NBA Players Data Report

The first step in this data report is to download the data that will be analyzed. For this I have taken data from all NBA games from October 22nd 2019 to February 12th 2021. This data can be downloaded here

```
rawdata <- read.csv("ASA All NBA Raw Data.csv")
summary(rawdata)
```

```
##          game_id           game_date           OT          H_A
##  202102020WAS:   84   2021-02-02:  465   Min.   :0.00000   A:19409
##  202102020UTA:   81   2021-01-30:  440   1st Qu.:0.00000   H:19500
##  202102020GSW:   78   2019-11-27:  351   Median :0.00000
##  202102020BRK:   75   2019-12-28:  350   Mean   :0.06914
##  202102020ORL:   75   2020-01-20:  349   3rd Qu.:0.00000
##  202102020IND:   72   2020-12-23:  336   Max.   :2.00000
##  (Other)     :38444   (Other)   :36618
##   Team_Abbrev      Team_Score      Team_pace       Team_efg_pct
##  MIA    : 1579   Min.   : 73.0   Min.   : 85.60   Min.   :0.3350
##  LAL    : 1517   1st Qu.:103.0   1st Qu.: 95.90   1st Qu.:0.4860
##  BOS    : 1503   Median :111.0   Median : 99.10   Median :0.5300
##  DEN    : 1503   Mean   :111.7   Mean   : 99.34   Mean   :0.5324
##  LAC    : 1472   3rd Qu.:120.0   3rd Qu.:102.70   3rd Qu.:0.5760
##  TOR    : 1413   Max.   :159.0   Max.   :116.10   Max.   :0.7620
##  (Other):29922
##   Team_tov_pct    Team_orb_pct    Team_ft_rate     Team_off_rtg
##  Min.   : 1.90   Min.   : 0.00   Min.   :0.0300   Min.   : 76.9
##  1st Qu.: 9.80   1st Qu.:17.40   1st Qu.:0.1460   1st Qu.:104.2
##  Median :12.30   Median :22.00   Median :0.1950   Median :111.8
##  Mean   :12.28   Mean   :22.18   Mean   :0.2026   Mean   :111.7
##  3rd Qu.:14.50   3rd Qu.:26.80   3rd Qu.:0.2500   3rd Qu.:119.1
##  Max.   :27.80   Max.   :46.30   Max.   :0.5730   Max.   :154.0
##
##                                                                  Inactives
##  Drew Eubanks, Keldon Johnson, Luka Samanic, Quinndary Weatherspoon         :  299
##  Andre Iguodala, Josh Jackson, John Konchar, Yuta Watanabe                  :  273
##  Zylan Cheatham, Josh Gray, Darius Miller, Zion Williamson                  :  247
##  Ryan Broekhoff, Antonius Cleveland, Josh Reaves, Isaiah Roby               :  234
##  Kostas Antetokounmpo, Devontae Cacok, DeMarcus Cousins, Talen Horton-Tucker:  221
##  Kyle Alexander, KZ Okpala, Chris Silva, Gabe Vincent                       :  208
##  (Other)                                                                    :37427
##  Opponent_Abbrev Opponent_Score  Opponent_pace    Opponent_efg_pct
##  MIA    : 1532   Min.   : 73.0   Min.   : 85.60   Min.   :0.3350
##  LAL    : 1518   1st Qu.:103.0   1st Qu.: 95.90   1st Qu.:0.4850
##  BOS    : 1486   Median :111.0   Median : 99.10   Median :0.5300
##  DEN    : 1466   Mean   :111.6   Mean   : 99.34   Mean   :0.5321
##  LAC    : 1439   3rd Qu.:120.0   3rd Qu.:102.70   3rd Qu.:0.5760
##  TOR    : 1418   Max.   :159.0   Max.   :116.10   Max.   :0.7620
##  (Other):30050
##  Opponent_tov_pct Opponent_orb_pct Opponent_ft_rate Opponent_off_rtg
```

```
##  Min.   : 1.90    Min.   : 0.00    Min.   :0.0300   Min.   : 76.9
##  1st Qu.: 9.80    1st Qu.:17.40    1st Qu.:0.1460   1st Qu.:104.1
##  Median :12.30    Median :22.20    Median :0.1950   Median :111.7
##  Mean   :12.27    Mean   :22.17    Mean   :0.2024   Mean   :111.7
##  3rd Qu.:14.50    3rd Qu.:26.80    3rd Qu.:0.2500   3rd Qu.:119.1
##  Max.   :27.80    Max.   :46.30    Max.   :0.5730   Max.   :154.0
##
##             player          player_id        starter              mp
##  Duncan Robinson:  120   grantje01:  120   Min.   :0.0000   0:00    : 6623
##  Jerami Grant   :  120   robindu01:  120   1st Qu.:0.0000   12:00   :   83
##  Bam Adebayo    :  119   adebaba01:  119   Median :0.0000   34:59:00:   33
##  Brad Wanamaker :  119   olynyke01:  119   Mean   :0.3919   24:02:00:   31
##  Kelly Olynyk   :  119   wanambr01:  119   3rd Qu.:1.0000   25:54:00:   30
##  Jae Crowder    :  116   crowdja01:  116   Max.   :1.0000   30:03:00:   30
##  (Other)        :38196   (Other)  :38196                    (Other) :32079
##        fg              fga             fg_pct            fg3
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 1.000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median : 2.000   Median : 6.000   Median :0.3850   Median : 0.0000
##  Mean   : 3.195   Mean   : 6.938   Mean   :0.3549   Mean   : 0.9762
##  3rd Qu.: 5.000   3rd Qu.:11.000   3rd Qu.:0.5330   3rd Qu.: 2.0000
##  Max.   :20.000   Max.   :41.000   Max.   :1.0000   Max.   :13.0000
##
##       fg3a           fg3_pct             ft              fta
##  Min.   : 0.000   Min.   :0.0000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 2.000   Median :0.0000   Median : 0.000   Median : 0.000
##  Mean   : 2.702   Mean   :0.2164   Mean   : 1.387   Mean   : 1.791
##  3rd Qu.: 4.000   3rd Qu.:0.4000   3rd Qu.: 2.000   3rd Qu.: 2.000
##  Max.   :22.000   Max.   :1.0000   Max.   :26.000   Max.   :27.000
##
##      ft_pct            orb              drb              trb
##  Min.   :0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.0000   Median : 2.000   Median : 3.000
##  Mean   :0.3563   Mean   : 0.7817   Mean   : 2.721   Mean   : 3.503
##  3rd Qu.:0.8000   3rd Qu.: 1.0000   3rd Qu.: 4.000   3rd Qu.: 5.000
##  Max.   :1.0000   Max.   :12.0000   Max.   :19.000   Max.   :26.000
##
##       ast              stl              blk              tov
##  Min.   : 0.00   Min.   :0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.00   1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 1.00   Median :0.0000   Median : 0.0000   Median : 1.000
##  Mean   : 1.91   Mean   :0.5951   Mean   : 0.3824   Mean   : 1.082
##  3rd Qu.: 3.00   3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 2.000
##  Max.   :19.00   Max.   :8.0000   Max.   :10.0000   Max.   :11.000
##
##        pf              pts            plus_minus        did_not_play
##  Min.   :0.000   Min.   : 0.000   Min.   :-4.8e+01   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.: 0.000   1st Qu.:-6.0e+00   1st Qu.:0.0000
##  Median :1.000   Median : 7.000   Median : 0.0e+00   Median :0.0000
##  Mean   :1.613   Mean   : 8.753   Mean   : 1.3e-04   Mean   :0.1699
##  3rd Qu.:3.000   3rd Qu.:14.000   3rd Qu.: 5.0e+00   3rd Qu.:0.0000
##  Max.   :6.000   Max.   :62.000   Max.   : 5.0e+01   Max.   :1.0000
```

```
##
##   is_inactive          ts_pct          efg_pct      fg3a_per_fga_pct
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.00000   Median :0.5000   Median :0.444   Median :0.2920
##  Mean   :0.01264   Mean   :0.4356   Mean   :0.409   Mean   :0.3179
##  3rd Qu.:0.00000   3rd Qu.:0.6580   3rd Qu.:0.625   3rd Qu.:0.5000
##  Max.   :1.00000   Max.   :1.5000   Max.   :1.500   Max.   :1.0000
##
##  fta_per_fga_pct     orb_pct           drb_pct          trb_pct
##  Min.   :0.0000   Min.   :  0.000   Min.   :  0.00   Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:  0.000   1st Qu.:  0.00   1st Qu.:  0.000
##  Median :0.0000   Median :  0.000   Median : 10.80   Median :  7.100
##  Mean   :0.2196   Mean   :  3.829   Mean   : 12.75   Mean   :  8.315
##  3rd Qu.:0.3330   3rd Qu.:  5.700   3rd Qu.: 19.70   3rd Qu.: 12.500
##  Max.   :8.0000   Max.   :100.000   Max.   :100.00   Max.   :100.000
##
##     ast_pct           stl_pct           blk_pct          tov_pct
##  Min.   :-528.40   Min.   :  0.000   Min.   : 0.000   Min.   :  0.0
##  1st Qu.:   0.00   1st Qu.:  0.000   1st Qu.: 0.000   1st Qu.:  0.0
##  Median :   7.50   Median :  0.000   Median : 0.000   Median :  6.1
##  Mean   :  11.52   Mean   :  1.228   Mean   : 1.527   Mean   : 10.3
##  3rd Qu.:  18.20   3rd Qu.:  2.000   3rd Qu.: 2.200   3rd Qu.: 16.7
##  Max.   : 100.00   Max.   :100.000   Max.   :77.400   Max.   :100.0
##
##     usg_pct          off_rtg          def_rtg             bpm
##  Min.   :  0.00   Min.   :  0.00   Min.   :-390.00   Min.   :-1000.000
##  1st Qu.:  8.30   1st Qu.: 45.00   1st Qu.:  95.00   1st Qu.:   -5.700
##  Median : 16.30   Median : 99.00   Median : 109.00   Median :    0.000
##  Mean   : 15.75   Mean   : 87.78   Mean   :  92.83   Mean   :   -1.081
##  3rd Qu.: 22.90   3rd Qu.:126.00   3rd Qu.: 119.00   3rd Qu.:    3.300
##  Max.   :100.00   Max.   :300.00   Max.   : 160.00   Max.   :  438.100
##
##     minutes       double_double     triple_double           DKP
##  Min.   : 0.000   Min.   :0.00000   Min.   :0.000000   Min.   :-1.00
##  1st Qu.: 6.767   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.: 3.75
##  Median :20.517   Median :0.00000   Median :0.000000   Median :15.75
##  Mean   :18.948   Mean   :0.07034   Mean   :0.004112   Mean   :18.02
##  3rd Qu.:29.633   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:27.75
##  Max.   :54.233   Max.   :1.00000   Max.   :1.000000   Max.   :92.75
##
##       FDP              SDP            DKP_per_minute  FDP_per_minute
##  Min.   :-2.30   Min.   :-2.25   0           :  918   0      :  926
##  1st Qu.: 3.50   1st Qu.: 3.75   1.071428571:   33   1      :   27
##  Median :15.50   Median :16.25   0.9375      :   31   1.2    :   24
##  Mean   :17.67   Mean   :18.45   0.789473684:   29   1.5    :   24
##  3rd Qu.:27.50   3rd Qu.:28.75   1           :   29   0.75   :   23
##  Max.   :88.80   Max.   :95.50   (Other)     :31247   (Other):31263
##                                  NA's        : 6622   NA's   : 6622
##      SDP_per_minute  pf_per_minute          ts              PG.
##  0           :  926   Min.   : 0.000   Min.   : 0.000   Min.   :  0.00
##  0.789473684:   29   1st Qu.: 0.034   1st Qu.: 1.880   1st Qu.:  0.00
##  1.071428571:   29   Median : 0.078   Median : 6.440   Median :  0.00
##  0.681818182:   28   Mean   : 0.094   Mean   : 7.727   Mean   : 18.57
```

```
##  0.9375     :   28   3rd Qu.: 0.127   3rd Qu.:11.880   3rd Qu.: 19.00
##  (Other)    :31247   Max.   :20.000   Max.   :47.560   Max.   :100.00
##  NA's       : 6622   NA's   :6623                      NA's   :110
##       SG.              SF.              PF.               C.
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Median :  0.00   Median :  2.00   Median :  1.00   Median :  0.00
##  Mean   : 18.76   Mean   : 19.81   Mean   : 20.34   Mean   : 22.62
##  3rd Qu.: 34.00   3rd Qu.: 35.00   3rd Qu.: 33.00   3rd Qu.: 26.00
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00
##  NA's   :110      NA's   :110      NA's   :110      NA's   :110
##  active_position_minutes
##  Min.   : 5.527
##  1st Qu.:45.921
##  Median :49.898
##  Mean   :49.446
##  3rd Qu.:53.827
##  Max.   :84.934
##  NA's   :6848
```

