Applied Data Analysis in Sport PG (10157), Semester 1 2022: Reproducible Data Analysis Project

Sreekar U3212459

1. Introduction

Every team owner wish to have top players only. But, its also a business and we need to "purchase!" only value for money players for maximum utilization of money. The report finds top 5 value for money players with the help of linear regression analysis performed with help of R packages(1).

The top 5 players are:

```
##   POSITION             PLAYER SALARYmillions
## 1       PG        Ben Simmons           6.43
## 2       SG       Devin Booker           3.31
## 3       SF     Brandon Ingram           5.75
## 4       PF      Julius Randle           8.64
## 5        C Karl-Anthony Towns           7.83
## 6                       TOTAL          31.96
```

a) Description of scenario

Chicago Bulls is a Basketball team which participates in the NBA (National Basketball Association) seasons(2). General manager of Chicago Bulls has approached me, i.e. the author of this project; to find 5 best players for 5 positions, one player for each position for the next NBA season 2019-20.

b) Background information

It is better to get some idea of the basketball game. Normally, a team has 5 players and these players are assigned to 5 positions. The basketball positions are assigned a number as under:

1. Point Guard [PG]

Main task of points guard is to score points. Besides that he should help in assists and limiting turnovers.

2. Shooting Guard [SG]

Main task of shooting guard is to score points. Besides that he should help in assists and limiting turnovers.

3. Small Forward [SF]

Main task of small forward is to score points. Besides that he should help in assists and limiting turnovers.

4. Power Forward [PF]

Main task of power forward is to score points. Besides that he should help in rebounds and limiting turnovers.

5. Center [C]

Rebounds, Blocking and limiting turnover are the key metrics for center.

c) Aim of project

The aim of this project is to find out the 5 best players, one for each position for Chicago Bulls for the next session 2019-20. It may be noted that we are not going to just pick the top 5 players. We need to consider the budget and need to develop a model which identifies the undervalued players.

d) Justification and importance

The budget of Chicago Bulls for player contracts next season is $118 million. If this project is able to provide an accurate prediction of the 5 player who are value for money, then the Chicago Bulls will be able to get good players in reasonable amount. Besides that the recommendation and selection of the players shall be free from bias as it shall be data based. Linear regression shall help in finding the undervalued players. We need to find the players with the ability to score high points but getting a lower to moderate salary. Thus, at first we shall develop a model to predict the players ability to score points and thereafter compare this with players salary and select the value for money players of different positions with help of this framework. As stated earlier that the top players are very costly, hence it makes sense to develop a metric to value players based on that metric to their relative contribution to score points. The confidence interval limits are positive. i.e. it is not that one limit is negative and one limit is positive.

2. Reading and cleaning the raw data

Date Description

The data set consists of 5 csv files. The data description is can be seen by clicking below link: https://github.com/Sreekardeshamoni/Assessment-4/blob/main/Data%20Description.pdf

Loading tidyverse library and reading the data

The csv files are saved in data frames tibbles of readr package.

```
library(tidyverse)
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'pillar'
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'tibble'
```

```
## Warning: replacing previous import 'lifecycle::last_warnings' by
## 'rlang::last_warnings' when loading 'hms'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)
player_stat = read_csv("data/raw/2018-19_nba_player-statistics.csv")
```

```
## Rows: 708 Columns: 29
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): player_name, Pos, Tm
## dbl (26): Age, G, GS, MP, FG, FGA, FG%, 3P, 3PA, 3P%, 2P, 2PA, 2P%, eFG%, FT...
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
player_salary = read_csv("data/raw/2018-19_nba_player-salaries.csv")
```

```
## Rows: 576 Columns: 3-- Column specification ------------------------------------------------------------
## Delimiter: ","
## chr (1): player_name
## dbl (2): player_id, salary
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
team_payroll = read_csv("data/raw/2019-20_nba_team-payroll.csv")
```

```
## Rows: 30 Columns: 3-- Column specification ------------------------------------------------------------
## Delimiter: ","
## chr (2): team, salary
## dbl (1): team_id
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
team_stat_1 = read_csv("data/raw/2018-19_nba_team-statistics_1.csv")
```

```
## New names:
## * `` -> ...23
## * `` -> ...24
## * `` -> ...25
## Rows: 30 Columns: 25-- Column specification ------------------------------------------------------------
## Delimiter: ","
## chr  (1): Team
## dbl (21): Rk, Age, W, L, PW, PL, MOV, SOS, SRS, ORtg, DRtg, NRtg, Pace, FTr,...
## lgl  (3): ...23, ...24, ...25
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
team_stat_2 = read_csv("data/raw/2018-19_nba_team-statistics_2.csv")
```

```
## Rows: 30 Columns: 25-- Column specification ------------------------------------------------------------
## Delimiter: ","
## chr  (1): Team
## dbl (24): Rk, G, MP, FG, FGA, FG%, 3P, 3PA, 3P%, 2P, 2PA, 2P%, FT, FTA, FT%,...
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Now, its time to have a look at data structure.

```
str(player_stat)
```

```
## spec_tbl_df [708 x 29] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ player_name: chr [1:708] "Alex Abrines" "Quincy Acy" "Jaylen Adams" "Steven Adams" ...
##  $ Pos        : chr [1:708] "SG" "PF" "PG" "C" ...
##  $ Age        : num [1:708] 25 28 22 25 21 21 25 33 21 23 ...
##  $ Tm         : chr [1:708] "OKC" "PHO" "ATL" "OKC" ...
##  $ G          : num [1:708] 31 10 34 80 82 19 7 81 10 38 ...
##  $ GS         : num [1:708] 2 0 1 80 28 3 0 81 1 2 ...
##  $ MP         : num [1:708] 588 123 428 2669 1913 ...
```

```
##  $ FG         : num [1:708] 56 4 38 481 280 11 3 684 13 67 ...
##  $ FGA        : num [1:708] 157 18 110 809 486 ...
##  $ FG%        : num [1:708] 0.357 0.222 0.345 0.595 0.576 0.306 0.3 0.519 0.333 0.376 ...
##  $ 3P         : num [1:708] 41 2 25 0 3 6 0 10 3 32 ...
##  $ 3PA        : num [1:708] 127 15 74 2 15 23 4 42 12 99 ...
##  $ 3P%        : num [1:708] 0.323 0.133 0.338 0 0.2 0.261 0 0.238 0.25 0.323 ...
##  $ 2P         : num [1:708] 15 2 13 481 277 5 3 674 10 35 ...
##  $ 2PA        : num [1:708] 30 3 36 807 471 ...
##  $ 2P%        : num [1:708] 0.5 0.667 0.361 0.596 0.588 0.385 0.5 0.528 0.37 0.443 ...
##  $ eFG%       : num [1:708] 0.487 0.278 0.459 0.595 0.579 0.389 0.3 0.522 0.372 0.466 ...
##  $ FT         : num [1:708] 12 7 7 146 166 4 1 349 8 45 ...
##  $ FTA        : num [1:708] 13 10 9 292 226 4 2 412 12 60 ...
##  $ FT%        : num [1:708] 0.923 0.7 0.778 0.5 0.735 1 0.5 0.847 0.667 0.75 ...
##  $ ORB        : num [1:708] 5 3 11 391 165 3 1 251 11 3 ...
##  $ DRB        : num [1:708] 43 22 49 369 432 16 3 493 15 20 ...
##  $ TRB        : num [1:708] 48 25 60 760 597 19 4 744 26 23 ...
##  $ AST        : num [1:708] 20 8 65 124 184 5 6 194 13 25 ...
##  $ STL        : num [1:708] 17 1 14 117 71 1 2 43 1 6 ...
##  $ BLK        : num [1:708] 6 4 5 76 65 4 0 107 0 6 ...
##  $ TOV        : num [1:708] 14 4 28 135 121 6 2 144 8 33 ...
##  $ PF         : num [1:708] 53 24 45 204 203 13 4 179 7 47 ...
##  $ PTS        : num [1:708] 165 17 108 1108 729 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   player_name = col_character(),
##   ..   Pos = col_character(),
##   ..   Age = col_double(),
##   ..   Tm = col_character(),
##   ..   G = col_double(),
##   ..   GS = col_double(),
##   ..   MP = col_double(),
##   ..   FG = col_double(),
##   ..   FGA = col_double(),
##   ..   `FG%` = col_double(),
##   ..   `3P` = col_double(),
##   ..   `3PA` = col_double(),
##   ..   `3P%` = col_double(),
##   ..   `2P` = col_double(),
##   ..   `2PA` = col_double(),
##   ..   `2P%` = col_double(),
##   ..   `eFG%` = col_double(),
##   ..   FT = col_double(),
##   ..   FTA = col_double(),
##   ..   `FT%` = col_double(),
##   ..   ORB = col_double(),
##   ..   DRB = col_double(),
##   ..   TRB = col_double(),
##   ..   AST = col_double(),
##   ..   STL = col_double(),
##   ..   BLK = col_double(),
##   ..   TOV = col_double(),
##   ..   PF = col_double(),
##   ..   PTS = col_double()
##   .. )
```

```
##  - attr(*, "problems")=<externalptr>
```

```
str(player_salary)
```

```
## spec_tbl_df [576 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ player_id  : num [1:576] 1 2 3 4 5 6 7 8 9 10 ...
##  $ player_name: chr [1:576] "Alex Abrines" "Quincy Acy" "Steven Adams" "Jaylen Adams" ...
##  $ salary     : num [1:576] 3667645 213948 24157304 236854 2955840 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   player_id = col_double(),
##   ..   player_name = col_character(),
##   ..   salary = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(team_payroll)
```

```
## spec_tbl_df [30 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ team_id: num [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##  $ team   : chr [1:30] "Miami" "Golden State" "Oklahoma City" "Toronto" ...
##  $ salary : chr [1:30] "$153,171,497" "$146,291,276" "$144,916,427" "$137,793,831" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   team_id = col_double(),
##   ..   team = col_character(),
##   ..   salary = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(team_stat_1)
```

```
## spec_tbl_df [30 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Rk    : num [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Team  : chr [1:30] "Milwaukee Bucks" "Golden State Warriors" "Toronto Raptors" "Utah Jazz" ...
##  $ Age   : num [1:30] 26.9 28.4 27.3 27.3 29.2 26.2 24.9 25.7 25.7 27 ...
##  $ W     : num [1:30] 60 57 58 50 53 53 54 49 49 48 ...
##  $ L     : num [1:30] 22 25 24 32 29 29 28 33 33 34 ...
##  $ PW    : num [1:30] 61 56 56 54 53 51 51 52 50 50 ...
##  $ PL    : num [1:30] 21 26 26 28 29 31 31 30 32 32 ...
##  $ MOV   : num [1:30] 8.87 6.46 6.09 5.26 4.77 4.2 3.95 4.44 3.4 3.33 ...
##  $ SOS   : num [1:30] -0.82 -0.04 -0.6 0.03 0.19 0.24 0.24 -0.54 0.15 -0.57 ...
##  $ SRS   : num [1:30] 8.04 6.42 5.49 5.28 4.96 4.43 4.19 3.9 3.56 2.76 ...
##  $ ORtg  : num [1:30] 114 116 113 111 116 ...
##  $ DRtg  : num [1:30] 105 110 107 106 111 ...
##  $ NRtg  : num [1:30] 8.6 6.4 6 5.2 4.8 4.2 4.1 4.4 3.3 3.4 ...
##  $ Pace  : num [1:30] 103.3 100.9 100.2 100.3 97.9 ...
##  $ FTr   : num [1:30] 0.255 0.227 0.247 0.295 0.279 0.258 0.232 0.215 0.266 0.242 ...
##  $ 3PAr  : num [1:30] 0.419 0.384 0.379 0.394 0.519 0.339 0.348 0.381 0.347 0.292 ...
##  $ TS%   : num [1:30] 0.583 0.596 0.579 0.572 0.581 0.568 0.558 0.567 0.545 0.561 ...
##  $ eFG%  : num [1:30] 0.55 0.565 0.543 0.538 0.542 0.528 0.527 0.534 0.514 0.53 ...
##  $ TOV%  : num [1:30] 12 12.6 12.4 13.4 12 12.1 11.9 11.5 11.7 12.4 ...
```

```
##  $ ORB%  : num [1:30] 20.8 22.5 21.9 22.9 22.8 26.6 26.6 21.6 26 21.9 ...
##  $ FT/FGA: num [1:30] 0.197 0.182 0.198 0.217 0.221 0.21 0.175 0.173 0.19 0.182 ...
##  $ DRB%  : num [1:30] 80.3 77.1 77.1 80.3 74.4 77.9 78 77 78.2 76.2 ...
##  $ ...23 : logi [1:30] NA NA NA NA NA NA ...
##  $ ...24 : logi [1:30] NA NA NA NA NA NA ...
##  $ ...25 : logi [1:30] NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Rk = col_double(),
##   ..   Team = col_character(),
##   ..   Age = col_double(),
##   ..   W = col_double(),
##   ..   L = col_double(),
##   ..   PW = col_double(),
##   ..   PL = col_double(),
##   ..   MOV = col_double(),
##   ..   SOS = col_double(),
##   ..   SRS = col_double(),
##   ..   ORtg = col_double(),
##   ..   DRtg = col_double(),
##   ..   NRtg = col_double(),
##   ..   Pace = col_double(),
##   ..   FTr = col_double(),
##   ..   `3PAr` = col_double(),
##   ..   `TS%` = col_double(),
##   ..   `eFG%` = col_double(),
##   ..   `TOV%` = col_double(),
##   ..   `ORB%` = col_double(),
##   ..   `FT/FGA` = col_double(),
##   ..   `DRB%` = col_double(),
##   ..   ...23 = col_logical(),
##   ..   ...24 = col_logical(),
##   ..   ...25 = col_logical()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(team_stat_2)
```

```
## spec_tbl_df [30 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Rk  : num [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Team: chr [1:30] "Milwaukee Bucks" "Golden State Warriors" "New Orleans Pelicans" "Philadelphia 7
##  $ G   : num [1:30] 82 82 82 82 82 82 82 82 82 82 ...
##  $ MP  : num [1:30] 19780 19805 19755 19805 19830 ...
##  $ FG  : num [1:30] 3555 3612 3581 3407 3384 ...
##  $ FGA : num [1:30] 7471 7361 7563 7233 7178 ...
##  $ FG% : num [1:30] 0.476 0.491 0.473 0.471 0.471 0.467 0.454 0.474 0.464 0.468 ...
##  $ 3P  : num [1:30] 1105 1087 842 889 821 ...
##  $ 3PA : num [1:30] 3134 2824 2449 2474 2118 ...
##  $ 3P% : num [1:30] 0.353 0.385 0.344 0.359 0.388 0.359 0.348 0.366 0.378 0.341 ...
##  $ 2P  : num [1:30] 2450 2525 2739 2518 2563 ...
##  $ 2PA : num [1:30] 4337 4537 5114 4759 5060 ...
##  $ 2P% : num [1:30] 0.565 0.557 0.536 0.529 0.507 0.523 0.51 0.539 0.504 0.543 ...
##  $ FT  : num [1:30] 1471 1339 1462 1742 1853 ...
##  $ FTA : num [1:30] 1904 1672 1921 2258 2340 ...
```

```
##  $ FT% : num [1:30] 0.773 0.801 0.761 0.771 0.792 0.814 0.713 0.804 0.726 0.768 ...
##  $ ORB : num [1:30] 762 797 909 892 796 ...
##  $ DRB : num [1:30] 3316 2990 2969 3025 2936 ...
##  $ TRB : num [1:30] 4078 3787 3878 3917 3732 ...
##  $ AST : num [1:30] 2136 2413 2216 2207 1970 ...
##  $ STL : num [1:30] 615 625 610 606 561 546 766 680 679 683 ...
##  $ BLK : num [1:30] 486 525 441 432 385 413 425 437 363 379 ...
##  $ TOV : num [1:30] 1137 1169 1215 1223 1193 ...
##  $ PF  : num [1:30] 1608 1757 1732 1745 1913 ...
##  $ PTS : num [1:30] 9686 9650 9466 9445 9442 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Rk = col_double(),
##   ..   Team = col_character(),
##   ..   G = col_double(),
##   ..   MP = col_double(),
##   ..   FG = col_double(),
##   ..   FGA = col_double(),
##   ..   `FG%` = col_double(),
##   ..   `3P` = col_double(),
##   ..   `3PA` = col_double(),
##   ..   `3P%` = col_double(),
##   ..   `2P` = col_double(),
##   ..   `2PA` = col_double(),
##   ..   `2P%` = col_double(),
##   ..   FT = col_double(),
##   ..   FTA = col_double(),
##   ..   `FT%` = col_double(),
##   ..   ORB = col_double(),
##   ..   DRB = col_double(),
##   ..   TRB = col_double(),
##   ..   AST = col_double(),
##   ..   STL = col_double(),
##   ..   BLK = col_double(),
##   ..   TOV = col_double(),
##   ..   PF = col_double(),
##   ..   PTS = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

