

```
##   salary18_19         PTS          MP         TOV         AST         STL
## 1    21590909  1.12354560  1.5536855  1.2222684  0.90172692  0.1196179
## 2     1911960 -0.54308294 -0.8773917 -0.4188509 -0.20733649 -0.6151041
## 3     1378242 -0.85660712 -1.0518710 -0.9238106 -0.98368087 -0.8600115
## 4    28928710  0.72751506  0.9953519  0.4648288  1.17899277  0.6094326
## 5     6957105  0.03446161  0.9139283 -0.2926109 -0.42914917  0.3645252
## 6    11536515 -0.06454603  0.1229558 -0.1663710 -0.04097697 -0.1252894
##             TRB Importance Incautiousness
## 1  1.5363734745        Yes            Yes
## 2 -0.9949822302         No             No
## 3 -0.7459964232         No             No
## 4  1.2458900330        Yes            Yes
## 5  1.5778711090        Yes             No
## 6  0.0009609979        Yes             No
```

5.2 Prallel Slope Model

5.2.1 Incautiousness Comparision

```
regression %>%
  ggplot(aes(x = salary18_19, y = (TOV * var(player_season_tidy$TOV)^(1/2) + mean(pla
yer_season_tidy$TOV)), colour = Incautiousness)) +
  geom_point() +
  geom_smooth(method="lm")+
  ylab("turnovers per game")
```



It's true that players with higher salaries make more turnovers. But the tendency is weak.So in fact, turnovers don't have much impact on salaries.

```
lm(formula = salary18_19 ~ Importance * Incautiousness, data=regression)
```

```
##
## Call:
## lm(formula = salary18_19 ~ Importance * Incautiousness, data = regression)
##
## Coefficients:
##                    (Intercept)                      ImportanceYes
##                        3275995                            3754147
##              IncautiousnessYes   ImportanceYes:IncautiousnessYes
##                        3048670                            3017297
```

This shows that when a player is important, he is paid more with fewer turnovers. But the impact is limited.

5.3 Stepwise Regression

```r
# Do Stepwise Regression
stepdata <- stats_scale %>%  dplyr::select(Age:salary18_19)
full.model <- lm(salary18_19~., data =stepdata)
step.model <- stepAIC(full.model, direction = "both",
                      trace = FALSE)
summary(step.model)
```

```
##
## Call:
## lm(formula = salary18_19 ~ Age + `2P%` + `FT%` + TRB + AST +
##     STL + PF + PTS, data = stepdata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -16726226  -3415620   -329804   2885318  19994129
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7029186     265808  26.445  < 2e-16 ***
## Age          2699715     263103  10.261  < 2e-16 ***
## `2P%`        -498000     300816  -1.655  0.09859 .
## `FT%`        -598141     317671  -1.883  0.06042 .
## TRB          1906249     421719   4.520 8.09e-06 ***
## AST           781799     400336   1.953  0.05151 .
## STL          1151355     403226   2.855  0.00452 **
## PF           -889339     396582  -2.243  0.02546 *
## PTS          3003649     473869   6.339 6.09e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5423000 on 412 degrees of freedom
## Multiple R-squared:  0.5536, Adjusted R-squared:  0.545
## F-statistic: 63.88 on 8 and 412 DF,  p-value: < 2.2e-16
```

```r
# Set seed for reproducibility
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(salary18_19 ~., data = regression,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:9),
                    trControl = train.control
                    )
step.model$results
```

```
##   nvmax    RMSE  Rsquared     MAE   RMSESD RsquaredSD    MAESD
## 1     1 6477031 0.3656265 4841275 740066.8  0.1181143 489025.3
## 2     2 6394271 0.3803728 4800648 640458.4  0.1150337 384294.5
## 3     3 6113242 0.4295432 4634334 621821.6  0.1299230 442098.5
## 4     4 6082951 0.4340008 4636211 662992.5  0.1335307 493985.8
## 5     5 6157695 0.4198971 4707174 644375.1  0.1258431 498597.2
## 6     6 6142967 0.4211164 4689635 668283.2  0.1298739 503369.9
## 7     7 6118851 0.4262623 4685282 678915.2  0.1321880 539508.8
## 8     8 6116129 0.4267740 4683816 673996.1  0.1312854 541582.6
## 9     9 6116129 0.4267740 4683816 673996.1  0.1312854 541582.6
```