There are a lot of variates in this data, but I don't think that we will be needing all of them. The first step is to think about what the goal of this report is and from there I can find out what variates I will and will not need. I want to find a way with this data to estimate a player's value offensively and defensively relative to their position. I also want to show at what point (in terms of games/minutes played) one would be able to conclude that Player A is better than Player B if I use my calculated offensive and defensive metrics.

The first thing to note is that this data includes a variate "plus_minus" which is equivalent to: $NumberOfPointsScoredByTeamWhilePlayerIsOnTheCourt - NumberOfPointsScoredAgainstTeamWhilePlayerIsOn$ This variate is a nice way of summarizing a player's value because to win a game of basketball you need to score more than the opposing team and this allows me to condense my player evaluation into one stat instead of multiple. So in my models I will be using "plus_minus" as my response variate. But, I do not want the response variate to be influenced by minutes played by a player so I will divide "plus_minus" by "minutes" to get my response variate and I will do the same thing with my explanatory variates as well to account for these differences in play time.

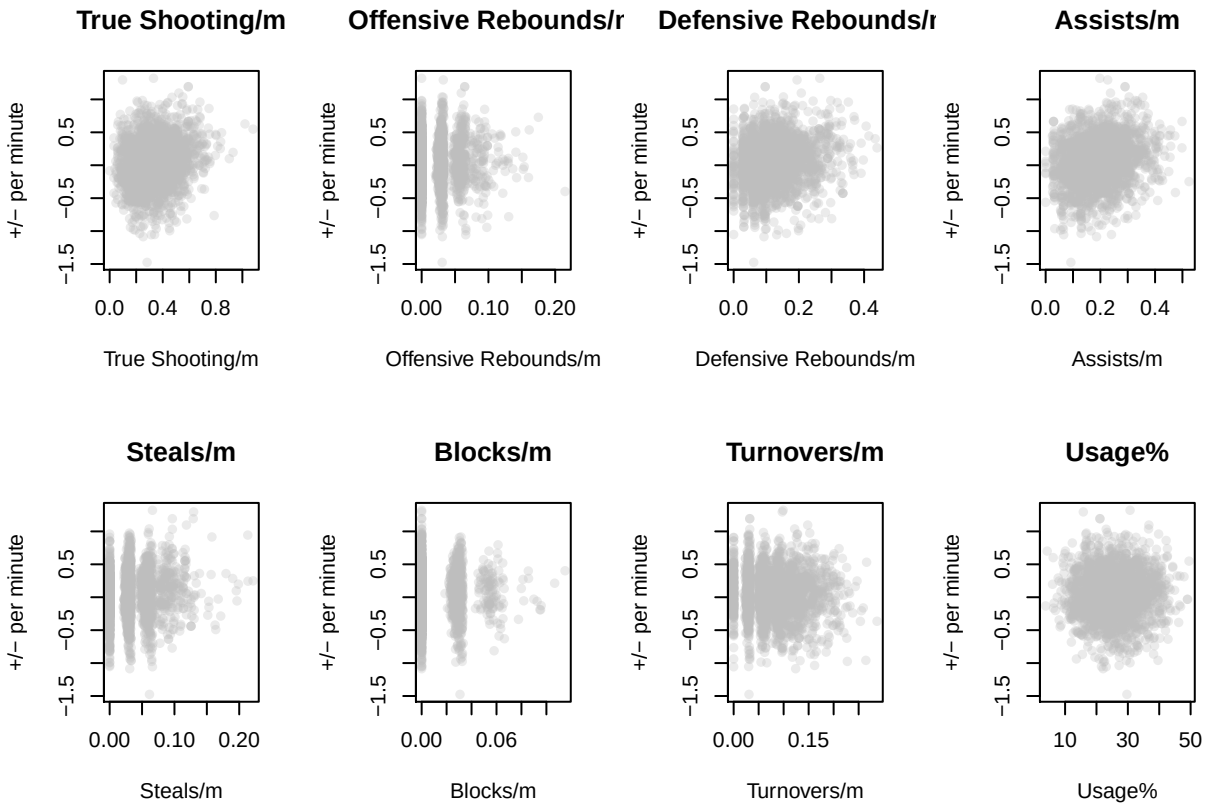
Another thing to note with the model is that many of the explanatory variates' coefficients will be further influenced by the position that a player is playing. For example, getting a rebound as a point guard might not be as critical to winning as getting a rebound as a center. To combat this I'm going to start by adding a factor variate that has the most played position by a player in the given game. This will allow me to model based on a player's position as well:

```
playeddata <- subset(rawdata, minutes > 0) #subsetting for only players that played that game
playeddata <- subset(playeddata, !is.na(PG.)) #getting rid of rows with missing data
playeddata$Pos <- "" #initializing the Pos variable
positions <- c("PG", "SG", "SF", "PF", "C") #all possible values for Pos
for (i in 1:nrow(playeddata)) { #iterate through every row of playeddata
  playeddata$Pos[i] <- positions[which.max(playeddata[i, c("PG.", "SG.", "SF.", "PF.", "C.")])]
  #the column that has the largest value in it will be returned and then that position
  #will be assigned to Pos[i]. Note that if two positions are tied it will pick the
  #first value, so if PG. = 0.5 and SG. = 0.5, playeddata$Pos[i] = "PG"
}
playeddata$Pos <- as.factor(playeddata$Pos) #change from character to factor variate
head(playeddata$Pos) #display first 6 values
```

```
## [1] SG PF PG SG C  C
## Levels: C PF PG SF SG
```

The next step is to find the response variates. Now I could just use my knowledge of basketball and pick a few of the variates in the data that I feel would make sense to be related to how good or bad a player is. Instead of that, I'm going to plot several variates against "plus_minus" and visually analyze whether or not they seem to have a relationship of any sort, whether that be linear, quadratic or something else. Before doing that I need to pick the variates I'm going to plot as well as modify them because I need them all to be relative to minutes played. I'm going to make a function that can do this for a given position, let's start by looking at how these variates compare to "plus_minus" for Point Guards:

```
variatePlot <- function(position) {
  posdata <- subset(playeddata, Pos == position) #position only
  posdata <- subset(posdata, minutes > 30) #players who played a significant amount of
  #minutes that game only
  plus_minus_pm <- with(posdata, plus_minus / minutes) #plus_minus per minute
  ts_pm <- with(posdata, ts_pct * (fga + fta) / minutes) #true shooting% is in relation
  #to how many shots a player has taken so I have to multiply it by the Field Goal Attempts
  #and Free Throw Attempts and then Minutes Played. This stat aims to adjust Field Goal% by
  #adding extra value to 3 pointers and also accounts for free throws
  orb_pm <- with(posdata, orb / minutes) #offensive rebounds per minute
  drb_pm <- with(posdata, drb/ minutes) #defensive rebounds per minute
  ast_pm <- with(posdata, ast / minutes) #assists per minute
  stl_pm <- with(posdata, stl / minutes) #steals per minute
  blk_pm <- with(posdata, blk / minutes) #blocks per minute
  tov_pm <- with(posdata, tov / minutes) #turnovers per minute
  usg_pct <- posdata$usg_pct #usage percent which is already not influenced by minutes played

  #Now that there are 8 variates lets plot them!

  par(mfrow = c(2,4)) #setting up the plots to show 8 at a time
  varNames <- c("True Shooting/m", "Offensive Rebounds/m", "Defensive Rebounds/m",
                "Assists/m", "Steals/m", "Blocks/m", "Turnovers/m", "Usage%")
  vars <- data.frame(ts_pm, orb_pm, drb_pm, ast_pm, stl_pm, blk_pm, tov_pm, usg_pct)
  for (i in 1:8) {
    plot(vars[,i], plus_minus_pm, pch = 16, col = adjustcolor(col = "grey", alpha = 0.3),
         xlab = varNames[i], ylab = "+/- per minute",
         main = varNames[i])
  }
}

variatePlot("PG")
```