R can manage the "illegal" variable names by surrounding the variable name by ", however, as a best practice it makes sense to rename the variables.

```
player_stat <-rename(player_stat,
                    FGp = `FG%`,
                    x3P = `3P`,
                    x3PA = `3PA`,
                    x3Pp = `3P%`,
                    x2P = `2P`,
                    x2PA = `2PA`,
                    x2Pp = `2P%`,
                    eFGp = `eFG%`,
                    FTp = `FT%`)
```

```r
team_stat_1 <-rename(team_stat_1,
                           x3PAr = `3PAr`,
                           TSp = `TS%`,
                           eFGp = `eFG%`,
                           TOVp = `TOV%`,
                           ORBp = `ORB%`,
                           FTpFGA = `FT/FGA`,
                           DRBp = `DRB%`)

team_stat_2 <-rename(team_stat_2,
                           FGp = `FG%`,
                           x3P = `3P`,
                           x3PA = `3PA`,
                           x3Pp = `3P%`,
                           x2P = `2P`,
                           x2PA = `2PA`,
                           x2Pp = `2P%`,
                           FTp = `FT%`)
```

3. Exploratory analysis

Checking missing values player__stat

At first need to check the missing values.

```r
sum(is.na(player_stat))
```

```
## [1] 117
```

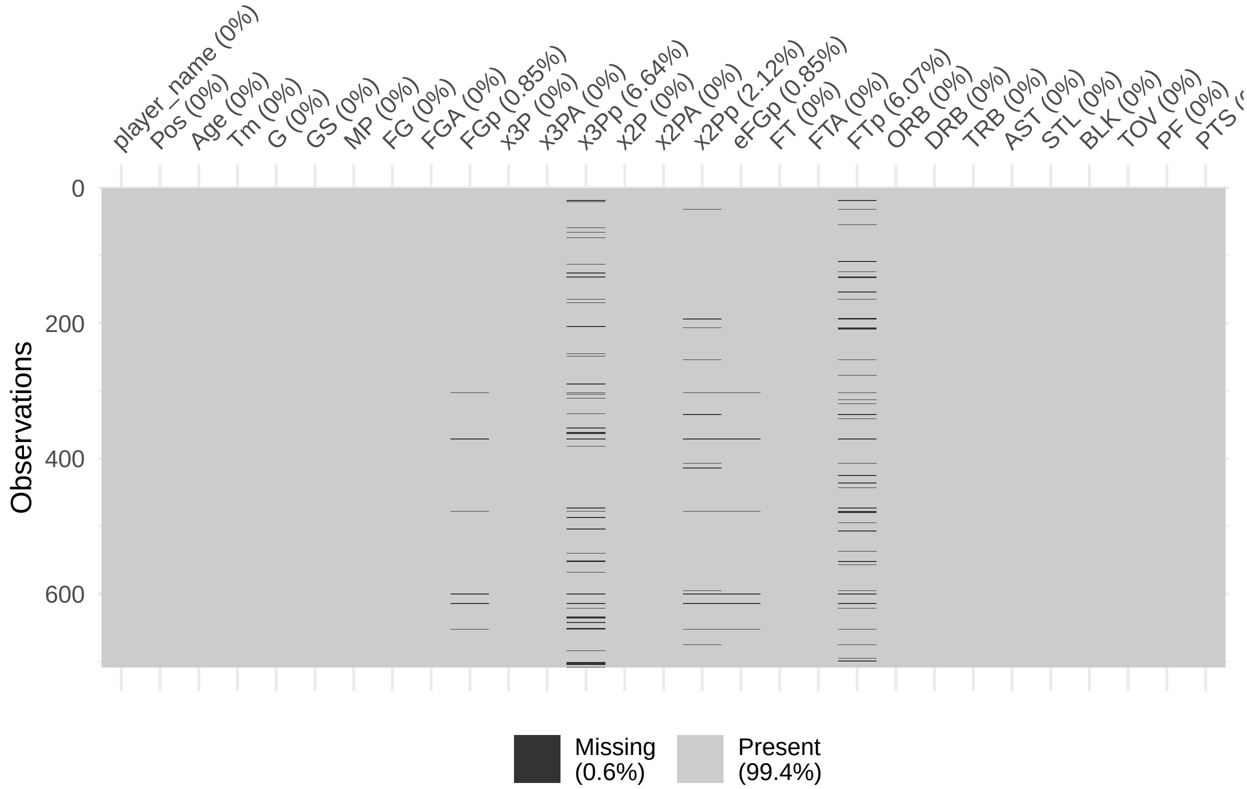
```r
which(is.na(player_stat), arr.ind = TRUE)
```

```
##        row col
##   [1,] 303  10
##   [2,] 371  10
##   [3,] 478  10
##   [4,] 600  10
##   [5,] 614  10
##   [6,] 652  10
##   [7,]  19  13
##   [8,]  21  13
##   [9,]  59  13
##  [10,]  66  13
##  [11,]  74  13
##  [12,] 113  13
##  [13,] 126  13
##  [14,] 132  13
##  [15,] 165  13
##  [16,] 170  13
##  [17,] 205  13
##  [18,] 245  13
##  [19,] 249  13
```

```
## [20,] 290 13
## [21,] 303 13
## [22,] 305 13
## [23,] 311 13
## [24,] 334 13
## [25,] 355 13
## [26,] 361 13
## [27,] 362 13
## [28,] 363 13
## [29,] 371 13
## [30,] 382 13
## [31,] 473 13
## [32,] 478 13
## [33,] 487 13
## [34,] 504 13
## [35,] 540 13
## [36,] 551 13
## [37,] 552 13
## [38,] 568 13
## [39,] 600 13
## [40,] 614 13
## [41,] 621 13
## [42,] 634 13
## [43,] 635 13
## [44,] 636 13
## [45,] 642 13
## [46,] 651 13
## [47,] 652 13
## [48,] 684 13
## [49,] 701 13
## [50,] 702 13
## [51,] 703 13
## [52,] 704 13
## [53,] 708 13
## [54,]  32 16
## [55,] 194 16
## [56,] 207 16
## [57,] 254 16
## [58,] 303 16
## [59,] 335 16
## [60,] 371 16
## [61,] 407 16
## [62,] 414 16
## [63,] 478 16
## [64,] 595 16
## [65,] 600 16
## [66,] 614 16
## [67,] 652 16
## [68,] 675 16
## [69,] 303 17
## [70,] 371 17
## [71,] 478 17
## [72,] 600 17
## [73,] 614 17
```

```
##  [74,] 652  17
##  [75,]  19  20
##  [76,]  32  20
##  [77,]  55  20
##  [78,] 109  20
##  [79,] 124  20
##  [80,] 132  20
##  [81,] 133  20
##  [82,] 154  20
##  [83,] 165  20
##  [84,] 193  20
##  [85,] 194  20
##  [86,] 207  20
##  [87,] 208  20
##  [88,] 209  20
##  [89,] 254  20
##  [90,] 277  20
##  [91,] 303  20
##  [92,] 313  20
##  [93,] 319  20
##  [94,] 335  20
##  [95,] 341  20
##  [96,] 371  20
##  [97,] 407  20
##  [98,] 425  20
##  [99,] 436  20
## [100,] 443  20
## [101,] 473  20
## [102,] 478  20
## [103,] 479  20
## [104,] 480  20
## [105,] 495  20
## [106,] 507  20
## [107,] 537  20
## [108,] 552  20
## [109,] 557  20
## [110,] 595  20
## [111,] 600  20
## [112,] 614  20
## [113,] 621  20
## [114,] 652  20
## [115,] 675  20
## [116,] 695  20
## [117,] 699  20
```

```
naniar::vis_miss(player_stat)
```

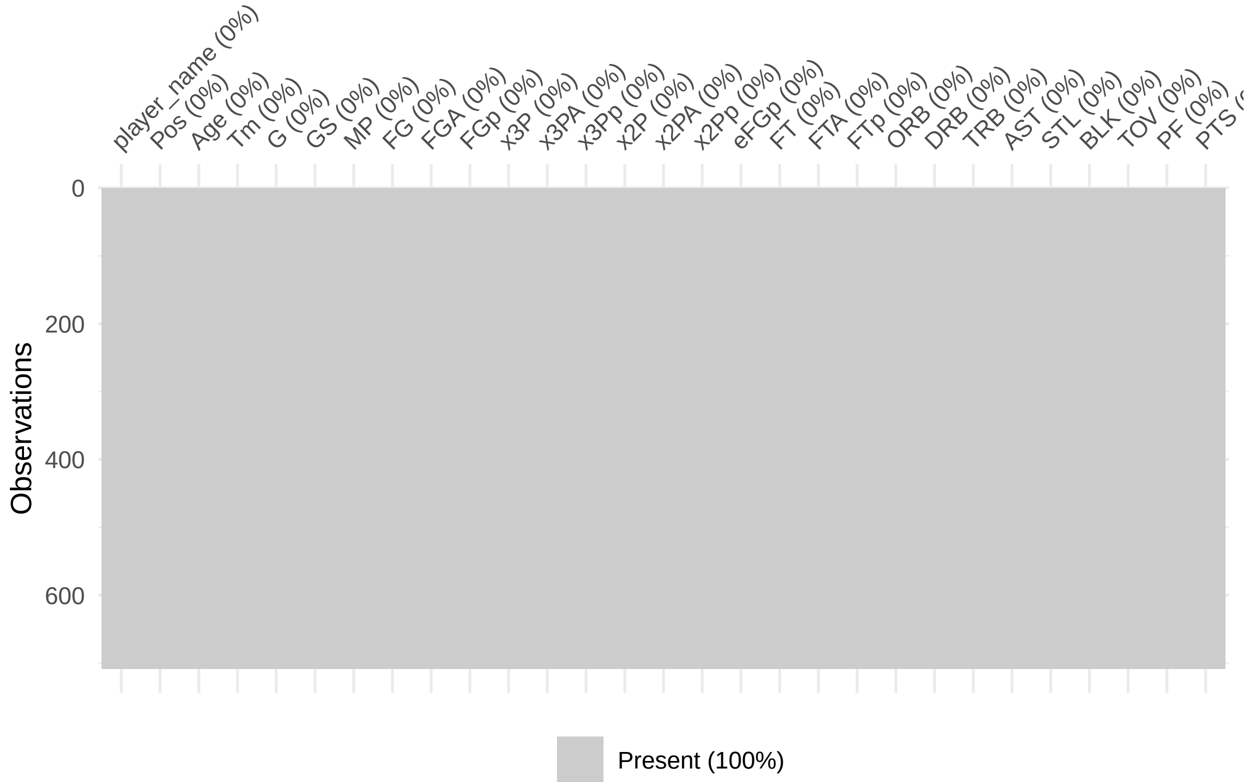


There are missing values for FGp, x3Pp, x2Pp, eFGp, and FTp. These are calculated field in the raw data itself, and where the denominator and/or numerator is zero, these fields becomes NA. However, it is understood that these are values where field goal, 2 Pointer and 3 Pointer was not done by the player. Hence, it is a good idea to replace these values by 0%.

```
player_stat <- replace_na(player_stat, list(FGp = 0,x3Pp = 0,x2Pp = 0, eFGp = 0, FTp = 0))
sum(is.na(player_stat))
```

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

```
naniar::vis_miss(player_stat)
```



If we check the player_name variable, we find that a total of 86 players have played for more than 1 team in the season. mydf1 = df1 %>% group_by(player_name) %>% summarise(num_of_teams =n()) %>% filter(num_of_teams>1)

Checking missing values player_salary

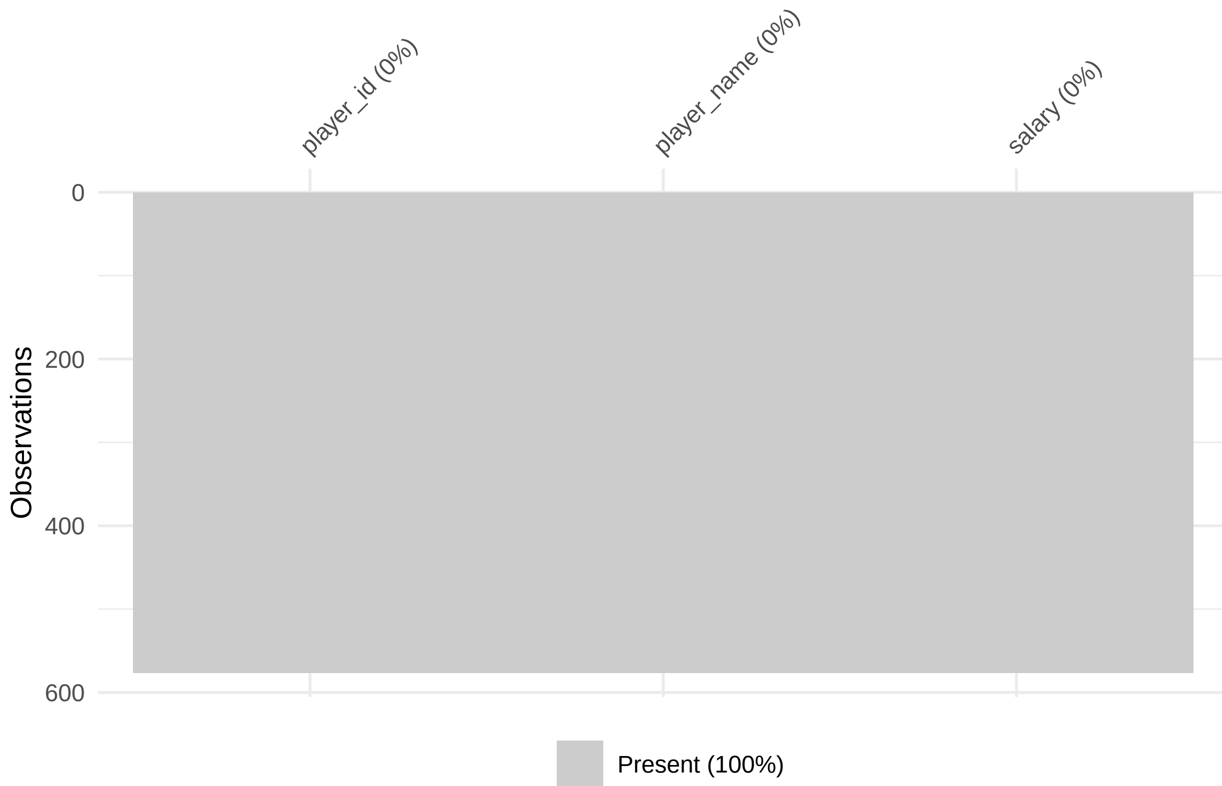
```
sum(is.na(player_salary))
```

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

```
which(is.na(player_salary), arr.ind = TRUE)
```

```
##      row col
```

```
naniar::vis_miss(player_salary)
```