From the result we can see that three-variable model's RMSE is the smallest.

```
summary(step.model$finalModel)
```

```
## Subset selection object
## 8 Variables  (and intercept)
##                    Forced in Forced out
## PTS                    FALSE      FALSE
## MP                     FALSE      FALSE
## TOV                    FALSE      FALSE
## AST                    FALSE      FALSE
## STL                    FALSE      FALSE
## TRB                    FALSE      FALSE
## ImportanceYes          FALSE      FALSE
## IncautiousnessYes      FALSE      FALSE
## 1 subsets of each size up to 4
## Selection Algorithm: backward
##          PTS MP  TOV AST STL TRB ImportanceYes IncautiousnessYes
## 1  ( 1 ) "*" " " " " " " " " " " " "           " "
## 2  ( 1 ) "*" " " " " "*" " " " " " "           " "
## 3  ( 1 ) "*" " " " " "*" " " "*" " "           " "
## 4  ( 1 ) "*" " " "*" "*" " " "*" " "           " "
```

```
coef(step.model$finalModel, 3)
```

```
## (Intercept)         PTS         AST         TRB
##     7095995     2678780     1764352     1642389
```

The best model is salary18_19 ~ PTS + AST + TRB

6 Conclusion

6.1 What i want to predict

As Lebron James fan, I am concerned about the new contract for the Lakers who may stay next season. Let's find them first.

```
salaries_v4 %>% filter(Tm == "LAL" & `2019-20` == "")%>%
  dplyr::select(Player)
```

```
## # A tibble: 8 x 1
## # Groups:   Player [8]
##   Player
##   <chr>
## 1 Kentavious Caldwell-Pope
## 2 Rajon Rondo
## 3 Mike Muscala
## 4 Lance Stephenson
## 5 Reggie Bullock
## 6 JaVale McGee
## 7 Andre Ingram
## 8 Scott Machado
```

```
Pope <- stats_scale %>% filter(Row.names == "Kentavious Caldwell-Pope")
Bullock <- stats_scale %>% filter(Row.names == "Reggie Bullock")
Stephenson <- stats_scale %>% filter(Row.names == "Lance Stephenson")
Pope
```

```
##                  Row.names        Age        MP       2P%       3P%
## 1 Kentavious Caldwell-Pope -0.2348656 0.5068101 0.4681242 0.3189064
##         FT%        TRB        AST       STL        BLK        TOV
## 1 0.9065815 -0.3310201 -0.4291492 0.6094326 -0.4974759 -0.4188509
##           PF       PTS salary18_19
## 1 -0.1395071 0.3644871     1.2e+07
```

6.2 Analysis conclusion

```
salary_prediction <- function(m, point,assists,rebounds){
  pre_new <- predict(m, data.frame(PTS = point, AST= assists, TRB= rebounds))
  msg <- paste( "Expected Salary: $", format(round(pre_new), big.mark = ","), sep = "
")
  print(msg)
}
model <- lm(salary18_19~ PTS + AST + TRB, data =regression)
salary_prediction(model,Pope$PTS,Pope$AST,Pope$TRB)
```

```
## [1] "Expected Salary: $6,771,542"
```

```
salary_prediction(model,Bullock$PTS,Bullock$AST,Bullock$TRB)
```

```
## [1] "Expected Salary: $7,275,901"
```

```
salary_prediction(model,Stephenson$PTS,Stephenson$AST,Stephenson$TRB)
```

```
## [1] "Expected Salary: $5,902,181"
```

So Salaries for Pope, Bullock and Stephenson for next season are $6,771,542, $7,275,901 and $5,902,181