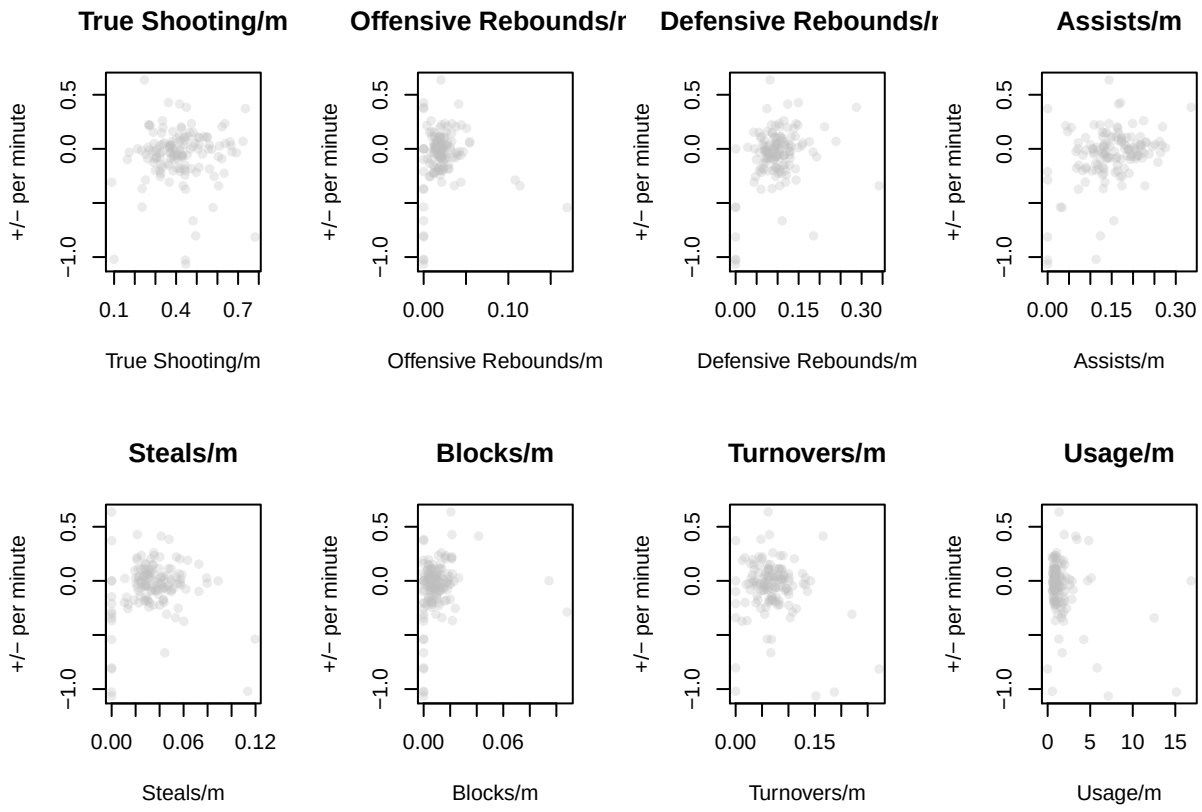
From these plots there is evidence, albeit not a significant amount, that there is a relationship between +/- and the variates: True Shooting, Defensive Rebounds, Assists, Steals and Turnovers. While the variates: Offensive Rebounds, Blocks and Usage% don't seem to have a relationship. However, it doesn't seem to be very obvious how to characterize these relationships and it seems as though there'd be a lot of error in the model because most of these plots have a lot of variance in the +/- for the same response variate value. I believe that a large reason for this variance is that the data used is individual game data from a player instead of a player's averages over the time frame for example. So, I'm going to create a function that is very similar to variatePlot, but this function will use a player's averages over the course of the games they've played in the available data.

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.6.3
```

```
variatePlot2 <- function(position) {
  posdata <- subset(playeddata, Pos == position) #position only
  posdata <- ddply(posdata, .(player, player_id), summarise,
                   plus_minus = sum(plus_minus), minutes = sum(minutes),
                   ts = sum(ts), orb = sum(orb), drb = sum(drb),
                   ast = sum(ast), stl = sum(stl), blk = sum(blk),
                   tov = sum(tov), usg = sum(usg_pct))
  #compiling player totals in the position
  plus_minus_pm <- with(posdata, plus_minus / minutes) #plus_minus per minute
  ts_pm <- with(posdata, ts / minutes) #true shooting per minute
  orb_pm <- with(posdata, orb / minutes) #offensive rebounds per minute
  drb_pm <- with(posdata, drb/ minutes) #defensive rebounds per minute
  ast_pm <- with(posdata, ast / minutes) #assists per minute
```

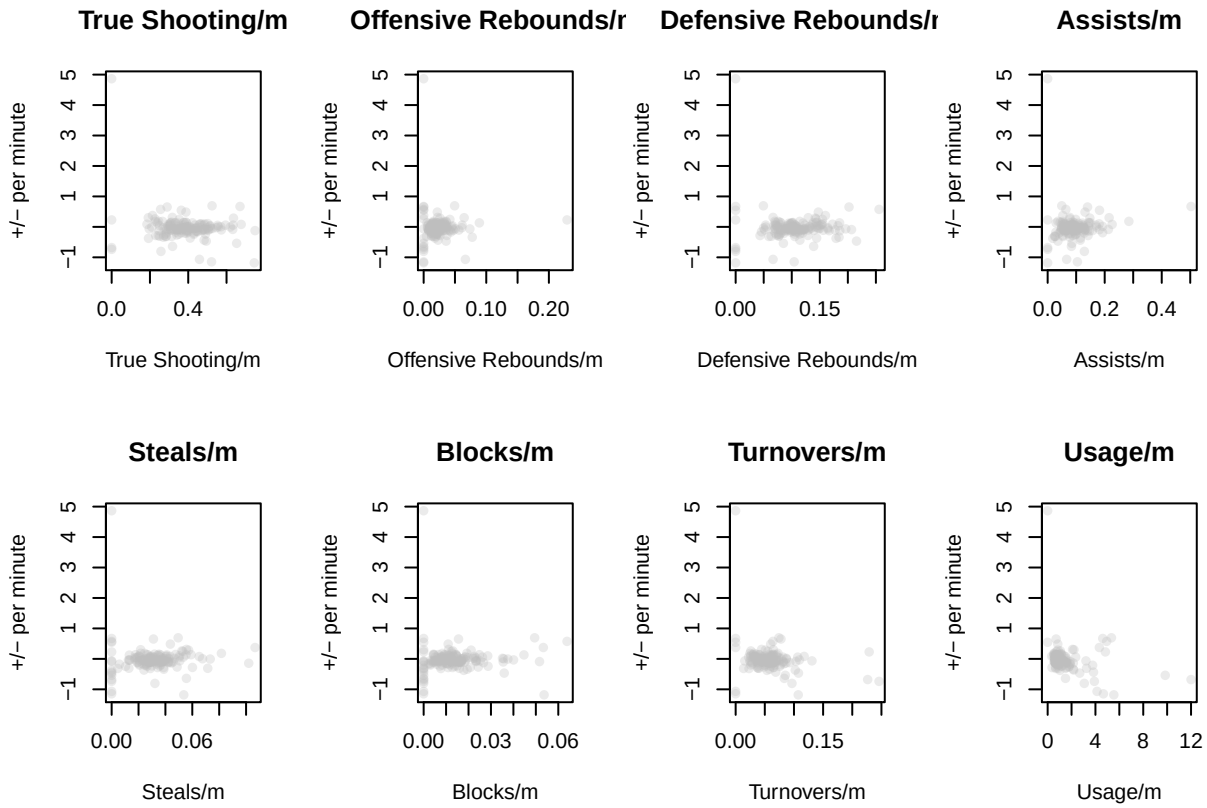
```
  stl_pm <- with(posdata, stl / minutes) #steals per minute
  blk_pm <- with(posdata, blk / minutes) #blocks per minute
  tov_pm <- with(posdata, tov / minutes) #turnovers per minute
  usg_pm <- with(posdata, usg / minutes) #usage per minute

  #Now that there are 8 variates lets plot them!

  par(mfrow = c(2,4)) #setting up the plots to show 8 at a time
  varNames <- c("True Shooting/m", "Offensive Rebounds/m", "Defensive Rebounds/m",
                "Assists/m", "Steals/m", "Blocks/m", "Turnovers/m", "Usage/m")
  vars <- data.frame(ts_pm, orb_pm, drb_pm, ast_pm, stl_pm, blk_pm, tov_pm, usg_pm)
  for (i in 1:8) {
    plot(vars[,i], plus_minus_pm, pch = 16, col = adjustcolor(col = "grey", alpha = 0.3),
         xlab = varNames[i], ylab = "+/- per minute",
         main = varNames[i])
  }
}

variatePlot2("PG")
```



Using the results from these plots, I'm going to narrow down the variates that will be used for each position. I'll do this by looking for the variates that seem to have a relationship with +/-. I will also keep note of whether or not the correlation seems to be positive or negative. In the case of a positive correlation, when the variate increases one would expect the +/- to increase as well and the opposite is true for negative correlation, when the variate increases I expect the +/- to decrease. This will be useful when evaluating if the model that I come up with makes sense. For point guards, it seems like the variates that have a relationship are:

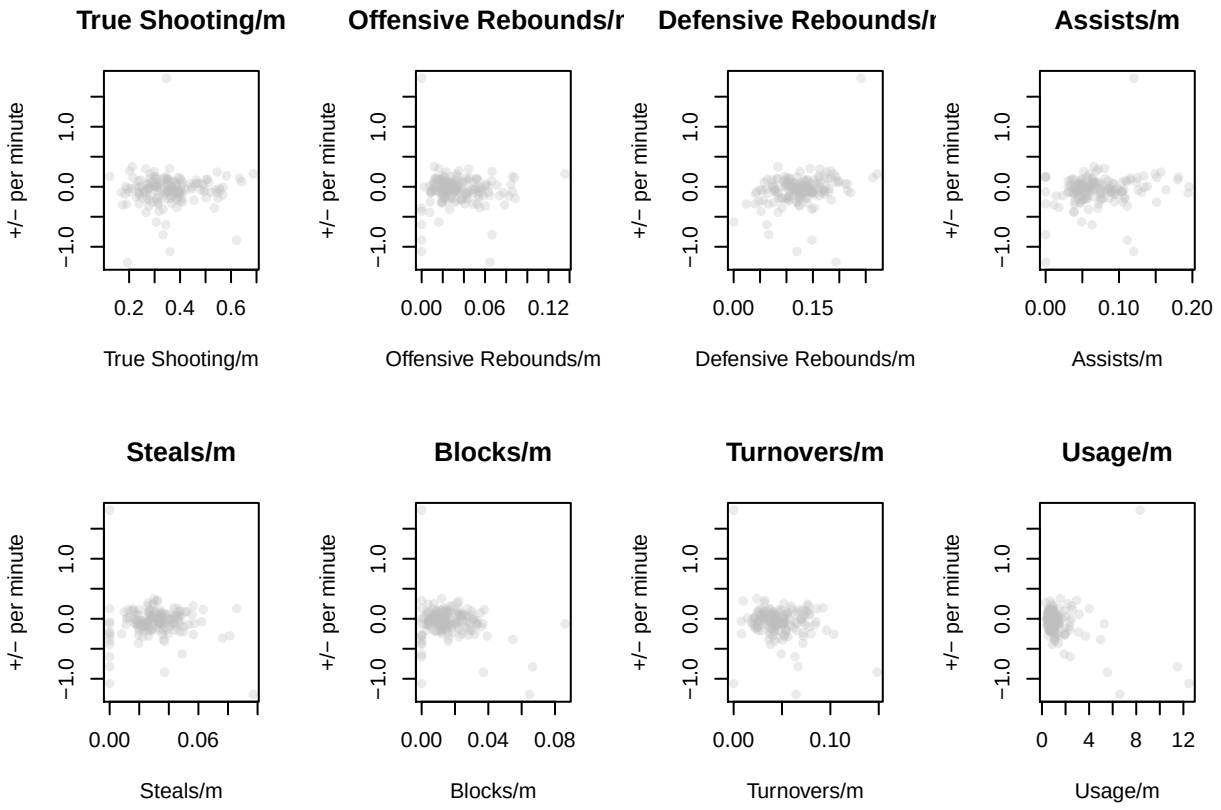
True Shooting/m, Defensive Rebounds/m, Assists/m, Steals/m and Turnovers/m with all of them having seemingly positive correlations aside from Turnovers/m which makes sense as I wouldn't expect turnovers to be an indicator of good play.