Checking missing values team_payroll

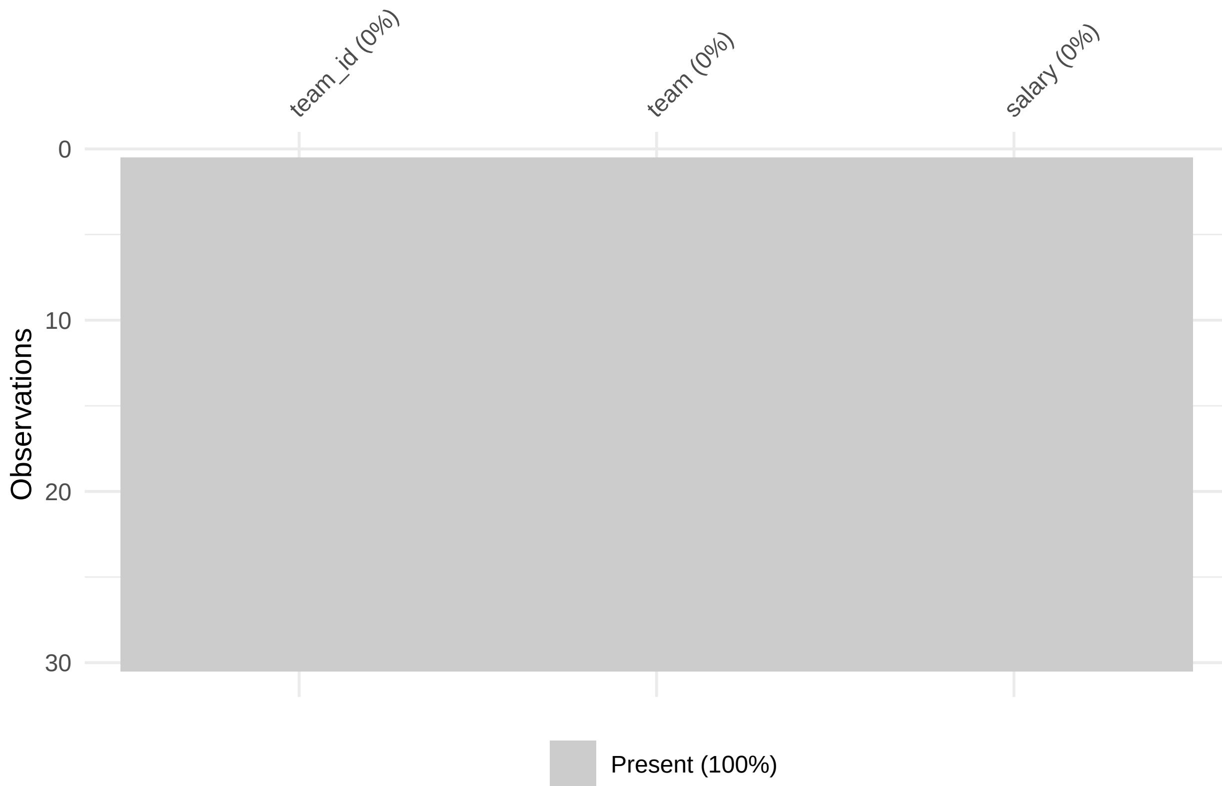
```r
sum(is.na(team_payroll))
```

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

```r
which(is.na(team_payroll), arr.ind = TRUE)
```

```
##      row col
```

```r
naniar::vis_miss(team_payroll)
```



Checking missing values team__stat__1

```
sum(is.na(team_stat_1))
```

```
## [1] 90
```

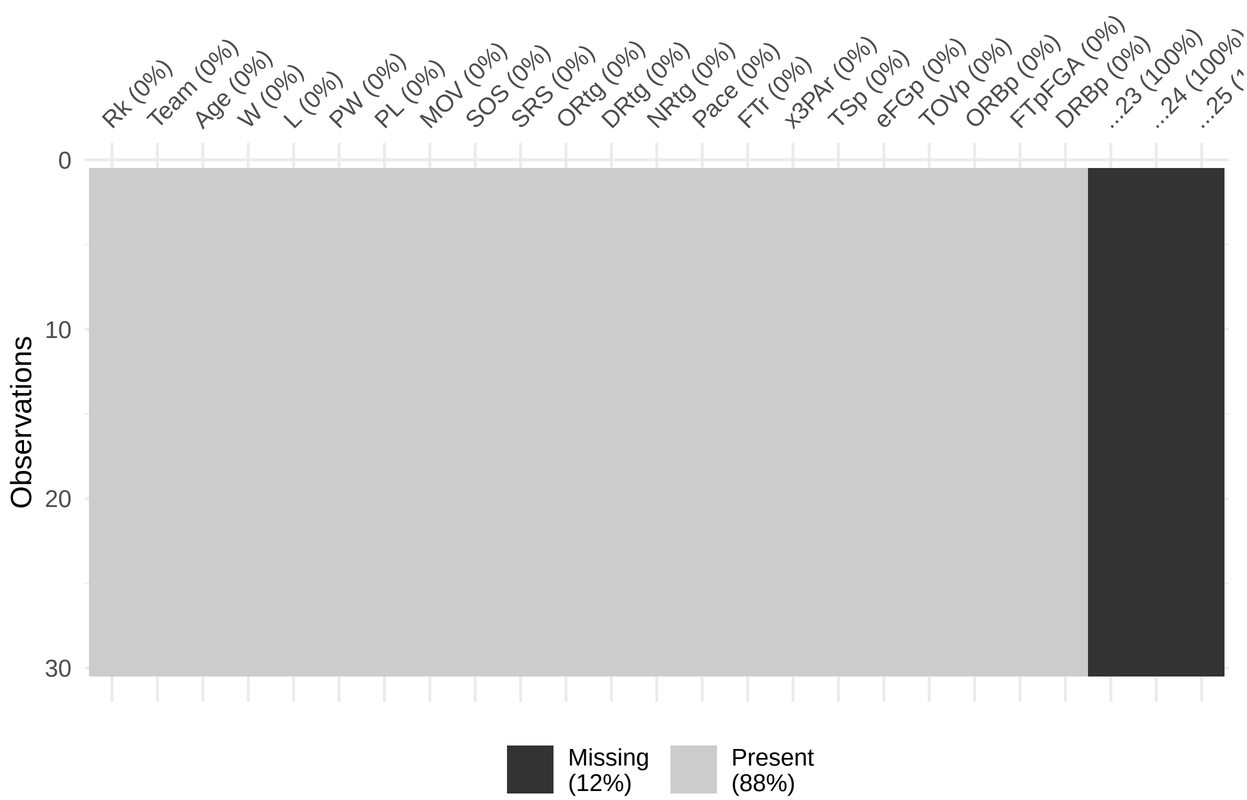
```
which(is.na(team_stat_1), arr.ind = TRUE)
```

```
##       row col
##  [1,]   1  23
##  [2,]   2  23
##  [3,]   3  23
##  [4,]   4  23
##  [5,]   5  23
##  [6,]   6  23
##  [7,]   7  23
##  [8,]   8  23
##  [9,]   9  23
## [10,]  10  23
## [11,]  11  23
## [12,]  12  23
## [13,]  13  23
## [14,]  14  23
```

```
## [15,]   15   23
## [16,]   16   23
## [17,]   17   23
## [18,]   18   23
## [19,]   19   23
## [20,]   20   23
## [21,]   21   23
## [22,]   22   23
## [23,]   23   23
## [24,]   24   23
## [25,]   25   23
## [26,]   26   23
## [27,]   27   23
## [28,]   28   23
## [29,]   29   23
## [30,]   30   23
## [31,]    1   24
## [32,]    2   24
## [33,]    3   24
## [34,]    4   24
## [35,]    5   24
## [36,]    6   24
## [37,]    7   24
## [38,]    8   24
## [39,]    9   24
## [40,]   10   24
## [41,]   11   24
## [42,]   12   24
## [43,]   13   24
## [44,]   14   24
## [45,]   15   24
## [46,]   16   24
## [47,]   17   24
## [48,]   18   24
## [49,]   19   24
## [50,]   20   24
## [51,]   21   24
## [52,]   22   24
## [53,]   23   24
## [54,]   24   24
## [55,]   25   24
## [56,]   26   24
## [57,]   27   24
## [58,]   28   24
## [59,]   29   24
## [60,]   30   24
## [61,]    1   25
## [62,]    2   25
## [63,]    3   25
## [64,]    4   25
## [65,]    5   25
## [66,]    6   25
## [67,]    7   25
## [68,]    8   25
```

```
## [69,]   9  25
## [70,]  10  25
## [71,]  11  25
## [72,]  12  25
## [73,]  13  25
## [74,]  14  25
## [75,]  15  25
## [76,]  16  25
## [77,]  17  25
## [78,]  18  25
## [79,]  19  25
## [80,]  20  25
## [81,]  21  25
## [82,]  22  25
## [83,]  23  25
## [84,]  24  25
## [85,]  25  25
## [86,]  26  25
## [87,]  27  25
## [88,]  28  25
## [89,]  29  25
## [90,]  30  25
```

```
naniar::vis_miss(team_stat_1)
```



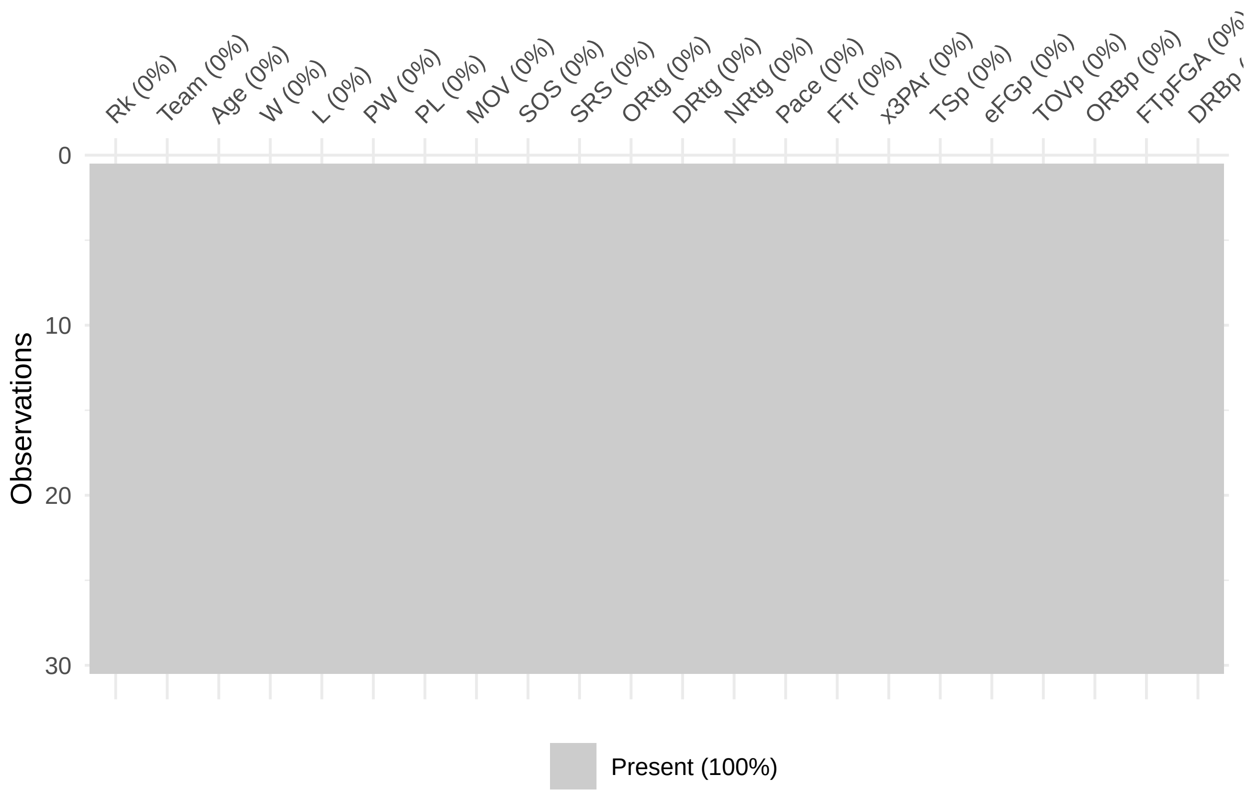
team_stat_1 has last 3 columns . . . 23, . . . 24, . . . 25 and these appears to be bogus columns and there is need to deleted these columns. The tidy team_stat_1 shall be as under:

```
team_stat_1 = select(team_stat_1,-c(...23:...25))
team_stat_1
```

```
## # A tibble: 30 x 22
##       Rk Team    Age     W     L    PW    PL   MOV   SOS   SRS  ORtg  DRtg  NRtg
##    <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1 Milw~  26.9    60    22    61    21  8.87 -0.82  8.04  114.  105.   8.6
##  2     2 Gold~  28.4    57    25    56    26  6.46 -0.04  6.42  116.  110.   6.4
##  3     3 Toro~  27.3    58    24    56    26  6.09 -0.6   5.49  113.  107.   6
##  4     4 Utah~  27.3    50    32    54    28  5.26  0.03  5.28  111.  106.   5.2
##  5     5 Hous~  29.2    53    29    53    29  4.77  0.19  4.96  116.  111.   4.8
##  6     6 Port~  26.2    53    29    51    31  4.2   0.24  4.43  115.  110.   4.2
##  7     7 Denv~  24.9    54    28    51    31  3.95  0.24  4.19  113   109.   4.1
##  8     8 Bost~  25.7    49    33    52    30  4.44 -0.54  3.9   112.  108.   4.4
##  9     9 Okla~  25.7    49    33    50    32  3.4   0.15  3.56  110.  107    3.3
## 10    10 Indi~  27      48    34    50    32  3.33 -0.57  2.76  110.  106.   3.4
## # ... with 20 more rows, and 9 more variables: Pace <dbl>, FTr <dbl>,
## #   x3PAr <dbl>, TSp <dbl>, eFGp <dbl>, TOVp <dbl>, ORBp <dbl>, FTpFGA <dbl>,
## #   DRBp <dbl>
```

Now, the bogus columns are gone. This can be crossed checked:

```
naniar::vis_miss(team_stat_1)
```



Checking missing values team_stat_2

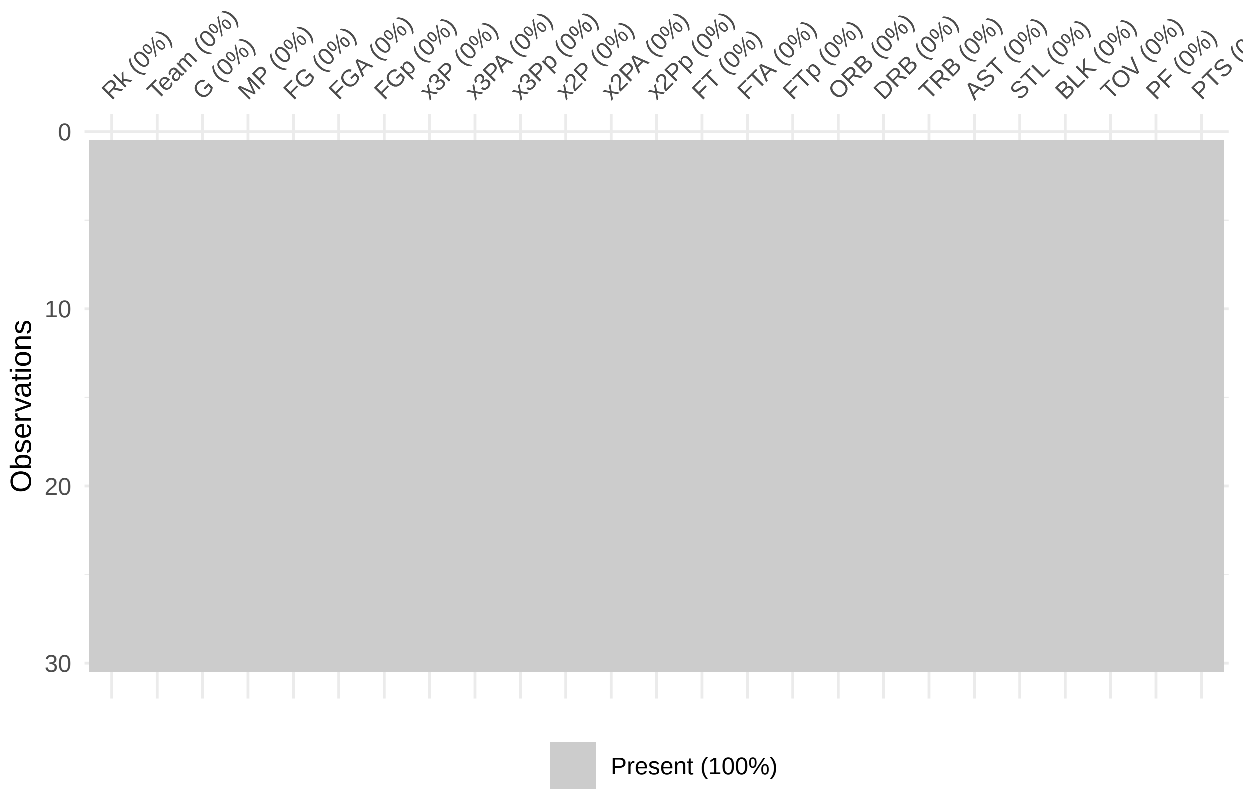
```
sum(is.na(team_stat_2))
```

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

```
which(is.na(team_stat_2), arr.ind = TRUE)
```

```
##      row col
```

```
naniar::vis_miss(team_stat_2)
```



We can combine team_stat_1 and team_stat_2.

```
team_stat <- left_join(x = team_stat_1[-1], y = team_stat_2[-1], by = "Team") #don't need cols 1 of Ran
```

Other aspects

Interesting fact is that there are 30 teams, but in the player statistics table, there are 31 teams.We find that extra team is TOT which represents the total of all the instances matrices,where a player played from 2 or more teams in the season. We need the total of performance done by such players for the teams they were playing. This can be get with row where team name is "TOT". Thus, fur such players,we can keep only the row having "TOT" as team.

```
player_stat <- player_stat %>% group_by(player_name) %>% add_tally() %>% filter(n==1 | n>1 & Tm == "TOT
```

Earlier, the player_stat has 730 rows, however, it has 530 rows.But one more issue is remaining.We find that there are a number of players who have played at different positions. However, we need to consider only a single position for these players. The best approach is to select the position for which the player has played most of the games. Luckily, we find that the raw data file, has already done this in some way. For example.. We just need to extract the characters before -, in the Pos column.

```
player_stat <- player_stat %>% separate(col = Pos, into = "Pos")
```

```
## Warning: Expected 1 pieces. Additional pieces discarded in 8 rows [33, 84, 101,
## 288, 324, 330, 448, 455].
```

Also, some of the players have played a lower number of games. In fact, there are 20 players who have played only 1 game. We need to create a sort of cutoff and minimum 10 games cutoff is a reasonable cut off.

```
player_stat <- player_stat %>% filter(G>=10)
```

There are 576 players in players salary table, however, there are only 530 unique players in the player statistics table. It means that some of the players did not played in NBA or the data for them is not available in player statistics. However, this fact is considered unimportant for the given project analysis. We can combine player_salary and player_stat

```
player_stat <- inner_join(x = player_stat, y = player_salary, by = "player_name")
```

4. Data modelling and results

Valuing Players

The main purpose of any basketball game is that the team wins that game. However, wining of games cannot alone a factor to select the players. In fact only team wins the matches and a player alone cannot have a credit to win the game and basket ball is a team game. Thus, we need find some metrics which includes several indirect measures to rate the players. A player alone may not win the game but it can score points. Points win the games and and after all players are recruited to score points. Now, we will create a metric to value the players based upon certain factors/variables. This metric can be called exp_PTS_per_game. We shall then identify the undervalue players with the help of this metric(3).

In order to predict the exp_PTS_per_game, we need to find some key variables for our analysis(4). There after we will perform a multiple linear regression model.

Some of the metrics to be used are below:

TRB Total Rebounds

AST Assists

STL Steals

BLK Blocks

TOV Turnovers

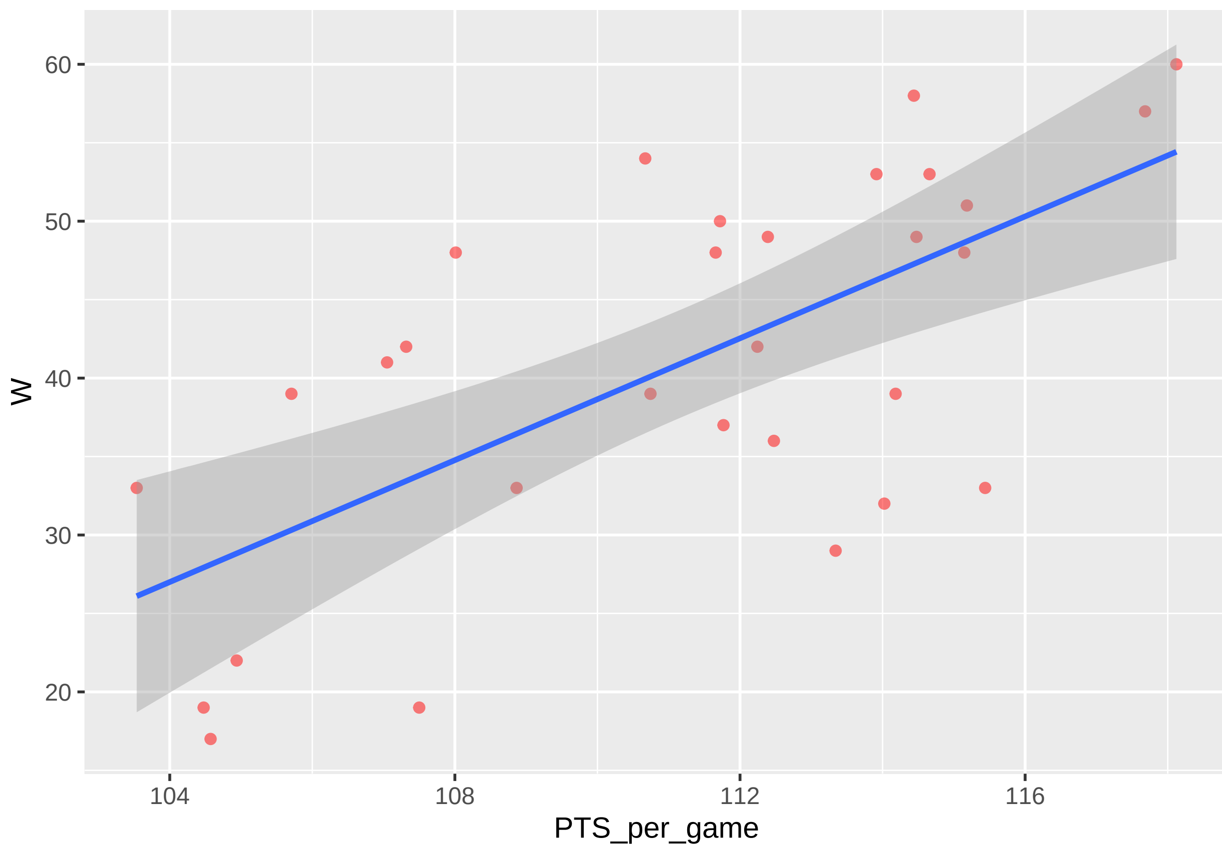
However, as number of games played by players are different, it makes sense to normalize these metrics per game.

```
normalised_team_stat <- team_stat %>% mutate(
  PTS_per_game = PTS / G,
  TRB_per_game = TRB / G,
  AST_per_game = AST / G,
  STL_per_game = STL / G,
  BLK_per_game = BLK / G,
  TOV_per_game = TOV / G)
```

Checking that PTS_per_game is correlated with Wins or not…

```
ggplot(normalised_team_stat, aes(x = PTS_per_game, y = W))+
geom_point(alpha = 0.5, colour = "red")+
geom_smooth(method = "lm")
```

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




```
#checking correlation between Win and Points_per_game
cor(x = normalised_team_stat$PTS_per_game, y = normalised_team_stat$W, method = "pearson")
```

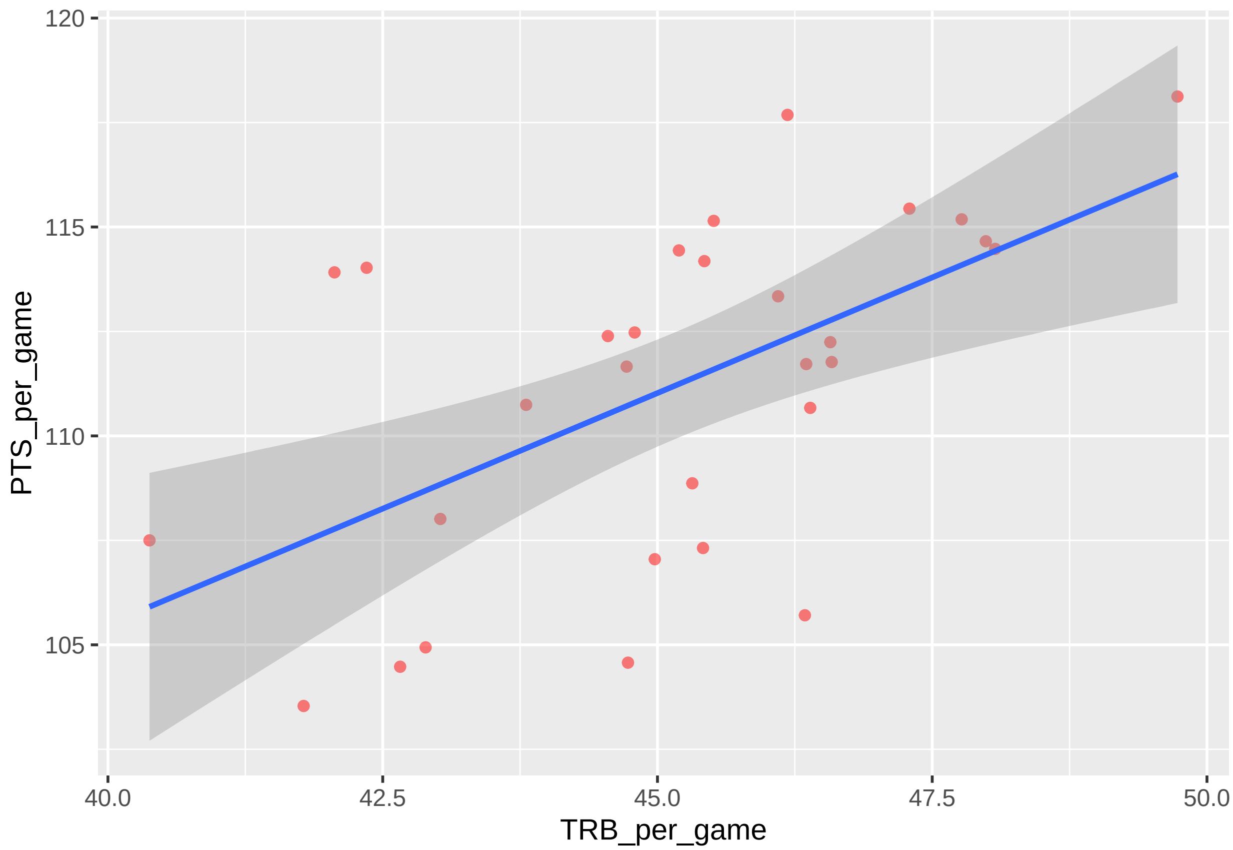
```
## [1] 0.6606001
```

Thus, we can see that Points_per_game has a positive correlation with Win.

Before developing a multiple regression model for PTS_per_game we should see confirm that dependent variable PTS_per_game and independent variables are related.

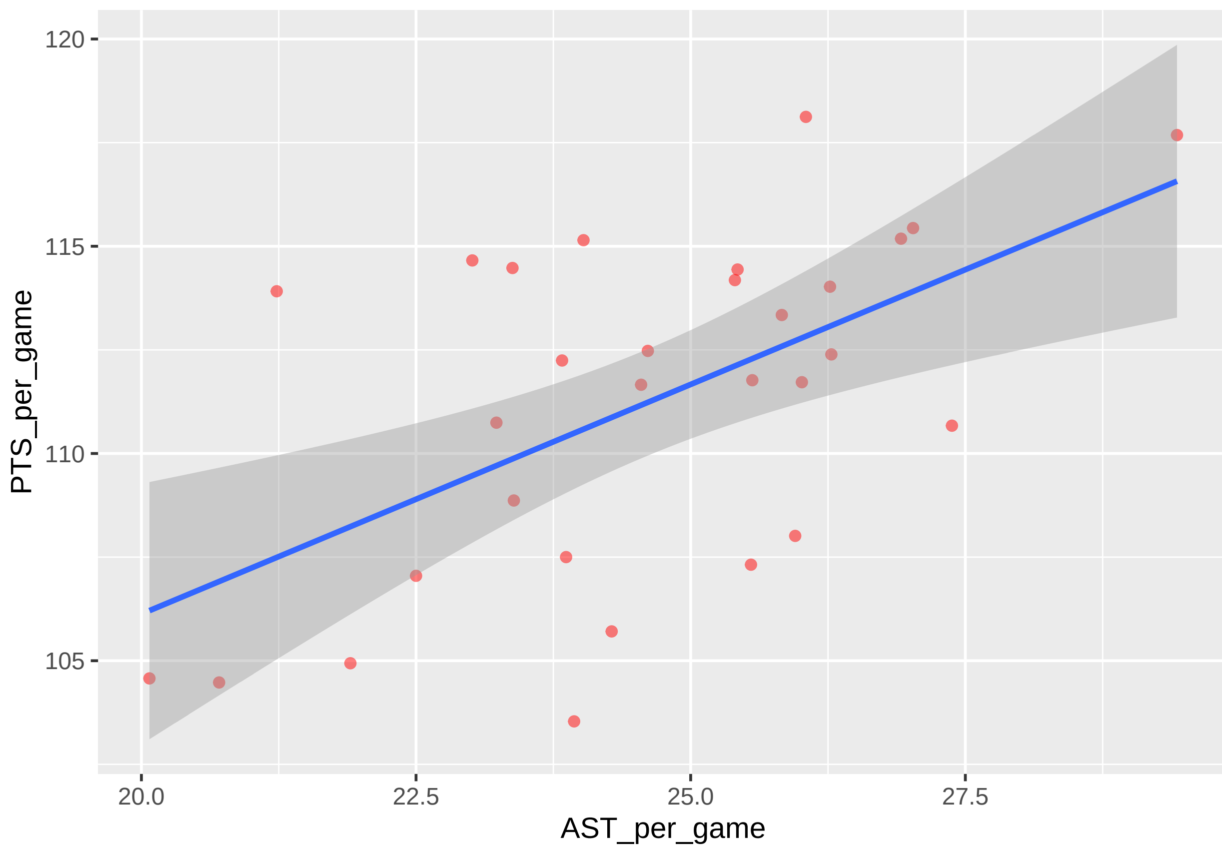
```
ggplot(normalised_team_stat, aes(x = TRB_per_game, y = PTS_per_game))+
geom_point(alpha = 0.5, colour = "red") +
geom_smooth(method = "lm")
```

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



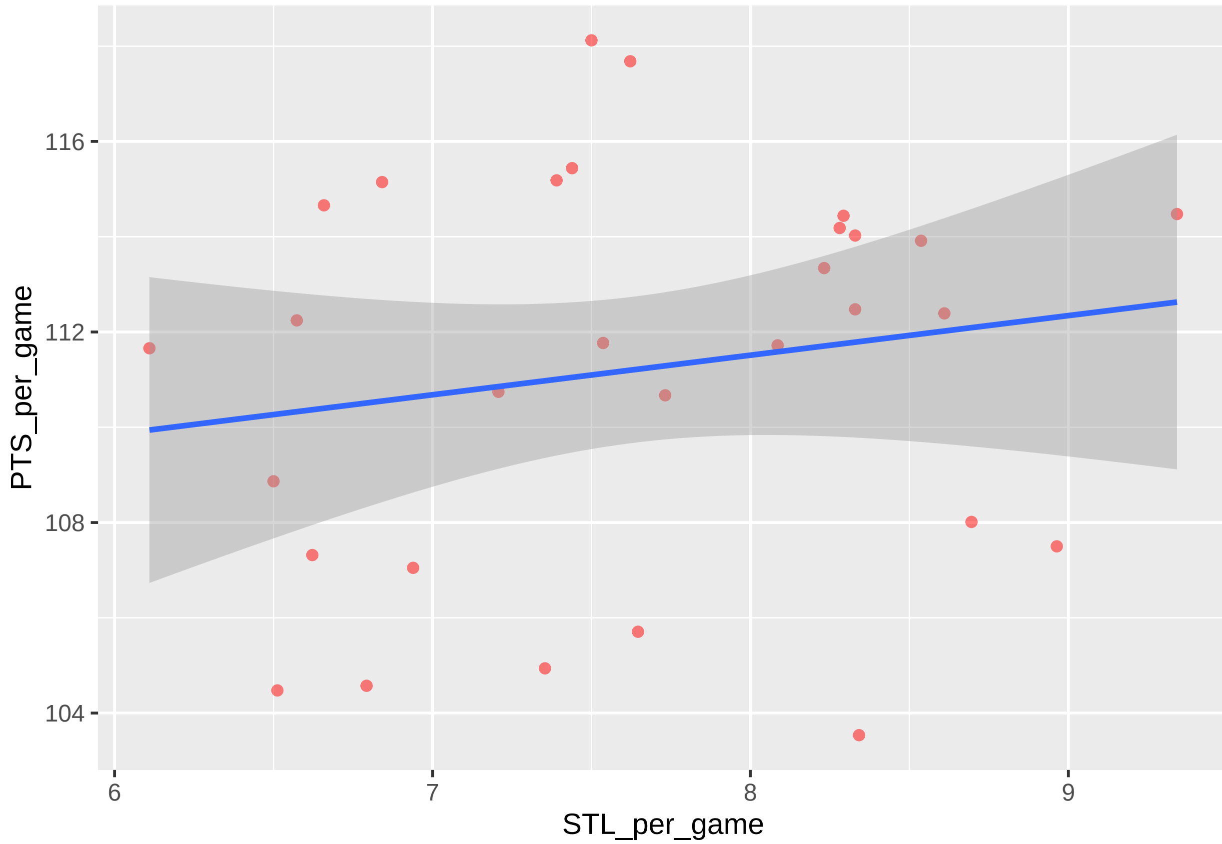

```
ggplot(normalised_team_stat, aes(x = AST_per_game, y = PTS_per_game))+
geom_point(alpha = 0.5, colour = "red") +
geom_smooth(method = "lm")
```