```
variatePlot2("SG")
```



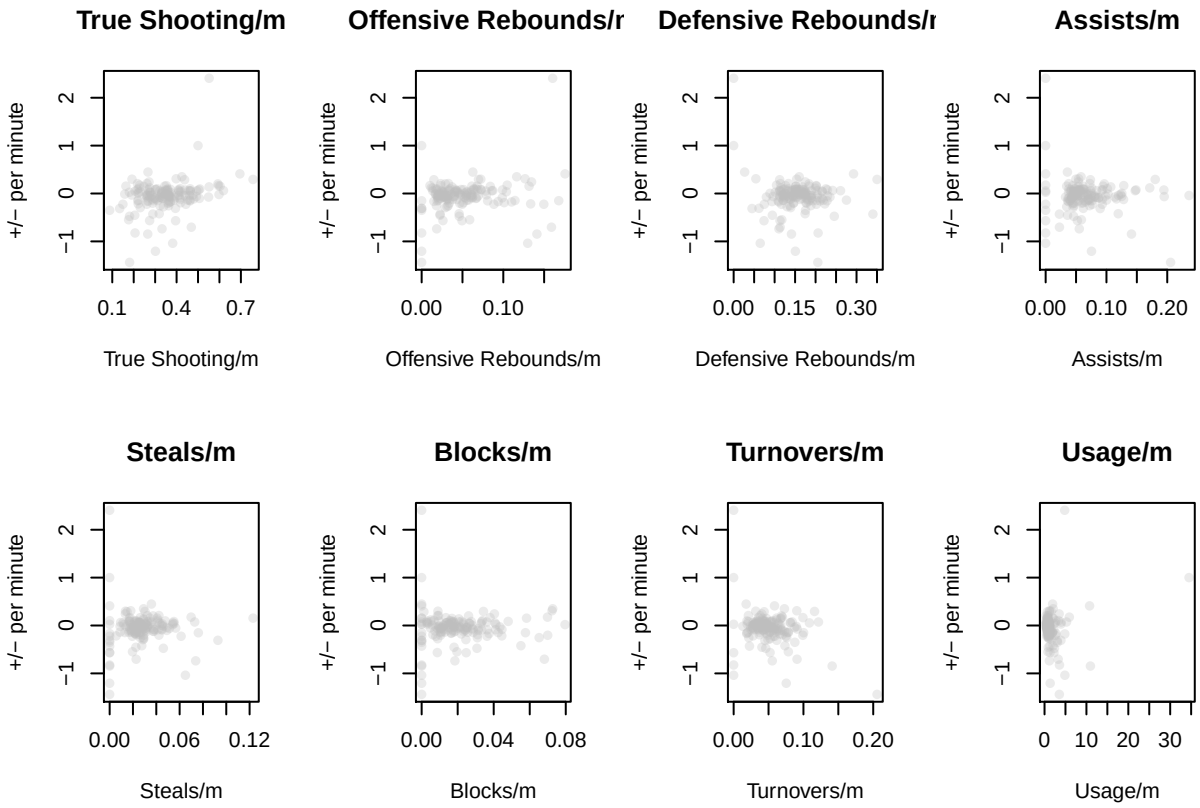
For shooting guards the variates are: True Shooting/m, Defensive Rebounds/m, Assists/m, Steals/m, Blocks/m, Turnovers/m and Usage/m. With all of them having positive correlation except for turnovers again and usage which I guess implies that it's better for a shooting guard to be a catch and shoot type of player with good defense than a second point guard.

```
variatePlot2("SF")
```



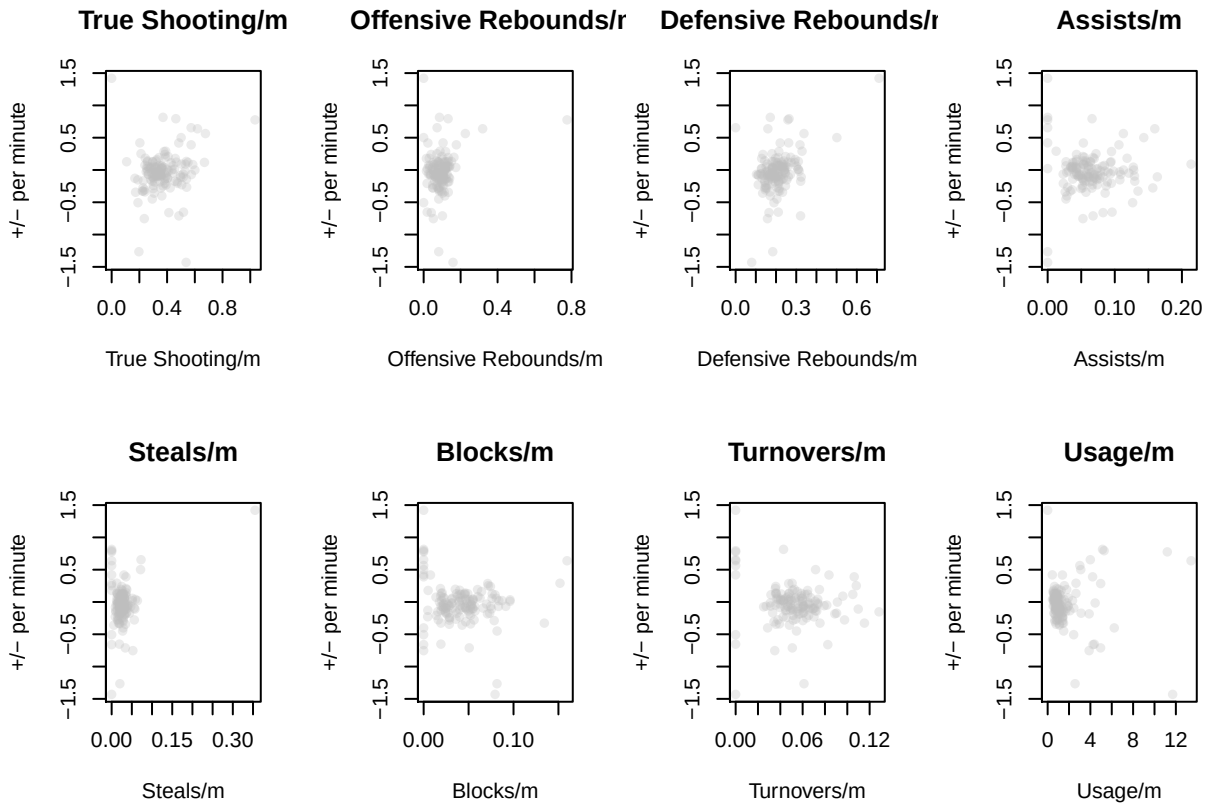
For small forwards the variates are: True Shooting/m, Offensive Rebounds/m, Defensive Rebounds/m, Assists/m and Steals/m. With all of them having positive correlation, the lack of turnovers and usage being present here implies that a small forward's ability to handle the ball and run the court isn't that important which makes sense since they aren't usually in that role and it seems to improve the team's performance when they aren't in that role.

```
variatePlot2("PF")
```



For power forwards the variates are: True Shooting/m, Offensive Rebounds/m, Defensive Rebounds/m and Blocks/m. With all of them having positive correlation, once again I am seeing a lack of ball movement skills being necessary for the role and even the type of defense at the position has changed from others as steals are no longer correlated. This makes sense as big men don't typically get a lot of steals because most of their defense is played in the post, so I'd expect to see a similar trend for centers.

```
variatePlot2("C")
```

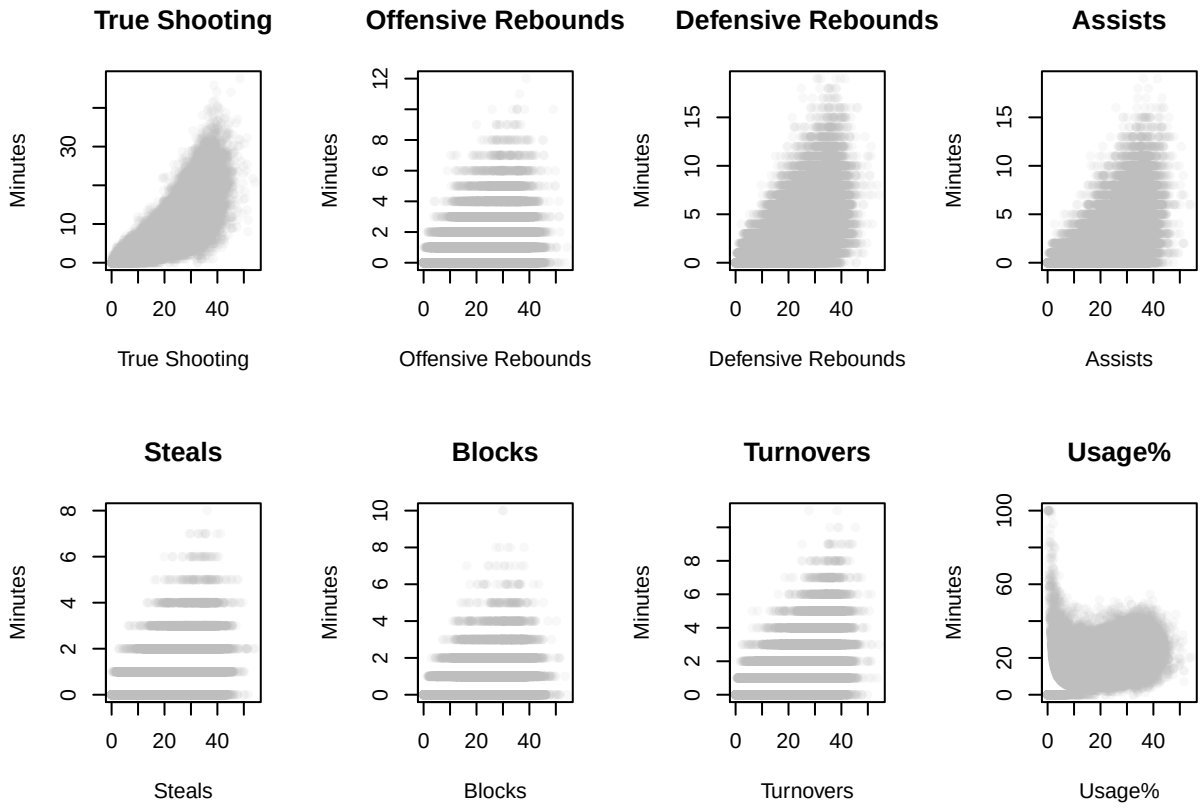


For centers the variates are: True Shooting/m, Offensive Rebounds/m, Defensive Rebounds/m, Assists/m, Blocks/m and Usage/m. With all of them having positive correlation, the trend expected from power forwards didn't quite materialize as expected since assists seem to be an important part of a center's skill set. This could be explained by the progressive increase in 3pt shooting in the league where centers who are good passers can find open shooters beyond the arc, as well as the emerging success of Nikola Jokic that has probably made coaches try to get their centers to play more like him as he's pretty much a point guard in a center's body.

Now that all of the explanatory variates have been selected, I can begin to look into making my model. However, before that because I want to use the data where each player has individual games and that I will be using rates for my variates that are per minute. I have to make sure that all of my explanatory variates have a linear relationship with minutes played and if that isn't the case I will have to only take a subset of my data.