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



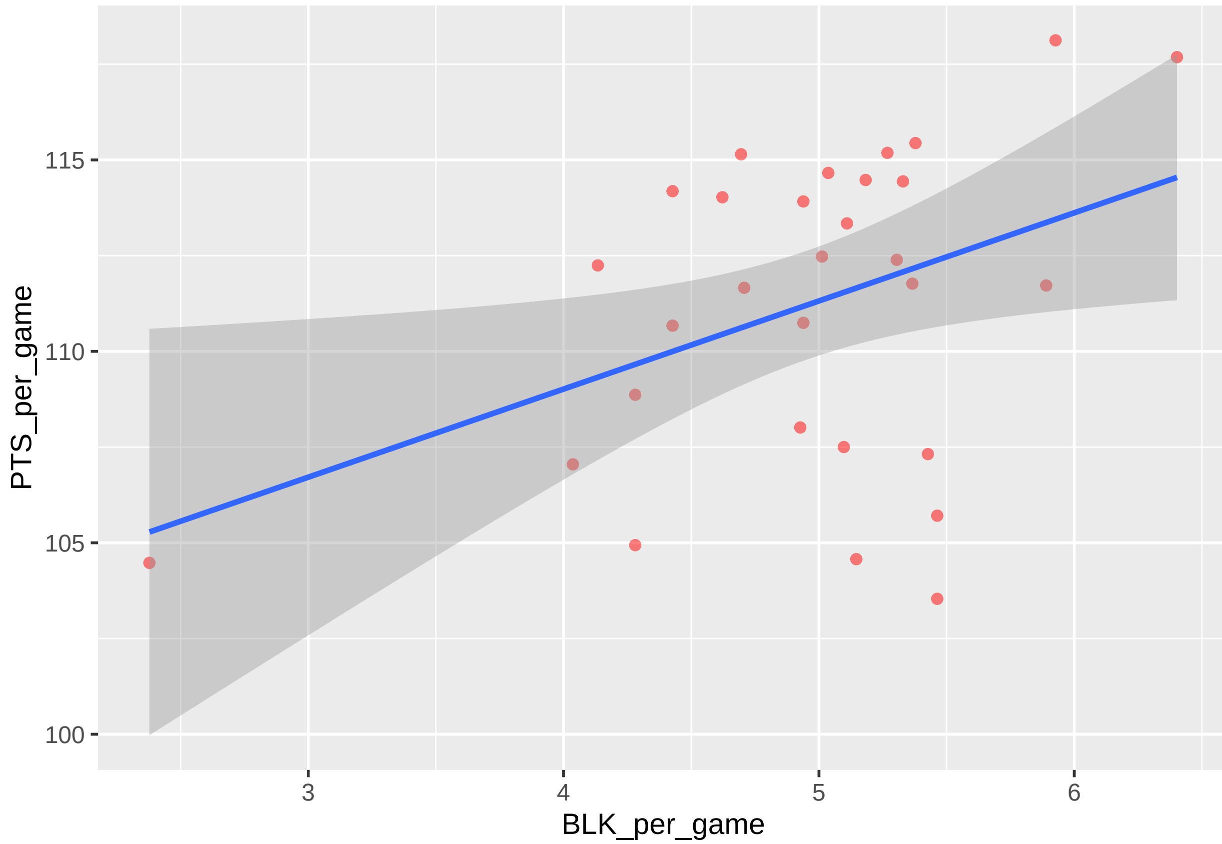

```
ggplot(normalised_team_stat, aes(x = STL_per_game, y = PTS_per_game))+
geom_point(alpha = 0.5, colour = "red") +
geom_smooth(method = "lm")
```

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



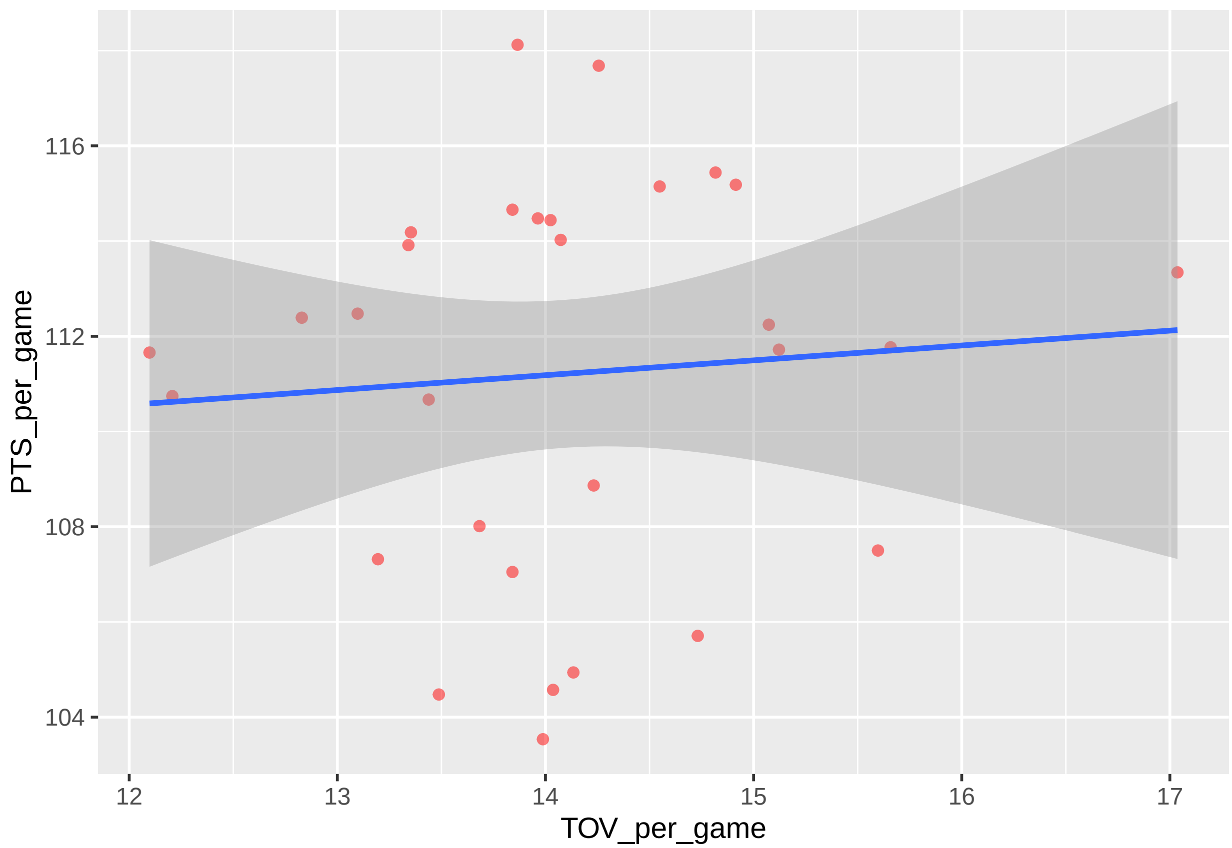

```
ggplot(normalised_team_stat, aes(x = BLK_per_game, y = PTS_per_game))+
geom_point(alpha = 0.5, colour = "red") +
geom_smooth(method = "lm")
```

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




```
ggplot(normalised_team_stat, aes(x = TOV_per_game, y = PTS_per_game))+
geom_point(alpha = 0.5, colour = "red") +
geom_smooth(method = "lm")
```

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



We can see that variables has a linear relations. Now, we create a multiple regression model for exp_PTS_per_game.

```
fit <- lm(PTS_per_game ~
  TRB_per_game  + AST_per_game  + STL_per_game  + BLK_per_game  + TOV_per_game, data = normalised_team_s
tidy(fit, conf.int = TRUE)
```

```
## # A tibble: 6 x 7
##   term         estimate std.error statistic p.value conf.low conf.high
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 (Intercept)    47.5      16.4       2.90  0.00785  13.7       81.4
## 2 TRB_per_game    1.01      0.341     2.97  0.00666   0.309      1.72
## 3 AST_per_game    0.663     0.353     1.88  0.0724   -0.0651     1.39
## 4 STL_per_game    1.11      0.834     1.33  0.196    -0.612      2.83
## 5 BLK_per_game   -0.116     1.05     -0.111 0.913    -2.28       2.05
## 6 TOV_per_game   -0.447     0.593    -0.753 0.459    -1.67       0.778
```

Our Model says:

$expPTSpergame = RBpergame + ASTpergame + STLpergame + BLKpergame + TOVpergame$

Checking Multiple Linear Regression Assumptions

1. The dependent variable should be continuous.

Yes, The Points_per_game is a continuous variable.

2. The independent variables should be continuous

Yes, Each of the independent variables is continuous.

3. Independence of observations

```
car::durbinWatsonTest(fit)
```

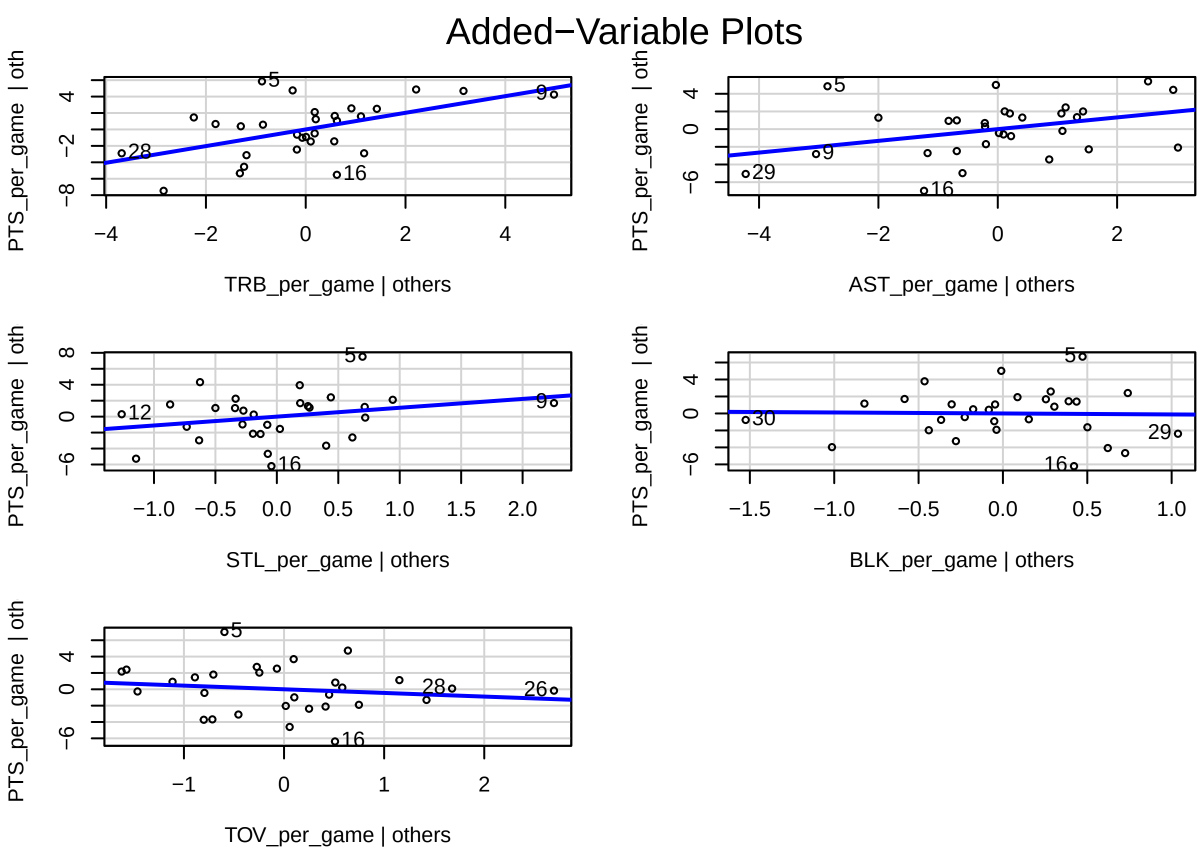
```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.09644946      2.179861    0.65
##  Alternative hypothesis: rho != 0
```

The DW statistics value of 2 indicates that there is no correlation at all. We can see that in our model the value is almost 2. This indicates there is almost no correlation at all among the residuals and that we have independence of observations.

4. Linearity

The dependent variable should have a linear relationship with each independent variable.

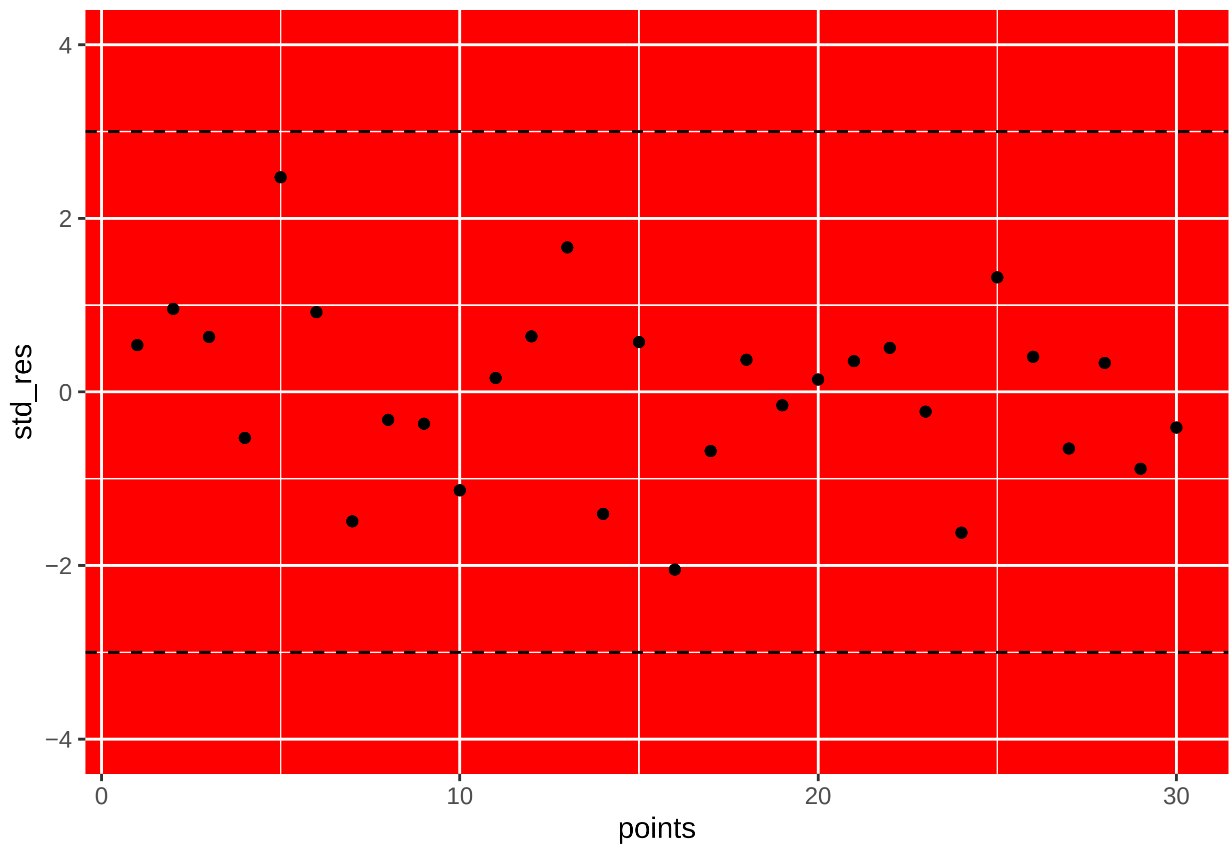
```
car::avPlots(fit)
```



We can see a linear relationship, Though it is weak in case of BLK-per_game.