```
mins <- playeddata$minutes
ts <- playeddata$ts
orb <- playeddata$orb
drb <- playeddata$drb
ast <- playeddata$ast
stl <- playeddata$stl
blk <- playeddata$blk
tov <- playeddata$tov
usg_pct <- playeddata$usg_pct
par(mfrow = c(2,4)) #setting up the plots to show 8 at a time
  varNames <- c("True Shooting", "Offensive Rebounds", "Defensive Rebounds",
                "Assists", "Steals", "Blocks", "Turnovers", "Usage%")
  vars <- data.frame(ts, orb, drb, ast, stl, blk, tov, usg_pct)
```

```
for (i in 1:8) {
  plot(mins, vars[,i], pch = 16, col = adjustcolor(col = "grey", alpha = 0.1),
       xlab = varNames[i], ylab = "Minutes",
       main = varNames[i])
}
```



All of these seem to be linear except for Usage, which is expected as this is a percentage and not a raw number and so it will not naturally increase as the game goes on. But even from the usage plot, I can see that after roughly 10-15 minutes it starts to become linear. To be safe I'll set the minimum amount of minutes required for a player to be in the model to 15 minutes played.

Now for the fun part, the model itself. I could simply use multiple linear regression, but that might not be the best idea for this specific case. This is because I'm using individual game data for each of my data points in the model and there are a lot of outlier games in the NBA where a player might not miss a single shot in a game or get 400% more blocks then usual. To combat this, I'm going to use the MASS package's rlm function which will weight different data based on if they are outliers or not. For comparison sake and illustration purposes, I will show how simply doing multiple linear regression differs from using robust regression.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.6.3
```

```
modelCreator <- function(position, variates) {
  min15data <- subset(playeddata, minutes > 15) #subsetting for only players who
                                                #played more than 15 minutes
                                                #that game
  posdata <- subset(min15data, Pos == position) #position only
  posdata$plus_minus_pm <- with(posdata, plus_minus / minutes) #plus_minus per
```

```
                                                                  #minute
  posdata$ts_pm <- with(posdata, ts / minutes) #true shooting per minute
  posdata$orb_pm <- with(posdata, orb / minutes) #offensive rebounds per minute
  posdata$drb_pm <- with(posdata, drb/ minutes) #defensive rebounds per minute
  posdata$ast_pm <- with(posdata, ast / minutes) #assists per minute
  posdata$stl_pm <- with(posdata, stl / minutes) #steals per minute
  posdata$blk_pm <- with(posdata, blk / minutes) #blocks per minute
  posdata$tov_pm <- with(posdata, tov / minutes) #turnovers per minute

  f <- as.formula(paste("plus_minus_pm", paste(variates, collapse = " + "),
                  sep = " ~ ")) #this is the formula that the models will be fit
                                #according to
  multipleModel <- lm(f, data = posdata) #the multiple regression model
  print(summary(multipleModel))
  robustModel <- rlm(f, data = posdata, psi = "psi.bisquare")
  print(summary(robustModel))
}
```

Now to apply this model creating function to all five positions with the variates decided upon for each position:

```
#The chosen variables for each position:
PGvars <- list("ts_pm", "drb_pm", "ast_pm", "stl_pm", "tov_pm")
SGvars <- list("ts_pm", "drb_pm", "ast_pm", "stl_pm", "blk_pm", "tov_pm", "usg_pct")
SFvars <- list("ts_pm", "orb_pm", "drb_pm", "ast_pm", "stl_pm")
PFvars <- list("ts_pm", "orb_pm", "drb_pm", "blk_pm")
Cvars <- list("ts_pm", "orb_pm", "drb_pm", "ast_pm", "blk_pm", "usg_pct")
posVars <- list(PGvars, SGvars, SFvars, PFvars, Cvars)
positions <- c("PG", "SG", "SF", "PF", "C") #list of positions
for (i in 1:5) {
  cat("Models for ", positions[i], "\n\n")
  pos.list <- rep("", length(posVars[[i]])) #create a blank list of the length
                                            #of the number of variates for that
                                            #position
  for (j in 1:length(posVars[[i]])) {
    pos.list[j] <- posVars[[i]][j]
  }
  modelCreator(positions[i], pos.list)
  cat("\n\n")
}
```

```
## Models for  PG
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65986 -0.28752 -0.00384  0.29267  1.66330
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.21304    0.02264  -9.410  < 2e-16 ***
## ts_pm        0.05723    0.04011   1.427    0.154
```

```
## drb_pm       0.88181    0.08950   9.853  < 2e-16 ***
## ast_pm       1.00156    0.06986  14.337  < 2e-16 ***
## stl_pm       1.17280    0.16289   7.200 6.94e-13 ***
## tov_pm      -1.41215    0.12115 -11.657  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4552 on 4902 degrees of freedom
## Multiple R-squared:  0.09041,    Adjusted R-squared:  0.08948
## F-statistic: 97.44 on 5 and 4902 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.666452 -0.287559 -0.005775  0.290070  1.661369
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.2049   0.0226    -9.0743
## ts_pm         0.0642   0.0400     1.6053
## drb_pm        0.8810   0.0893     9.8705
## ast_pm        0.9730   0.0697    13.9663
## stl_pm        1.0400   0.1624     6.4021
## tov_pm       -1.4115   0.1208   -11.6830
##
## Residual standard error: 0.429 on 4902 degrees of freedom
##
##
## Models for  SG
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7006 -0.2810  0.0074  0.2884  1.8111
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.184942   0.021643  -8.545  < 2e-16 ***
## ts_pm       -0.103529   0.198055  -0.523  0.60119
## drb_pm       0.985385   0.089508  11.009  < 2e-16 ***
## ast_pm       0.826033   0.086207   9.582  < 2e-16 ***
## stl_pm       1.261006   0.170194   7.409 1.48e-13 ***
## blk_pm       0.839234   0.271157   3.095  0.00198 **
## tov_pm      -1.000201   0.232032  -4.311 1.66e-05 ***
## usg_pct      0.002007   0.004625   0.434  0.66435
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4517 on 5038 degrees of freedom
## Multiple R-squared:  0.06107,    Adjusted R-squared:  0.05977
```

```
## F-statistic: 46.81 on 7 and 5038 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.695175 -0.281276  0.006592  0.286506  1.829799
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -0.1667  0.0216    -7.7165
## ts_pm       -0.3557  0.1977    -1.7995
## drb_pm       0.9328  0.0893    10.4416
## ast_pm       0.7813  0.0860     9.0814
## stl_pm       1.2170  0.1699     7.1650
## blk_pm       0.7510  0.2706     2.7751
## tov_pm      -1.0999  0.2316    -4.7499
## usg_pct      0.0073  0.0046     1.5728
##
## Residual standard error: 0.4215 on 5038 degrees of freedom
##
##
## Models for  SF
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.93425 -0.30011  0.00412  0.29586  1.91281
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.20527    0.02085  -9.845  < 2e-16 ***
## ts_pm        0.04757    0.04372   1.088   0.2766
## orb_pm      -0.36602    0.17252  -2.122   0.0339 *
## drb_pm       0.79554    0.08495   9.365  < 2e-16 ***
## ast_pm       0.91939    0.09984   9.208  < 2e-16 ***
## stl_pm       0.94585    0.17545   5.391 7.33e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4713 on 5029 degrees of freedom
## Multiple R-squared:  0.04678,    Adjusted R-squared:  0.04583
## F-statistic: 49.36 on 5 and 5029 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.930691 -0.299743  0.004999  0.295475  1.904800
##
## Coefficients:
```

```
##             Value   Std. Error t value
## (Intercept) -0.1941  0.0209    -9.2995
## ts_pm        0.0183  0.0438     0.4185
## orb_pm      -0.3517  0.1727    -2.0370
## drb_pm       0.8123  0.0850     9.5533
## ast_pm       0.8884  0.0999     8.8900
## stl_pm       0.9463  0.1756     5.3886
##
## Residual standard error: 0.4427 on 5029 degrees of freedom
##
##
## Models for  PF
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.68665 -0.29524  0.00875  0.30378  1.92676
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.18122    0.01980  -9.154  < 2e-16 ***
## ts_pm        0.16776    0.04313   3.890 0.000102 ***
## orb_pm       0.19356    0.13904   1.392 0.163954
## drb_pm       0.63336    0.07326   8.645  < 2e-16 ***
## blk_pm       0.73565    0.21886   3.361 0.000782 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4753 on 5037 degrees of freedom
## Multiple R-squared:  0.0266, Adjusted R-squared:  0.02583
## F-statistic: 34.41 on 4 and 5037 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.688534 -0.298378  0.007005  0.300919  1.924602
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -0.1746  0.0199    -8.7973
## ts_pm        0.1677  0.0433     3.8772
## orb_pm       0.0998  0.1394     0.7161
## drb_pm       0.6338  0.0735     8.6269
## blk_pm       0.7431  0.2195     3.3860
##
## Residual standard error: 0.4444 on 5037 degrees of freedom
##
##
## Models for  C
##
```

```
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83600 -0.29579  0.00175  0.30605  1.78796
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.275138   0.024978 -11.015  < 2e-16 ***
## ts_pm        0.720124   0.131124   5.492 4.19e-08 ***
## orb_pm      -0.048989   0.101469  -0.483    0.629
## drb_pm       0.919636   0.066527  13.823  < 2e-16 ***
## ast_pm       0.759915   0.109216   6.958 3.94e-12 ***
## blk_pm       1.417196   0.139907  10.130  < 2e-16 ***
## usg_pct     -0.015387   0.002867  -5.367 8.39e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4776 on 4572 degrees of freedom
## Multiple R-squared:  0.07746,    Adjusted R-squared:  0.07625
## F-statistic: 63.98 on 6 and 4572 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.843554 -0.299898 -0.003963  0.299352  1.778407
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.2638   0.0249   -10.5862
## ts_pm         0.7302   0.1308     5.5821
## orb_pm       -0.0423   0.1012    -0.4181
## drb_pm        0.8769   0.0664    13.2122
## ast_pm        0.7248   0.1090     6.6524
## blk_pm        1.4520   0.1396    10.4028
## usg_pct      -0.0154   0.0029    -5.3886
##
## Residual standard error: 0.4445 on 4572 degrees of freedom
```

The first thing to check before moving on from these models is that the estimated values make sense. Now it might not be very clear what making sense means, but essentially I want to make sure that the variates that in my opinion should be positively contributing to a player's +/- are actually given positive estimates in the models. If there are examples where that isn't the case that may be due to the variate not having a relationship with +/- or it could be due to an accounted for trend with the NBA that I will have to find a way to account for. One example of one of these trends from the MLB is that if one were to make a model to predict Home Runs and included a response variate related to a player's speed then they might find that being fast is bad for hitting home runs. While that is technically the case with the data, it isn't actually a negative thing to be fast it's just that many baseball players have very different builds: some are large and hit a lot of home runs and some are more thin and very fast. If I didn't account for this in my model then players who are fast would be punished even though being fast and able to hit home runs are not mutually exclusive.

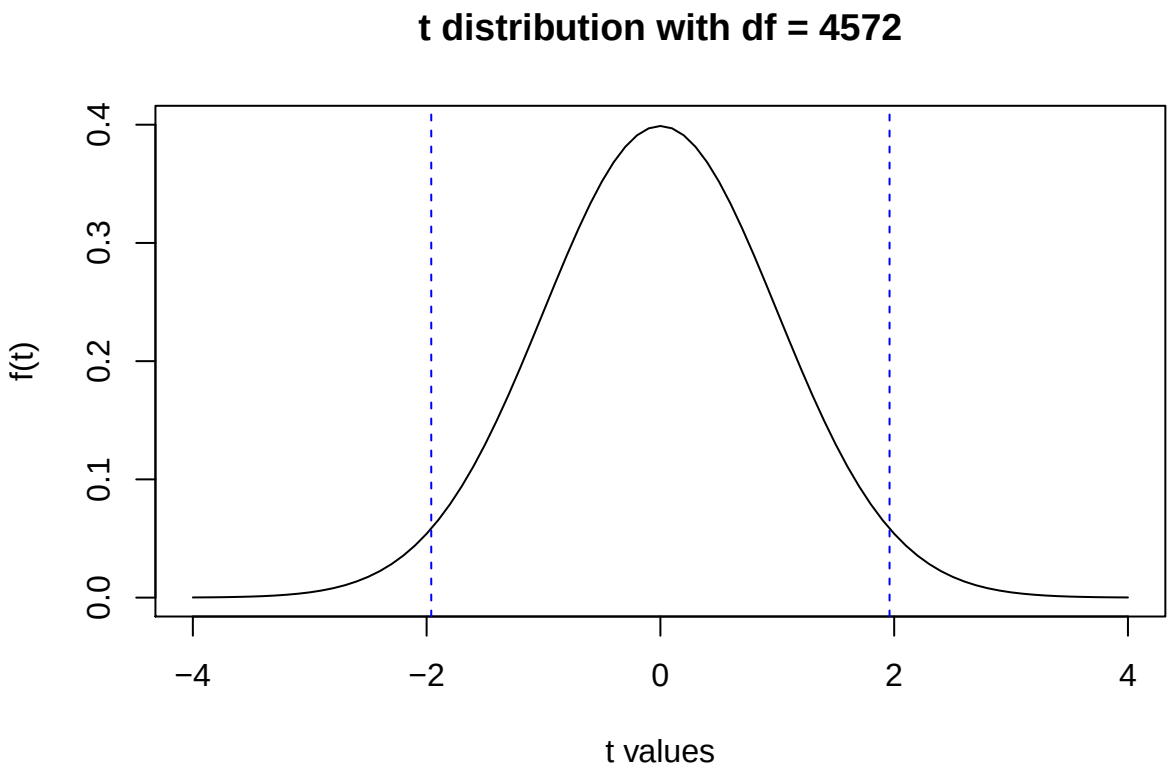
So, the next step is to find if there are any variates that should be removed from any of the models for the

different positions. The I will do this is if the t value beside the variable, if the absolute value of the t-value of for that variable is less than 1.960483 then it falls under the null hypothesis $H_0 : variate = 0$ where there is no relationship. The value of 1.960483 is calculated assuming that the significance level is $\alpha = 0.05$ and that the t distribution has degrees of freedom equivalent to that of the position with the fewest qualifying games (Center). Even though this isn't technically fair to use for the other positions which have differing degrees of freedom due to the degrees of freedom being so high it won't make a significant difference in the final obtained value. The value is calculated by the following R code:

```
alpha = 0.05 #significance level
df = 4572 #degrees of freedom of Centers data
tval <- qt(1 - alpha/2, df)
print(tval)
```

```
## [1] 1.960483
```

```
t.values <- seq(-4,4,.1)
plot(x = t.values, y = dt(t.values, df), type = "l", ylim = c(0,.4),
     xlab = "t values", ylab = "f(t)", main = "t distribution with df = 4572")
abline(v = -tval, lty = 2, col = "blue")
abline(v = tval, lty = 2, col = "blue")
```



Now to find remove all explanatory variates from the models where $|t_value| < 1.960483$; for the point guards, I remove only True shooting, for the shooting guards, the variates True Shooting and Usage% are removed, True Shooting is once again removed for small forwards and for power forwards and centers the only variates removed are Offensive Rebounds. Removing those variates and rerunning the model I get the following:

```
#The adjusted variables for each position:
PGvars <- list("drb_pm", "ast_pm", "stl_pm", "tov_pm")
```

```r
SGvars <- list("drb_pm", "ast_pm", "stl_pm", "blk_pm", "tov_pm")
SFvars <- list("orb_pm", "drb_pm", "ast_pm", "stl_pm")
PFvars <- list("ts_pm", "drb_pm", "blk_pm")
Cvars <- list("ts_pm", "drb_pm", "ast_pm", "blk_pm", "usg_pct")
posVars <- list(PGvars, SGvars, SFvars, PFvars, Cvars)
positions <- c("PG", "SG", "SF", "PF", "C") #list of positions
for (i in 1:5) {
  cat("Models for ", positions[i], "\n\n")
  pos.list <- rep("", length(posVars[[i]])) #create a blank list of the length
                                            #of the number of variates for that
                                            #position
  for (j in 1:length(posVars[[i]])) {
    pos.list[j] <- posVars[[i]][j]
  }
  modelCreator(positions[i], pos.list)
  cat("\n\n")
}
```

```
## Models for  PG
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.66330 -0.28962 -0.00511  0.29125  1.67027
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.19200    0.01718 -11.173  < 2e-16 ***
## drb_pm       0.89264    0.08919  10.009  < 2e-16 ***
## ast_pm       1.01153    0.06951  14.551  < 2e-16 ***
## stl_pm       1.15948    0.16264   7.129 1.16e-12 ***
## tov_pm      -1.38420    0.11956 -11.577  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4552 on 4903 degrees of freedom
## Multiple R-squared:  0.09003,    Adjusted R-squared:  0.08929
## F-statistic: 121.3 on 4 and 4903 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.670503 -0.288443 -0.005801  0.290701  1.669119
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.1812   0.0171   -10.5731
## drb_pm        0.8938   0.0890    10.0470
## ast_pm        0.9849   0.0693    14.2048
## stl_pm        1.0234   0.1622     6.3085
```

```
## tov_pm       -1.3805   0.1193   -11.5755
##
## Residual standard error: 0.4293 on 4903 degrees of freedom
##
##
## Models for  SG
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.70439 -0.28293  0.00792  0.28818  1.80816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.19053    0.01578 -12.073  < 2e-16 ***
## drb_pm       0.98377    0.08946  10.997  < 2e-16 ***
## ast_pm       0.82076    0.08535   9.616  < 2e-16 ***
## stl_pm       1.25597    0.16989   7.393 1.67e-13 ***
## blk_pm       0.84090    0.27088   3.104  0.00192 **
## tov_pm      -0.93002    0.12945  -7.184 7.74e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4517 on 5040 degrees of freedom
## Multiple R-squared:  0.06099,    Adjusted R-squared:  0.06006
## F-statistic: 65.48 on 5 and 5040 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.706436 -0.281825  0.007085  0.288173  1.818649
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.1806   0.0158   -11.4571
## drb_pm        0.9278   0.0893    10.3854
## ast_pm        0.7707   0.0852     9.0410
## stl_pm        1.2025   0.1697     7.0874
## blk_pm        0.7540   0.2705     2.7870
## tov_pm       -0.8348   0.1293    -6.4572
##
## Residual standard error: 0.4229 on 5040 degrees of freedom
##
##
## Models for  SF
##
##
## Call:
## lm(formula = f, data = posdata)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.92375 -0.29832  0.00514  0.29551  1.90386
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.19005    0.01546 -12.295  < 2e-16 ***
## orb_pm      -0.35275    0.17209  -2.050   0.0404 *
## drb_pm       0.80227    0.08472   9.469  < 2e-16 ***
## ast_pm       0.93734    0.09847   9.519  < 2e-16 ***
## stl_pm       0.94988    0.17542   5.415 6.42e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4713 on 5030 degrees of freedom
## Multiple R-squared:  0.04655,    Adjusted R-squared:  0.0458
## F-statistic:  61.4 on 4 and 5030 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.926604 -0.298885  0.004969  0.294790  1.901473
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.1883   0.0155   -12.1682
## orb_pm       -0.3472   0.1722    -2.0158
## drb_pm        0.8156   0.0848     9.6177
## ast_pm        0.8953   0.0986     9.0839
## stl_pm        0.9468   0.1756     5.3923
##
## Residual standard error: 0.4411 on 5030 degrees of freedom
##
##
## Models for  PF
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.68587 -0.29854  0.01032  0.30327  1.92335
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.17836    0.01969  -9.058  < 2e-16 ***
## ts_pm        0.17864    0.04242   4.211 2.58e-05 ***
## drb_pm       0.64112    0.07306   8.776  < 2e-16 ***
## blk_pm       0.75426    0.21847   3.452  0.00056 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.4754 on 5038 degrees of freedom
## Multiple R-squared:  0.02623,    Adjusted R-squared:  0.02565
## F-statistic: 45.23 on 3 and 5038 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.687985 -0.301643  0.007861  0.300510  1.922894
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -0.1731  0.0197    -8.7731
## ts_pm        0.1729  0.0425     4.0676
## drb_pm       0.6384  0.0732     8.7204
## blk_pm       0.7531  0.2189     3.4396
##
## Residual standard error: 0.4466 on 5038 degrees of freedom
##
##
## Models for  C
##
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84239 -0.29534  0.00145  0.30617  1.78932
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.277898   0.024313 -11.430  < 2e-16 ***
## ts_pm        0.713072   0.130297   5.473 4.67e-08 ***
## drb_pm       0.916154   0.066130  13.854  < 2e-16 ***
## ast_pm       0.762878   0.109034   6.997 3.00e-12 ***
## blk_pm       1.416910   0.139894  10.128  < 2e-16 ***
## usg_pct     -0.015303   0.002861  -5.348 9.32e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4776 on 4573 degrees of freedom
## Multiple R-squared:  0.07741,    Adjusted R-squared:  0.07641
## F-statistic: 76.74 on 5 and 4573 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.849274 -0.299541 -0.003982  0.298771  1.779540
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -0.2662   0.0243   -10.9721
```

```
## ts_pm         0.7243   0.1300     5.5717
## drb_pm        0.8733   0.0660    13.2354
## ast_pm        0.7276   0.1088     6.6883
## blk_pm        1.4514   0.1396    10.3985
## usg_pct      -0.0153   0.0029    -5.3715
##
## Residual standard error: 0.4437 on 4573 degrees of freedom
```