5. Outliers

```r
#Check the data for outliers.
std_res <- rstandard(fit)
points <- 1:length(std_res)
ggplot(data = NULL, aes(x = points, y = std_res)) +
geom_point(colour = "black") +
ylim(c(-4, 4)) +
geom_hline(yintercept = c(-3, 3), colour = "black", linetype = "dashed")+
theme(panel.background = element_rect(fill = 'red'))
```

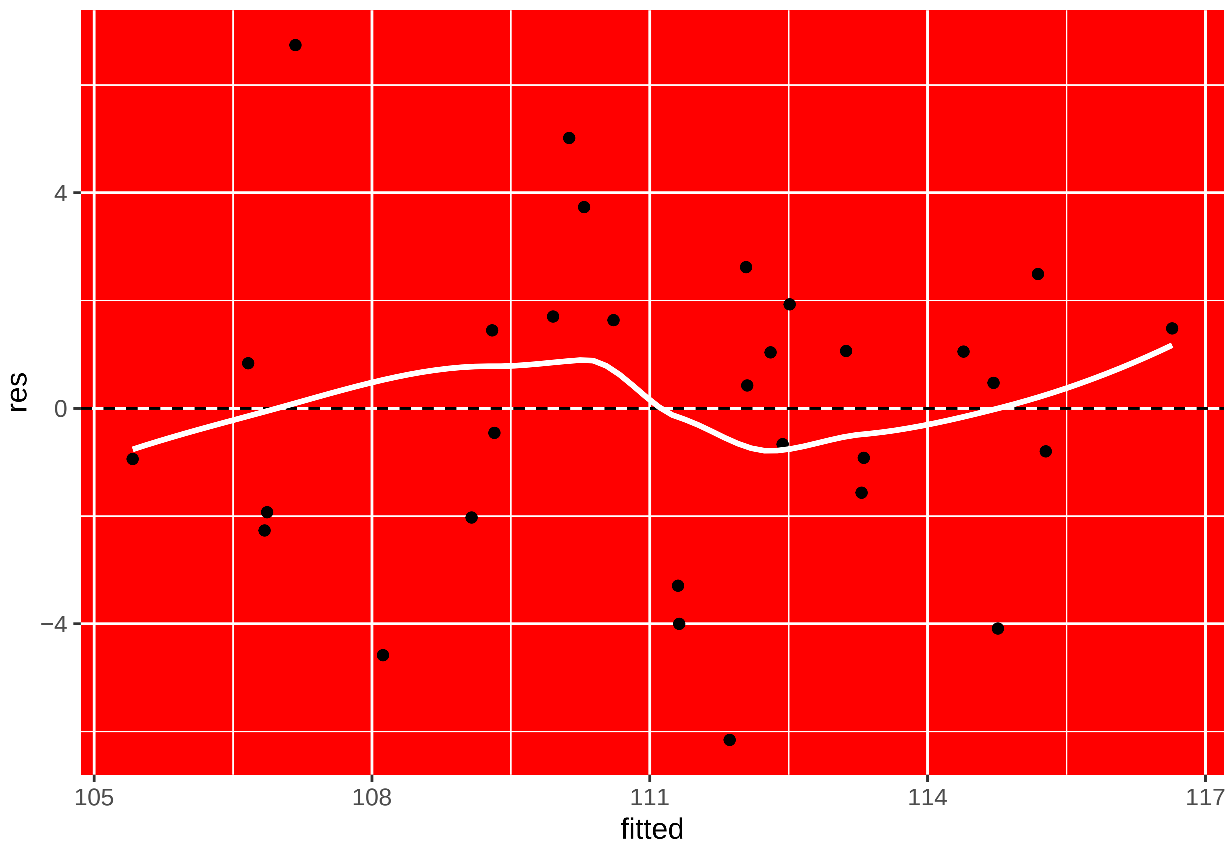


There are some outliers, but most of the points are within 3 standard deviation.

6. Homoscedasticity

```r
res<- residuals(fit)
fitted <- fit %>%  predict()
ggplot(normalised_team_stat, aes(x = fitted, y = res))+
geom_point(colour = "black")+
geom_hline(yintercept = 0, colour = "black", linetype = "dashed")+
theme(panel.background = element_rect(fill = 'red'))+
geom_smooth(se = FALSE, colour = "white")
```

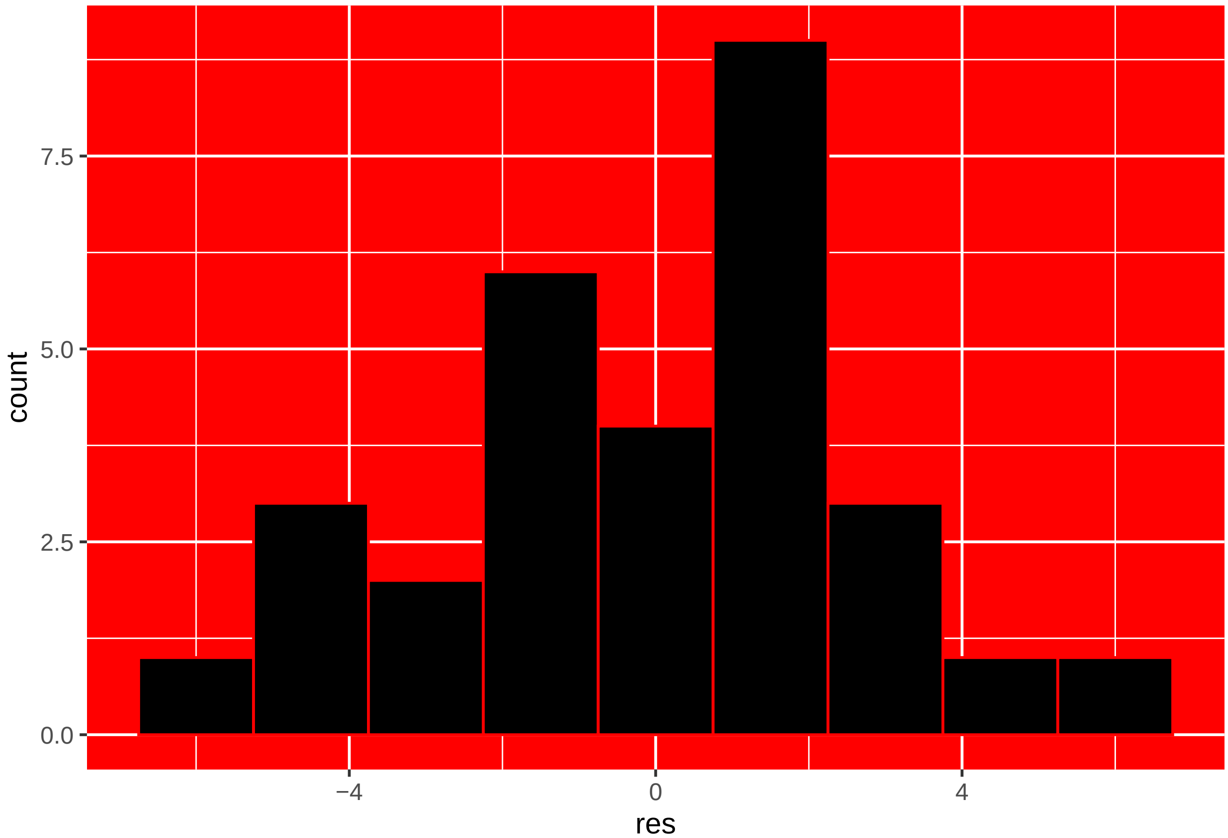
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Yes, the data shows homoscedasticity evident from randomisation visible.

7. Normality

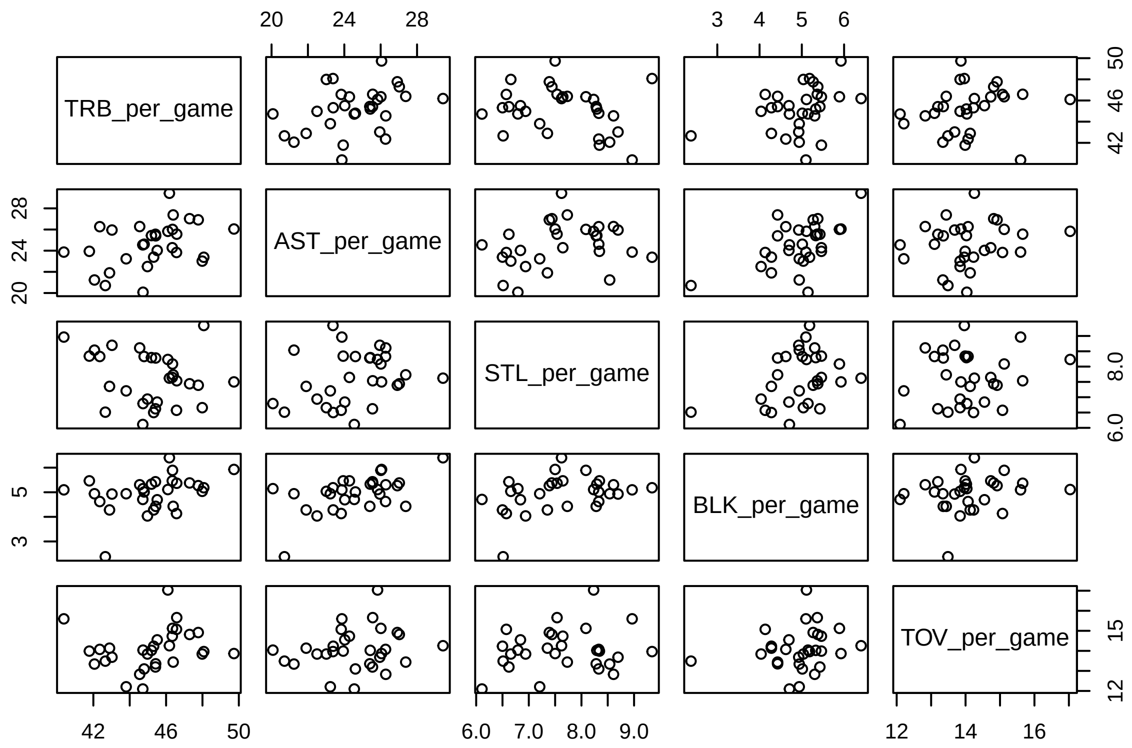
```
ggplot(data =NULL, aes(x= res))+
geom_histogram( colour = "red", fill = "black", binwidth = 1.5)+
theme(panel.background = element_rect(fill = 'red'))
```



Yes, the data points appear normally distributed.

8. Multicollinearity

```
pairs(formula = ~ TRB_per_game  + AST_per_game  + STL_per_game  + BLK_per_game  + TOV_per_game, data = r
```




```r
car::vif(fit)
```

```
## TRB_per_game AST_per_game STL_per_game BLK_per_game TOV_per_game
##     1.532366     1.598148     1.457992     1.713277     1.088499
```

```r
sqrt(car::vif(fit))
```

```
## TRB_per_game AST_per_game STL_per_game BLK_per_game TOV_per_game
##     1.237888     1.264179     1.207474     1.308922     1.043311
```

The model does not show any multicollinearity and this is the required aspect.

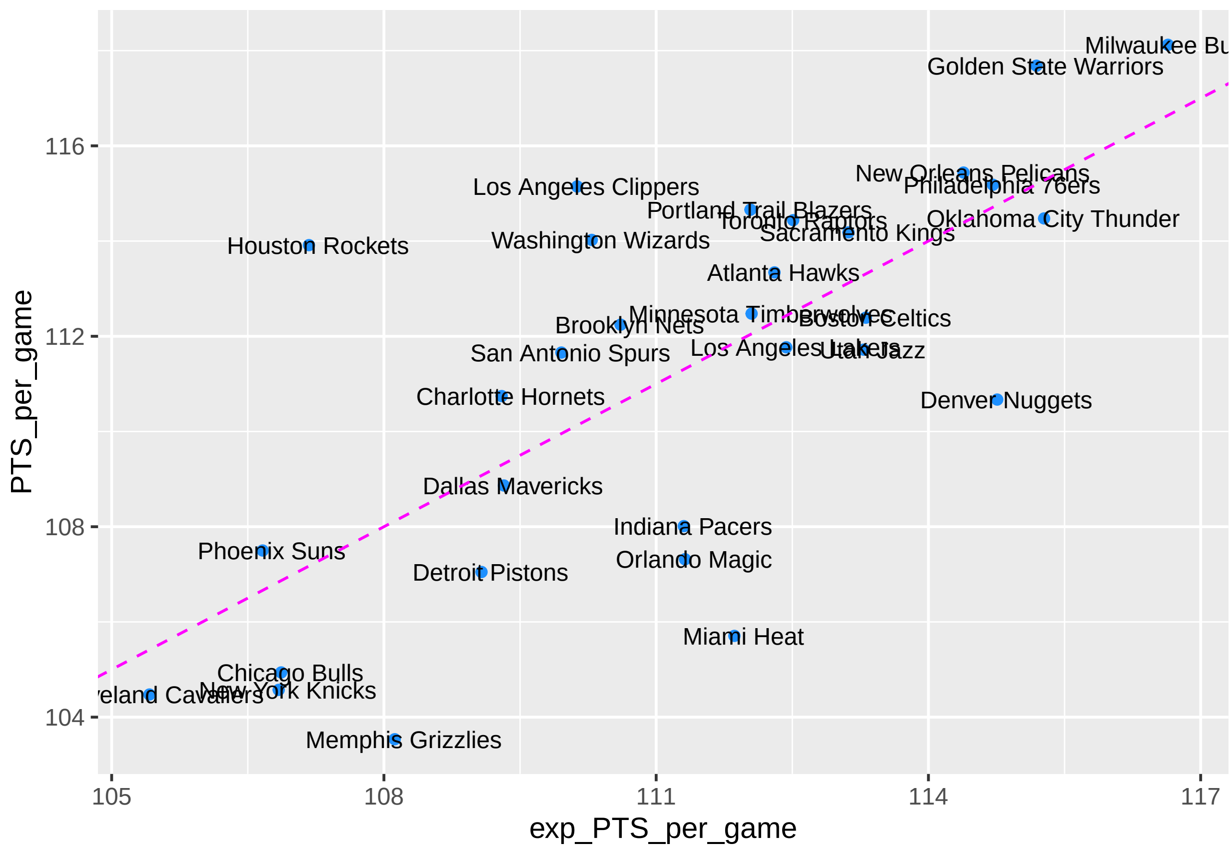
Model Testing & Player Metric

Applying the model

```r
normalised_team_stat <- mutate(normalised_team_stat,exp_PTS_per_game=predict(fit,newdata=normalised_tea
```

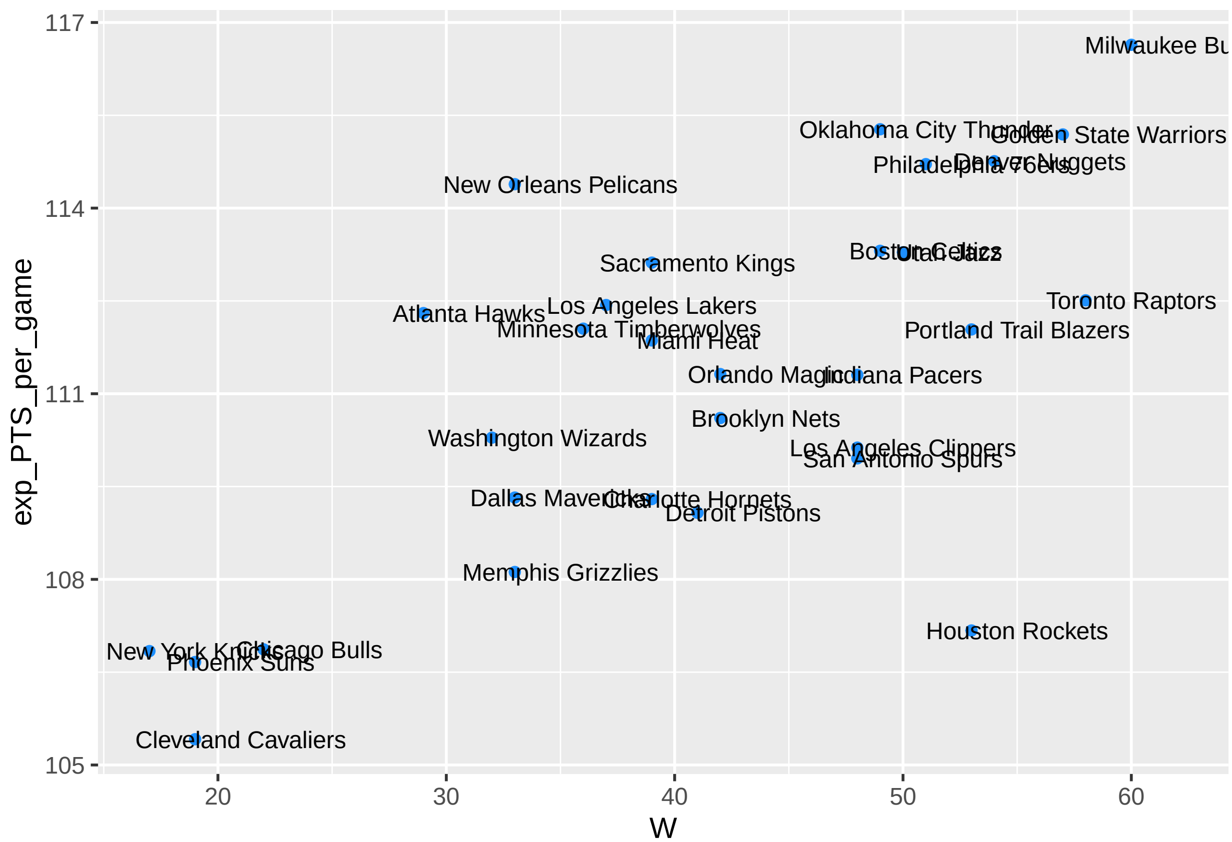
graph

```r
ggplot(normalised_team_stat, aes(exp_PTS_per_game, PTS_per_game, label = Team)) +
geom_point(colour = "dodgerblue") +
geom_text(nudge_x = 0.1, cex = 3) +
geom_abline(linetype = "dashed", colour = "magenta")
```



graph

```
ggplot(normalised_team_stat, aes(x = W, y = exp_PTS_per_game, label = Team)) +
geom_point(colour = "dodgerblue") +
geom_text(nudge_x = 2, cex = 3)
```



Player Metric

At first we normalize the player_stat

```
normalised_player_stat <- player_stat %>% mutate(
  PTS_per_game = PTS / G,
  TRB_per_game = TRB / G,
  AST_per_game = AST / G,
  STL_per_game = STL / G,
  BLK_per_game = BLK / G,
  TOV_per_game = TOV / G)
str(normalised_player_stat)
```

```
## grouped_df [454 x 38] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ player_name : chr [1:454] "Alex Abrines" "Quincy Acy" "Jaylen Adams" "Steven Adams" ...
##  $ Pos         : chr [1:454] "SG" "PF" "PG" "C" ...
##  $ Age         : num [1:454] 25 28 22 25 21 21 33 21 23 20 ...
##  $ Tm          : chr [1:454] "OKC" "PHO" "ATL" "OKC" ...
##  $ G           : num [1:454] 31 10 34 80 82 19 81 10 38 80 ...
##  $ GS          : num [1:454] 2 0 1 80 28 3 81 1 2 80 ...
##  $ MP          : num [1:454] 588 123 428 2669 1913 ...
##  $ FG          : num [1:454] 56 4 38 481 280 11 684 13 67 335 ...
##  $ FGA         : num [1:454] 157 18 110 809 486 ...
##  $ FGp         : num [1:454] 0.357 0.222 0.345 0.595 0.576 0.306 0.519 0.333 0.376 0.59 ...
##  $ x3P         : num [1:454] 41 2 25 0 3 6 10 3 32 6 ...
##  $ x3PA        : num [1:454] 127 15 74 2 15 23 42 12 99 45 ...
```

```
##  $ x3Pp        : num [1:454] 0.323 0.133 0.338 0 0.2 0.261 0.238 0.25 0.323 0.133 ...
##  $ x2P         : num [1:454] 15 2 13 481 277 5 674 10 35 329 ...
##  $ x2PA        : num [1:454] 30 3 36 807 471 ...
##  $ x2Pp        : num [1:454] 0.5 0.667 0.361 0.596 0.588 0.385 0.528 0.37 0.443 0.629 ...
##  $ eFGp        : num [1:454] 0.487 0.278 0.459 0.595 0.579 0.389 0.522 0.372 0.466 0.595 ...
##  $ FT          : num [1:454] 12 7 7 146 166 4 349 8 45 197 ...
##  $ FTA         : num [1:454] 13 10 9 292 226 4 412 12 60 278 ...
##  $ FTp         : num [1:454] 0.923 0.7 0.778 0.5 0.735 1 0.847 0.667 0.75 0.709 ...
##  $ ORB         : num [1:454] 5 3 11 391 165 3 251 11 3 191 ...
##  $ DRB         : num [1:454] 43 22 49 369 432 16 493 15 20 481 ...
##  $ TRB         : num [1:454] 48 25 60 760 597 19 744 26 23 672 ...
##  $ AST         : num [1:454] 20 8 65 124 184 5 194 13 25 110 ...
##  $ STL         : num [1:454] 17 1 14 117 71 1 43 1 6 43 ...
##  $ BLK         : num [1:454] 6 4 5 76 65 4 107 0 6 120 ...
##  $ TOV         : num [1:454] 14 4 28 135 121 6 144 8 33 103 ...
##  $ PF          : num [1:454] 53 24 45 204 203 13 179 7 47 184 ...
##  $ PTS         : num [1:454] 165 17 108 1108 729 ...
##  $ n           : int [1:454] 1 1 1 1 1 1 1 1 1 1 ...
##  $ player_id   : num [1:454] 1 2 4 3 5 6 10 11 12 13 ...
##  $ salary      : num [1:454] 3667645 213948 236854 24157304 2955840 ...
##  $ PTS_per_game: num [1:454] 5.32 1.7 3.18 13.85 8.89 ...
##  $ TRB_per_game: num [1:454] 1.55 2.5 1.76 9.5 7.28 ...
##  $ AST_per_game: num [1:454] 0.645 0.8 1.912 1.55 2.244 ...
##  $ STL_per_game: num [1:454] 0.548 0.1 0.412 1.462 0.866 ...
##  $ BLK_per_game: num [1:454] 0.194 0.4 0.147 0.95 0.793 ...
##  $ TOV_per_game: num [1:454] 0.452 0.4 0.824 1.688 1.476 ...
##  - attr(*, "groups")= tibble [454 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ player_name: chr [1:454] "Aaron Gordon" "Aaron Holiday" "Abdel Nader" "Al Horford" ...
##   ..$ .rows      : list<int> [1:454]
##   .. ..$ : int 169
##   .. ..$ : int 202
##   .. ..$ : int 317
##   .. ..$ : int 208
##   .. ..$ : int 12
##   .. ..$ : int 76
##   .. ..$ : int 1
##   .. ..$ : int 89
##   .. ..$ : int 262
##   .. ..$ : int 355
##   .. ..$ : int 292
##   .. ..$ : int 229
##   .. ..$ : int 107
##   .. ..$ : int 412
##   .. ..$ : int 226
##   .. ..$ : int 230
##   .. ..$ : int 132
##   .. ..$ : int 213
##   .. ..$ : int 53
##   .. ..$ : int 189
##   .. ..$ : int 437
##   .. ..$ : int 386
##   .. ..$ : int 454
##   .. ..$ : int 115
##   .. ..$ : int 409
```

```
##   .. ..$ : int 48
##   .. ..$ : int 34
##   .. ..$ : int 363
##   .. ..$ : int 58
##   .. ..$ : int 5
##   .. ..$ : int 293
##   .. ..$ : int 384
##   .. ..$ : int 46
##   .. ..$ : int 180
##   .. ..$ : int 281
##   .. ..$ : int 352
##   .. ..$ : int 51
##   .. ..$ : int 52
##   .. ..$ : int 429
##   .. ..$ : int 36
##   .. ..$ : int 168
##   .. ..$ : int 215
##   .. ..$ : int 247
##   .. ..$ : int 375
##   .. ..$ : int 271
##   .. ..$ : int 67
##   .. ..$ : int 79
##   .. ..$ : int 151
##   .. ..$ : int 199
##   .. ..$ : int 396
##   .. ..$ : int 345
##   .. ..$ : int 266
##   .. ..$ : int 17
##   .. ..$ : int 337
##   .. ..$ : int 211
##   .. ..$ : int 342
##   .. ..$ : int 156
##   .. ..$ : int 357
##   .. ..$ : int 123
##   .. ..$ : int 297
##   .. ..$ : int 57
##   .. ..$ : int 344
##   .. ..$ : int 446
##   .. ..$ : int 287
##   .. ..$ : int 299
##   .. ..$ : int 438
##   .. ..$ : int 83
##   .. ..$ : int 451
##   .. ..$ : int 380
##   .. ..$ : int 59
##   .. ..$ : int 241
##   .. ..$ : int 260
##   .. ..$ : int 147
##   .. ..$ : int 373
##   .. ..$ : int 42
##   .. ..$ : int 236
##   .. ..$ : int 267
##   .. ..$ : int 261
##   .. ..$ : int 130
```

```
##   .. ..$ : int 181
##   .. ..$ : int 402
##   .. ..$ : int 158
##   .. ..$ : int 175
##   .. ..$ : int 111
##   .. ..$ : int 143
##   .. ..$ : int 209
##   .. ..$ : int 453
##   .. ..$ : int 300
##   .. ..$ : int 101
##   .. ..$ : int 328
##   .. ..$ : int 43
##   .. ..$ : int 360
##   .. ..$ : int 153
##   .. ..$ : int 296
##   .. ..$ : int 22
##   .. ..$ : int 240
##   .. ..$ : int 447
##   .. ..$ : int 121
##   .. ..$ : int 105
##   .. .. [list output truncated]
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

Next, step is to calculate player specific expected points.

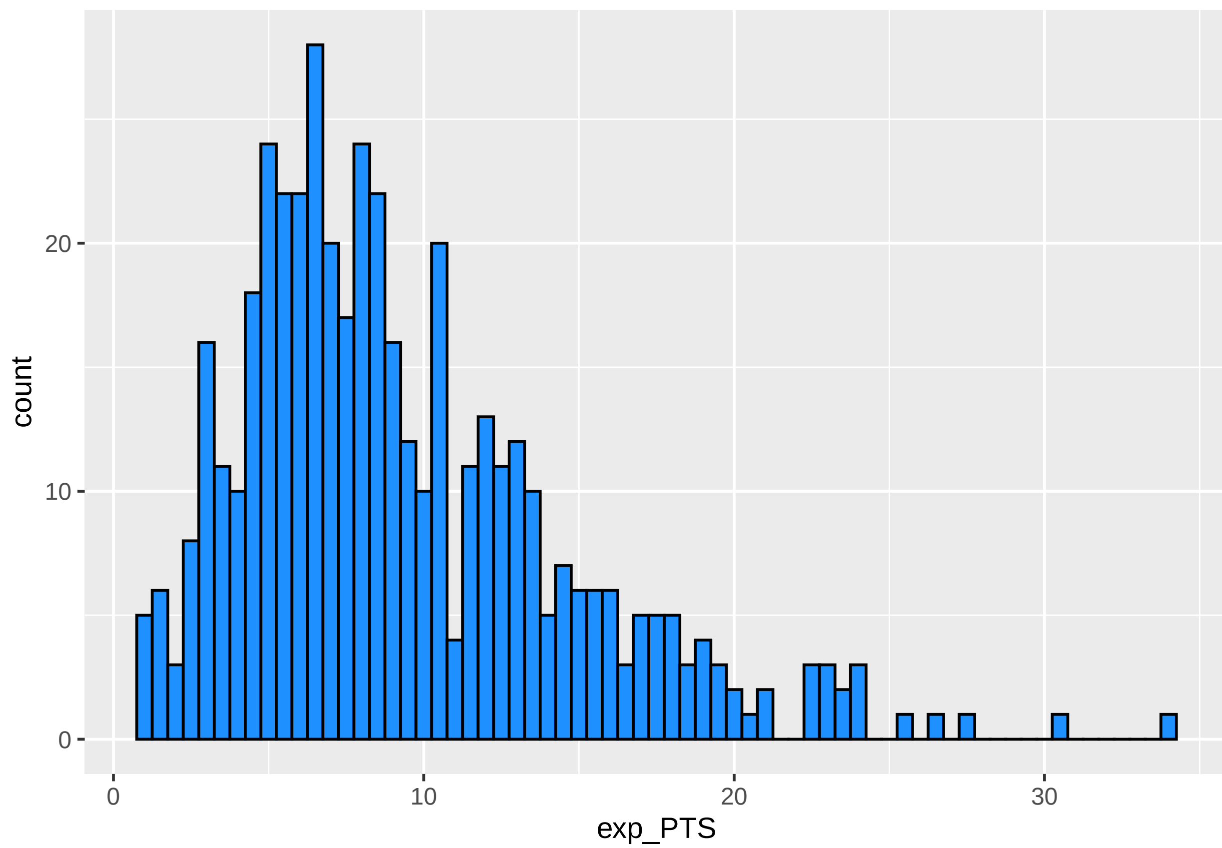
```
fit <- lm(PTS_per_game ~
  TRB_per_game  + AST_per_game  + STL_per_game  + BLK_per_game  + TOV_per_game, data = normalised_player
tidy(fit, conf.int = TRUE)
```

```
## # A tibble: 6 x 7
##   term         estimate std.error statistic  p.value conf.low conf.high
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)    0.357     0.287       1.24  2.14e- 1   -0.207     0.920
## 2 TRB_per_game   0.368     0.0929      3.96  8.59e- 5    0.186     0.551
## 3 AST_per_game  -0.587     0.173      -3.39  7.48e- 4   -0.927    -0.247
## 4 STL_per_game   2.58      0.497       5.19  3.14e- 7    1.60      3.56
## 5 BLK_per_game  -0.0830    0.473      -0.175 8.61e- 1   -1.01      0.847
## 6 TOV_per_game   6.15      0.418      14.7   3.19e-40    5.33      6.97
```

```
normalised_player_stat <- normalised_player_stat%>% ungroup() %>% mutate(exp_PTS = predict(fit,newdata =
```

histogram

```
normalised_player_stat %>%
ggplot(aes(x = exp_PTS)) +
geom_histogram(binwidth = 0.5, colour = "black", fill = "dodgerblue")
```

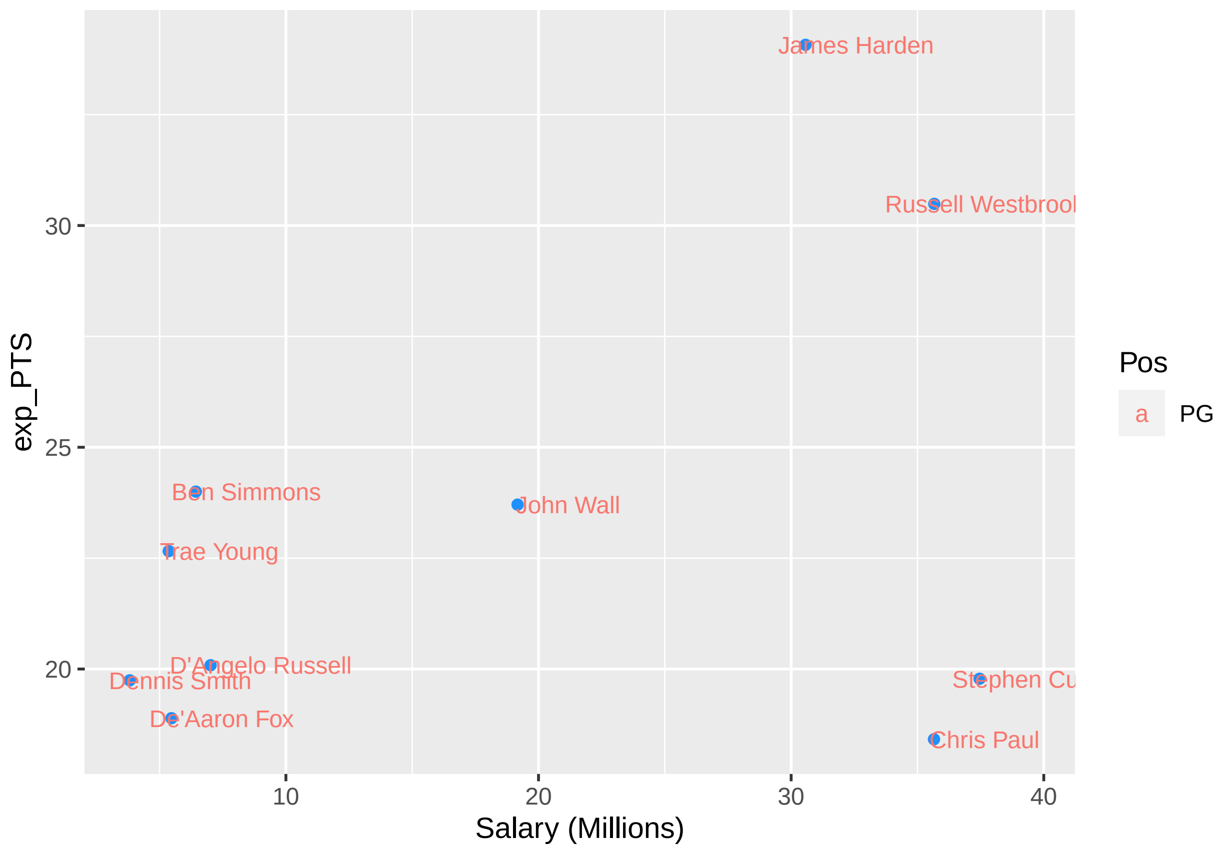


5. Player recommendations

Value for money player for Point Guard Position