Now for the final part of the process of creating the model, making sure that the estimates "make sense." As I explained earlier there is no objective way to do this and I don't want my own biases to affect things too much but I believe that all of the variates except for Turnovers and Usage% should have positive estimates with Usage% having the possibility of being either, so if I find an exception in any of these variates then I will once again remove that variate from the model. Looking at the estimates, the only case where the estimate doesn't follow the rule set up above is for the small forwards model which has a negative estimate for Offensive Rebounds. The interesting and comforting finding to note is that the t value for this variate is very close to the boundary set from the previous step, so it is more likely that this estimate isn't reliable because the t value is so close to the cutoff point.

Finally, removing that on variate for small forwards, the small forward model is:

```
#The final variables for each small forwards:
SFvars <- list("drb_pm", "ast_pm", "stl_pm")
posVars <- list(PGvars, SGvars, SFvars, PFvars, Cvars)
cat("Models for SF \n\n")
```

```
## Models for SF
```

```
modelCreator("SF", SFvars)
```

```
##
## Call:
## lm(formula = f, data = posdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9341 -0.3003  0.0072  0.2943  1.8746
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.19891    0.01485 -13.398  < 2e-16 ***
## drb_pm       0.78939    0.08452   9.340  < 2e-16 ***
## ast_pm       0.94115    0.09849   9.556  < 2e-16 ***
## stl_pm       0.93194    0.17526   5.318  1.1e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4714 on 5031 degrees of freedom
## Multiple R-squared:  0.04576,    Adjusted R-squared:  0.04519
## F-statistic: 80.41 on 3 and 5031 DF,  p-value: < 2.2e-16
##
##
## Call: rlm(formula = f, data = posdata, psi = "psi.bisquare")
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.937291 -0.300868  0.006815  0.294701  1.871962
##
## Coefficients:
```

```
##             Value    Std. Error t value
## (Intercept)  -0.1963   0.0149   -13.2106
## drb_pm        0.7999   0.0846     9.4562
## ast_pm        0.8961   0.0986     9.0905
## stl_pm        0.9312   0.1754     5.3088
##
## Residual standard error: 0.4412 on 5031 degrees of freedom
```

```
cat("\n\n")
```

Now that the estimates are finalized, I can create estimated plus/minus values for my data, separating by position:

```
#mPM will be the new variable name for the predicted plus/minus values using
#multiple linear regression and rPM for robust regression

xPM <- function(position) {
  tempdata <- subset(playeddata, Pos == position)
  tempdata$ts_pm <- with(tempdata, ts / minutes) #true shooting per minute
  tempdata$orb_pm <- with(tempdata, orb / minutes) #offensive rebounds per minute
  tempdata$drb_pm <- with(tempdata, drb/ minutes) #defensive rebounds per minute
  tempdata$ast_pm <- with(tempdata, ast / minutes) #assists per minute
  tempdata$stl_pm <- with(tempdata, stl / minutes) #steals per minute
  tempdata$blk_pm <- with(tempdata, blk / minutes) #blocks per minute
  tempdata$tov_pm <- with(tempdata, tov / minutes) #turnovers per minute
  tempdata$plus_minus_pm <- with(tempdata, plus_minus / minutes) #+/- per minute
  if (position == "PG") {
    tempdata$mPM <- with(tempdata, -0.19200 + 0.89264*drb_pm
                         + 1.01153*ast_pm + 1.15948*stl_pm
                         - 1.38420*tov_pm)
    tempdata$rPM <- with(tempdata, -0.1812 + 0.8938*drb_pm
                         + 0.9849*ast_pm + 1.0234*stl_pm
                         - 1.3805*tov_pm)
    assign(paste(position, "Data", sep = ""), tempdata[,c("player", "player_id",
                                                          "minutes", "plus_minus_pm",
                                                          "drb_pm", "ast_pm",
                                                          "stl_pm", "tov_pm",
                                                          "mPM", "rPM")],
           envir = .GlobalEnv)
    #Use only the variates that are used in the model or to identify a player
  }
  if (position == "SG") {
    tempdata$mPM <- with(tempdata, -0.19053 + 0.98377*drb_pm
                         + 0.82076*ast_pm + 1.25597*stl_pm
                         + 0.84090*blk_pm - 0.93002*tov_pm)
    tempdata$rPM <- with(tempdata, -0.1806 + 0.9278*drb_pm
                         + 0.7707*ast_pm + 1.2025*stl_pm
                         + 0.7540*blk_pm - 0.8348*tov_pm)
    assign(paste(position, "Data", sep = ""), tempdata[,c("player", "player_id",
                                                          "minutes", "plus_minus_pm",
                                                          "drb_pm", "ast_pm",
                                                          "stl_pm", "blk_pm",
                                                          "tov_pm", "mPM", "rPM")],
           envir = .GlobalEnv)
    #Use only the variates that are used in the model or to identify a player
```

```
  }
  if (position == "SF") {
    tempdata$mPM <- with(tempdata, -0.19891 + 0.78939*drb_pm
                         + 0.94115*ast_pm + 0.93194*stl_pm)
    tempdata$rPM <- with(tempdata, -0.1963 + 0.7999*drb_pm
                         + 0.8961*ast_pm + 0.9312*stl_pm)
    assign(paste(position, "Data", sep = ""), tempdata[,c("player", "player_id",
                                                          "minutes", "plus_minus_pm",
                                                          "drb_pm", "ast_pm",
                                                          "stl_pm", "mPM", "rPM")],
           envir = .GlobalEnv)
    #Use only the variates that are used in the model or to identify a player
  }
  if (position == "PF") {
    tempdata$mPM <- with(tempdata, -0.17836 + 0.17864*ts_pm
                         + 0.64112*drb_pm + 0.75426*blk_pm)
    tempdata$rPM <- with(tempdata, -0.1731 + 0.1729*ts_pm
                         + 0.6384*drb_pm + 0.7531*blk_pm)
    assign(paste(position, "Data", sep = ""), tempdata[,c("player", "player_id",
                                                          "minutes", "plus_minus_pm",
                                                          "ts_pm", "drb_pm",
                                                          "blk_pm", "mPM", "rPM")],
           envir = .GlobalEnv)
    #Use only the variates that are used in the model or to identify a player
  }
  if (position == "C") {
    tempdata$mPM <- with(tempdata, -0.277898 + 0.713072*ts_pm
                         + 0.916154*drb_pm + 0.762878*ast_pm
                         + 1.416910*blk_pm - 0.015303*usg_pct)
    tempdata$rPM <- with(tempdata, -0.2662 + 0.7243*ts_pm
                         + 0.8733*drb_pm + 0.7276*ast_pm
                         + 1.4514*blk_pm - 0.0153*usg_pct)
    assign(paste(position, "Data", sep = ""), tempdata[,c("player", "player_id",
                                                          "minutes", "plus_minus_pm",
                                                          "ts_pm", "drb_pm",
                                                          "ast_pm", "blk_pm",
                                                          "usg_pct", "mPM", "rPM")],
           envir = .GlobalEnv)
    #Use only the variates that are used in the model or to identify a player
  }
}

for (i in 1:5) { #apply function to all 5 positions
  xPM(positions[i])
}
```

Now that that those results have been compiled, let's take a look at how the estimated values of +/- compare to the actual ones observed from a general population standpoint for each position.

```
vars <- c("plus_minus_pm", "mPM", "rPM")
summary(PGData[,vars])
```

```
##  plus_minus_pm            mPM                 rPM
##  Min.   :-30.00000   Min.   :-2.634706   Min.   :-2.617377
```

```
##  1st Qu.: -0.34522   1st Qu.:-0.094791   1st Qu.:-0.089951
##  Median :  0.00000   Median : 0.006355   Median : 0.009333
##  Mean   : -0.03494   Mean   : 0.010795   Mean   : 0.012802
##  3rd Qu.:  0.33172   3rd Qu.: 0.111566   3rd Qu.: 0.110388
##  Max.   : 18.46154   Max.   : 2.235672   Max.   : 2.182560
```

```
summary(SGData[,vars])
```

```
##  plus_minus_pm            mPM                 rPM
##  Min.   :-60.00000   Min.   :-1.205097   Min.   :-1.091291
##  1st Qu.: -0.36697   1st Qu.:-0.097383   1st Qu.:-0.090654
##  Median :  0.00000   Median :-0.014979   Median :-0.013387
##  Mean   : -0.07106   Mean   :-0.003721   Mean   :-0.002242
##  3rd Qu.:  0.29944   3rd Qu.: 0.070487   3rd Qu.: 0.067349
##  Max.   : 30.00000   Max.   : 9.394378   Max.   : 8.926477
```

```
summary(SFData[,vars])
```

```
##  plus_minus_pm            mPM                 rPM
##  Min.   :-40.00000   Min.   :-0.198910   Min.   :-0.196300
##  1st Qu.: -0.38415   1st Qu.:-0.084785   1st Qu.:-0.083237
##  Median :  0.00000   Median :-0.016750   Median :-0.015873
##  Mean   : -0.08289   Mean   :-0.001777   Mean   :-0.001181
##  3rd Qu.:  0.32591   3rd Qu.: 0.064783   3rd Qu.: 0.064981
##  Max.   :  7.20000   Max.   : 2.587172   Max.   : 2.626876
```

```
summary(PFData[,vars])
```

```
##  plus_minus_pm           mPM                 rPM
##  Min.   :-6.92308   Min.   :-0.178360   Min.   :-0.173100
##  1st Qu.:-0.39091   1st Qu.:-0.054065   1st Qu.:-0.051063
##  Median : 0.00000   Median :-0.002186   Median : 0.000648
##  Mean   :-0.05157   Mean   : 0.008371   Mean   : 0.011003
##  3rd Qu.: 0.32860   3rd Qu.: 0.059006   3rd Qu.: 0.060957
##  Max.   :10.00000   Max.   : 9.253832   Max.   : 8.956020
```

```
summary(CData[,vars])
```

```
##  plus_minus_pm             mPM                 rPM
##  Min.   :-180.00000   Min.   :-1.252699   Min.   :-1.240810
##  1st Qu.:  -0.42060   1st Qu.:-0.101094   1st Qu.:-0.092011
##  Median :  -0.02878   Median :-0.004777   Median : 0.001438
##  Mean   :  -0.11963   Mean   : 0.005160   Mean   : 0.011345
##  3rd Qu.:   0.32673   3rd Qu.: 0.102772   3rd Qu.: 0.106618
##  Max.   :   6.66667   Max.   : 2.470564   Max.   : 2.353700
```