```
## Selecting by exp_PTS
```

```
## # A tibble: 10 x 4
##    player_name       Pos      salary exp_PTS
##    <chr>             <chr>     <dbl>   <dbl>
##  1 James Harden      PG     30570000    34.1
##  2 Russell Westbrook PG     35665000    30.5
##  3 Ben Simmons       PG      6434520    24.0
##  4 John Wall         PG     19169800    23.7
##  5 Trae Young        PG      5363280    22.7
##  6 D'Angelo Russell  PG      7019698    20.1
##  7 Stephen Curry     PG     37457154    19.8
##  8 Dennis Smith      PG      3819960    19.7
##  9 De'Aaron Fox      PG      5470920    18.9
## 10 Chris Paul        PG     35654150    18.4
```

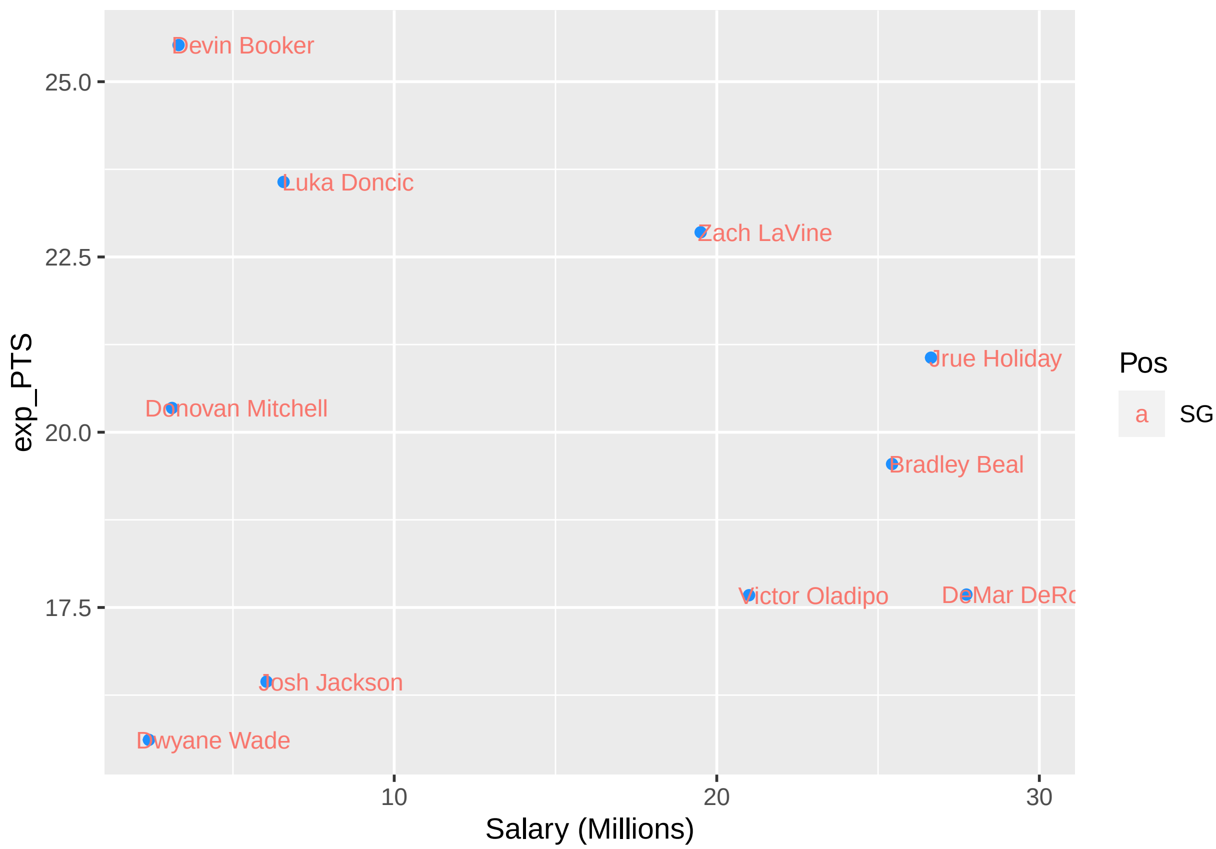


Ben Simmons & Trae Young are two players best suited for point guard position. We can see that Ben Simmons has exp_PTS of 24.0 at just 6.43 millions. So, he is our player for point guard position. More detail of players can be see at NBA website(5). We can consider Trae Young for bench strength.

Value for money player for Shooting Guard Position

```
## Selecting by exp_PTS
```

```
## # A tibble: 10 x 4
##    player_name      Pos      salary exp_PTS
##    <chr>            <chr>     <dbl>   <dbl>
##  1 Devin Booker     SG      3314365    25.5
##  2 Luka Doncic      SG      6569040    23.6
##  3 Zach LaVine      SG     19500000    22.9
##  4 Jrue Holiday     SG     26641111    21.1
##  5 Donovan Mitchell SG      3111480    20.3
##  6 Bradley Beal     SG     25434262    19.5
##  7 DeMar DeRozan    SG     27739975    17.7
##  8 Victor Oladipo   SG     21000000    17.7
##  9 Josh Jackson     SG      6041520    16.4
## 10 Dwyane Wade      SG      2393887    15.6
```

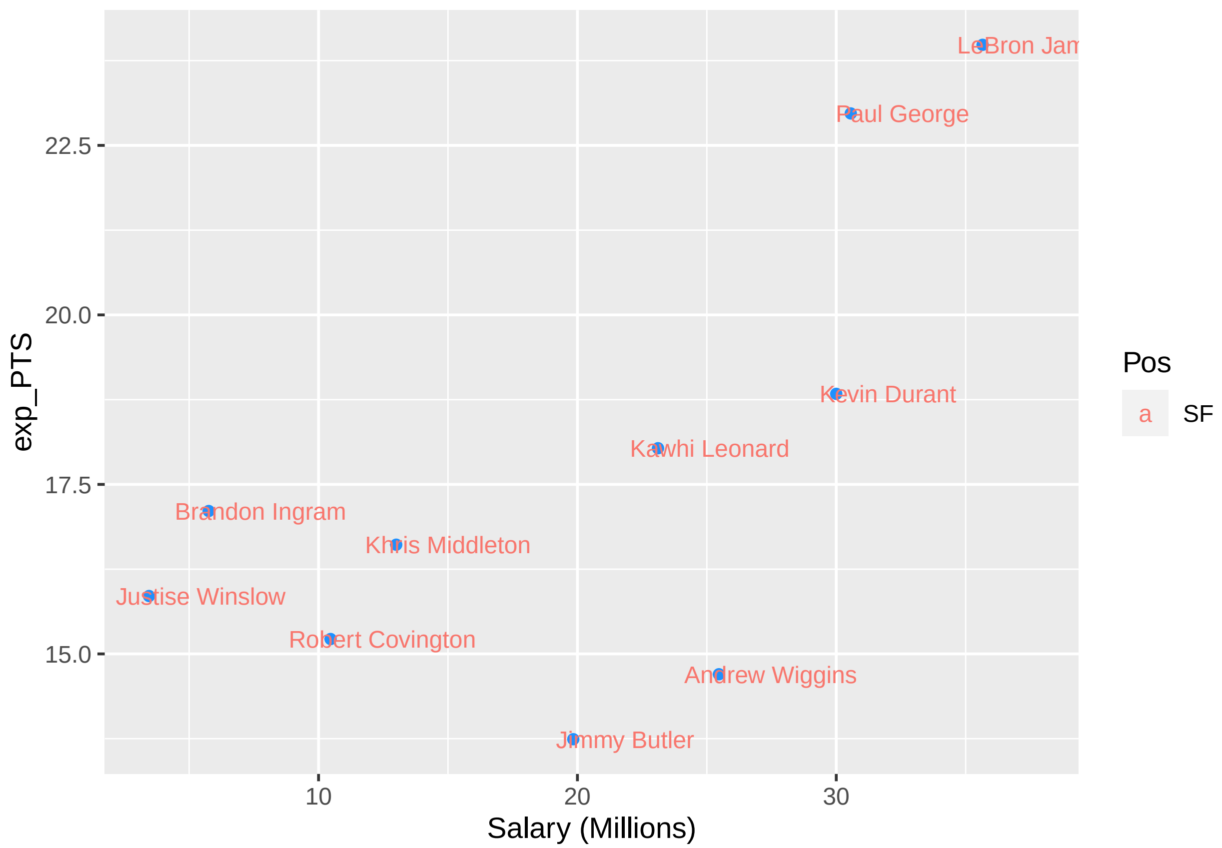


Devin Booker & Luka Doncic are two players best suited for shooting guard position. We can see that Devin Booker has exp_PTS of 25.5 just at 3.31 millions. So, he is our player for shooting guard position. We can put Luka Doncic for bench strength.

Value for money player for Small Forward Position

```
## Selecting by exp_PTS
```

```
## # A tibble: 10 x 4
##    player_name      Pos      salary exp_PTS
##    <chr>            <chr>     <dbl>   <dbl>
##  1 LeBron James     SF     35654150    24.0
##  2 Paul George      SF     30560700    23.0
##  3 Kevin Durant     SF     30000000    18.8
##  4 Kawhi Leonard    SF     23114066    18.0
##  5 Brandon Ingram   SF      5757120    17.1
##  6 Khris Middleton  SF     13000000    16.6
##  7 Justise Winslow  SF      3448926    15.9
##  8 Robert Covington SF     10464092    15.2
##  9 Andrew Wiggins   SF     25467250    14.7
## 10 Jimmy Butler     SF     19841627    13.7
```

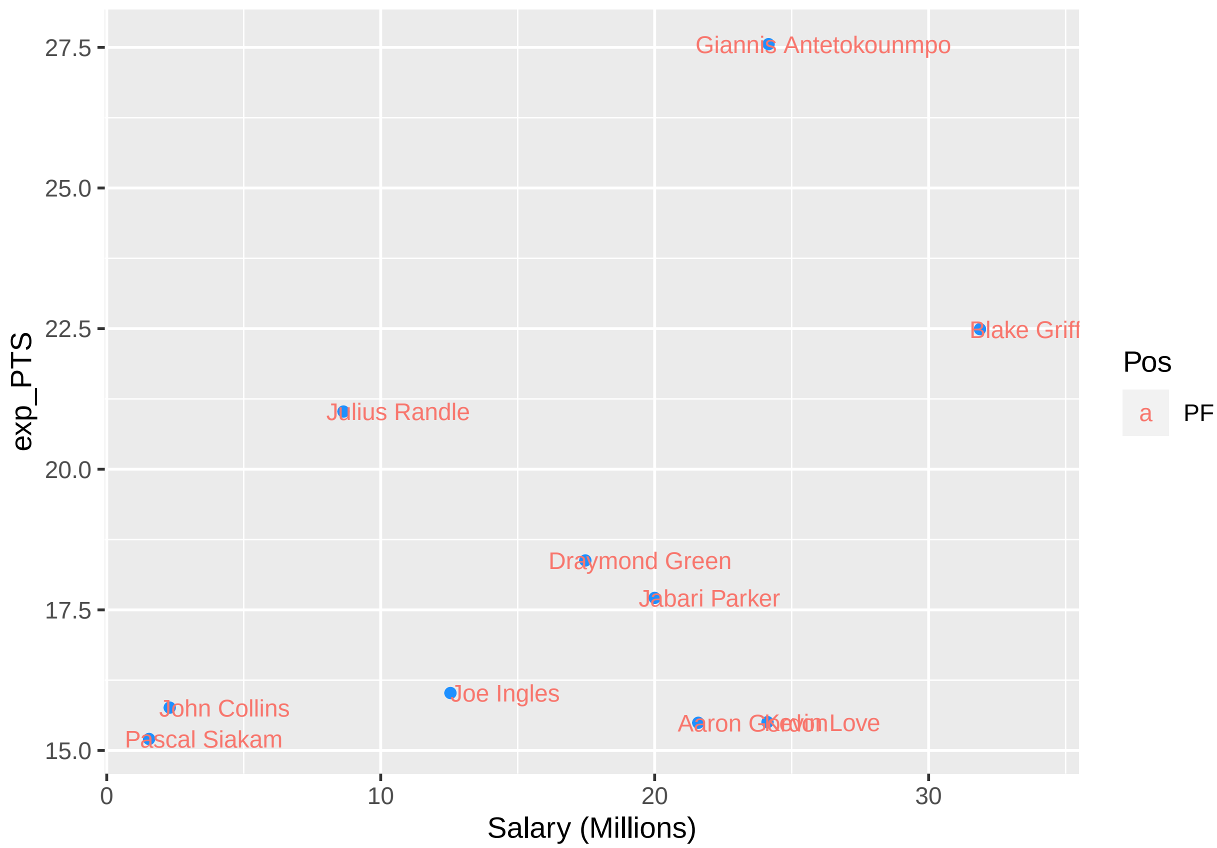


Brandon Ingram & Justise Winslow are two players for small forward position. We can see that Brandon Ingram has exp_PTS of 17.1 just at 5.75 millions. So, he is our player for small forward position. We can put Justise Winslow for bench strength.

Value for money player for Power Forward Position

```
## Selecting by exp_PTS
```

```
## # A tibble: 10 x 4
##    player_name           Pos      salary exp_PTS
##    <chr>                 <chr>     <dbl>   <dbl>
##  1 Giannis Antetokounmpo PF     24157304    27.6
##  2 Blake Griffin         PF     31873932    22.5
##  3 Julius Randle         PF      8641000    21.0
##  4 Draymond Green        PF     17469565    18.4
##  5 Jabari Parker         PF     20000000    17.7
##  6 Joe Ingles            PF     12545455    16.0
##  7 John Collins          PF      2299080    15.8
##  8 Kevin Love            PF     24119025    15.5
##  9 Aaron Gordon          PF     21590909    15.5
## 10 Pascal Siakam         PF      1544951    15.2
```



Julius Randle & John Collins are two best players for power forward position. We can see that Julius Randle has exp_PTS of 21 at just 8.64 millions. So, he is our player for power forward position. We can put John Collins for bench strenght.

Value for money player for Center Position

```
## Selecting by exp_PTS

## # A tibble: 10 x 4
##    player_name        Pos     salary exp_PTS
##    <chr>              <chr>    <dbl>   <dbl>
##  1 Joel Embiid        C     25467250    26.6
##  2 Karl-Anthony Towns C      7839435    24.2
##  3 Andre Drummond     C     25434262    23.2
##  4 Nikola Jokic       C     25467250    22.6
##  5 DeMarcus Cousins   C      5337000    19.4
##  6 Jusuf Nurkic       C     11111111    19.1
##  7 DeAndre Jordan     C     22897200    19.0
##  8 Anthony Davis      C     25434263    18.6
##  9 Nikola Vucevic     C     12750000    17.3
## 10 Domantas Sabonis   C      2659800    17.0
```

Karl-Anthony Towns & DeMarcus Cousins, are two players suited for center position. We can see that Karl-Anthony Towns has exp_PTS of 24.2 at just 7.83 millions. So, he is our player for center position. We can put DeMarcus Cousins for bench strength.

Lets save the processed data

```r
write_csv(x = final_players, path = "data/processed/finalplayers.csv")
```

```
## Warning: The ‘path‘ argument of ‘write_csv()‘ is deprecated as of readr 1.4.0.
## Please use the ‘file‘ argument instead.
```

```r
write_csv(x = normalised_player_stat, path = "data/processed/playersstat.csv")
write_csv(x = normalised_team_stat, path = "data/processed/teamstat.csv")
```

6. Summary

```
##   POSITION               PLAYER SALARYmillions
## 1       PG          Ben Simmons           6.43
## 2       SG         Devin Booker           3.31
## 3       SF       Brandon Ingram           5.75
## 4       PF        Julius Randle           8.64
## 5        C Karl-Anthony Towns           7.83
## 6                          TOTAL          31.96
```

We are able to find top 5 value for money players in just 26.19 millions. We are left with ample money to make remaining team.

7. Reference List

1. [Internet]. Rstudio.com. 2022 [cited 3 May 2022]. Available from: https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

2. Chicago Bulls Basketball - Bulls News, Scores, Stats, Rumors & More | ESPN [Internet]. ESPN. 2022 [cited 3 May 2022]. Available from: https://www.espn.in/nba/team/_/name/chi/chicago-bulls

3. García J, Ibáñez S, Martinez De Santos R, Leite N, Sampaio J. Identifying Basketball Performance Indicators in Regular Season and Playoff Games. Journal of Human Kinetics [Internet]. 2013 [cited 27 April 2022];36(1):161-168. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3661887/

4. Csataljay G, O’Donoghue P, Hughes M, Dancs H. Performance indicators that distinguish winning and losing teams in basketball. International Journal of Performance Analysis in Sport [Internet]. 2009 [cited 1 May 2022];9(1):60-66. Available from: https://www.researchgate.net/publication/233682287_Performance_indicators_that_distinguish_winning_and_losing_teams_in_basketball

5. NBA Players & Team Rosters | NBA.com [Internet]. Nba.com. 2022 [cited 3 May 2022]. Available from: https://www.nba.com/players