The results obtained from this are promising, the first thing to note is that for all positions the minimum and maximum values are less extreme than the actual observed values. This makes sense due to the models moving every value closer towards the median which is going to be close to 0. The median for both models is very close to 0 for all positions, with the median for the robust regression model being slightly closer to 0 for every position except for point guards. This coupled with the smaller intervals of [Min, Max] for the robust regression model is why I'm going to continue using this model as it is less influenced by extreme performances one way or another and I want to get the best idea as to how good an individual player is as a whole and I don't want this to be heavily influenced by "fluke" performance whether they are good or bad. The next step is to compile a data frame that contains player's totals to see how players rank based on the model and get a good idea as to how effective it is:

```
totPGdata <- ddply(PGData, .(player, player_id), summarise, position = "PG",
                   games = length(minutes), mins = sum(minutes),
                   plus_minus_pm = weighted.mean(plus_minus_pm, minutes),
                   drb_pm = weighted.mean(drb_pm, minutes),
                   ast_pm = weighted.mean(ast_pm, minutes),
                   stl_pm = weighted.mean(stl_pm, minutes),
                   tov_pm = weighted.mean(tov_pm, minutes),
                   rPM_pm = weighted.mean(rPM, minutes))
totPGdata$plus_minus <- with(totPGdata, plus_minus_pm * mins)
totPGdata$rPM <- with(totPGdata, rPM_pm * mins)
head(totPGdata[order(totPGdata$rPM, decreasing = TRUE),c(1,13)]) #view highest rPM PGs
```

```
##              player      rPM
## 86     LeBron James 614.5351
## 89      Luka Doncic 459.2801
## 17       Chris Paul 341.6196
## 111     Ricky Rubio 338.8708
## 6       Ben Simmons 334.1128
## 29  Dejounte Murray 291.3571
```

From the highest rated point guards list, there are some immediate good signs, the first thing being that Lebron James and Luka Doncic are the top 2 players. I think that it'd be hard to find anyone who can make a solid argument as to why Lebron James wouldn't be first and Luka Doncic has also played amazingly well the last year and a half (which is the span of data being considered), so his name at second makes sense as well. The other players are all players that would be considered near the top of point guards in the league, one thing to keep in mind is that players who have been injured or have not played many minutes in the last year and a half won't be as highly ranked, so someone like Stephen Curry will be much lower than what people might rank him. There does also seem to be a pretty heavy emphasis on players who are strong defensively; with Chris Paul, Ricky Rubio, Ben Simmons and Dejounte Murray all being strong defenders. Some people might think that there is a bias that is too strongly defensively skewed, but maybe there is actually an undervaluing of defense at the point guard position currently. Let's take a look at the top players when considering all positions:

```
#Shooting guards model data:
totSGdata <- ddply(SGData, .(player, player_id), summarise, position = "SG",
                   games = length(minutes), mins = sum(minutes),
                   plus_minus_pm = weighted.mean(plus_minus_pm, minutes),
                   drb_pm = weighted.mean(drb_pm, minutes),
                   ast_pm = weighted.mean(ast_pm, minutes),
                   stl_pm = weighted.mean(stl_pm, minutes),
                   blk_pm = weighted.mean(blk_pm, minutes),
                   tov_pm = weighted.mean(tov_pm, minutes),
                   rPM_pm = weighted.mean(rPM, minutes))
totSGdata$plus_minus <- with(totSGdata, plus_minus_pm * mins)
totSGdata$rPM <- with(totSGdata, rPM_pm * mins)

#Small forwards model data:
totSFdata <- ddply(SFData, .(player, player_id), summarise, position = "SF",
                   games = length(minutes), mins = sum(minutes),
                   plus_minus_pm = weighted.mean(plus_minus_pm, minutes),
                   drb_pm = weighted.mean(drb_pm, minutes),
                   ast_pm = weighted.mean(ast_pm, minutes),
                   stl_pm = weighted.mean(stl_pm, minutes),
                   rPM_pm = weighted.mean(rPM, minutes))
totSFdata$plus_minus <- with(totSFdata, plus_minus_pm * mins)
```

```
totSFdata$rPM <- with(totSFdata, rPM_pm * mins)

#Power forwards model data:
totPFdata <- ddply(PFData, .(player, player_id), summarise, position = "PF",
                   games = length(minutes), mins = sum(minutes),
                   plus_minus_pm = weighted.mean(plus_minus_pm, minutes),
                   ts_pm = weighted.mean(ts_pm, minutes),
                   drb_pm = weighted.mean(drb_pm, minutes),
                   blk_pm = weighted.mean(blk_pm, minutes),
                   rPM_pm = weighted.mean(rPM, minutes))
totPFdata$plus_minus <- with(totPFdata, plus_minus_pm * mins)
totPFdata$rPM <- with(totPFdata, rPM_pm * mins)

#Center model data:
totCdata <- ddply(CData, .(player, player_id), summarise, position = "C",
                  games = length(minutes), mins = sum(minutes),
                  plus_minus_pm = weighted.mean(plus_minus_pm, minutes),
                  ts_pm = weighted.mean(ts_pm, minutes),
                  drb_pm = weighted.mean(drb_pm, minutes),
                  ast_pm = weighted.mean(ast_pm, minutes),
                  blk_pm = weighted.mean(blk_pm, minutes),
                  usg_pct = weighted.mean(usg_pct, minutes),
                  rPM_pm = weighted.mean(rPM, minutes))
totCdata$plus_minus <- with(totCdata, plus_minus_pm * mins)
totCdata$rPM <- with(totCdata, rPM_pm * mins)

#Set up for the data frame with all of the players from all positions:
totData <- rbind(totPGdata[,c(1:5, 12:13)], totSGdata[,c(1:5, 13:14)],
                 totSFdata[,c(1:5, 11:12)], totPFdata[,c(1:5, 11:12)],
                 totCdata[,c(1:5, 13:14)])
head(totData[order(totData$rPM, decreasing = TRUE),c(1,7)]) #view highest rPM players
```

```
##                     player      rPM
## 485 Giannis Antetokounmpo 621.6584
## 86           LeBron James 614.5351
## 89            Luka Doncic 459.2801
## 199          James Harden 405.2330
## 359          Jimmy Butler 393.8893
## 640      Hassan Whiteside 382.7334
```

Every single one of these players, except for Hassan Whiteside would easily be able to have a conversation for being a top player in the NBA over the last year and a half. Why is Hassan Whiteside here then? Since he's a player who gets way more blocks than the average center almost every game, all of his games are outliers relative to the population of centers. This leads to his blocks being valued too highly and is the same reason some other players, who are very good in other categories like steals and blocks where players will average very low numbers, are valued higher than one might expect. So, I can conclude that this model isn't perfect, it has a general trend of placing players where they "should" be but has some issues with certain types of players. However, the point of this report was not only to see if I could find a way to rank players in the NBA by skill, but also to illustrate the importance of sample size. To do this I'm going to use the example of Lamelo Ball.

Lamelo Ball is a rookie point guard for the Charlotte Hornets who has played 26 games this season. The first thing to note when talking about sample size's importance is that I'll be evaluating the player's value of rPM per minute. This will be my "skill" metric, it isn't too important how accurate it is but more so how much this value is changing with the number of games.

To start, Lamelo Ball's rPM/min is 0.1220021043. This places him at 8th out of NBA point guards, but let's be a little bit more selective with who qualifies as ranked and say that we'll only consider players who have played at least 70 games in the last year and a half. That would place Lamelo at 5th in the NBA with his closest comparison in terms of skill being Ben Simmons who is 6th with an rPM/min of 0.1217612245. So does this mean that Lamelo Ball is the 5th best point guard in the NBA? No, it means that I have to dive a little bit deeper into the sampling errors.

The first thing to note is that I'll have to use the residual standard error for the robust regression model used on point guards since that was the model used for the rPM/min values. The residual standard error is $\hat{\sigma} = 0.4293$. This standard error is set using individual games of data, so if I want to get the standard error for rPM/min when it is the average over all of the player's games, I have to use this equation $SE = \frac{\hat{\sigma}}{\sqrt{n}}$ where n is the number of games played by the player.

Using the standard error and the number of games played by Lamelo Ball (26), I can make a 95% confidence interval with the following equation: $rPM/min \pm z \times \frac{\hat{\sigma}}{\sqrt{n}}$ where z is the z score of the normal distribution calculated with the following R code:

```
alpha <- 0.05
z <- qnorm(1-alpha/2)
z
```

```
## [1] 1.959964
```

Putting all of these values together, Lamelo Ball's 95% confidence interval for his rPM/min is:

$$rPM/min \pm z \times \frac{\hat{\sigma}}{\sqrt{n}} = 0.1220021043 \pm 1.959964 \times \frac{0.4293}{\sqrt{26}} = [-0.04301247, 0.2870167]$$

Looking at where these values would place him; the minimum value of -0.04301247 would place him 37th out of all 41 qualified point guards (70 games or more) which is in between Austin Rivers and Dennis Schroder, while his maximum value of 0.2870167 would place him first with almost double Lebron James's rPM/min (0.153011523). So what does that mean? That means that his true skill is somewhere between the best point guard in the NBA handedly and the 37th best point guard in the league in 95% of attempts where this experiment is run. So, was anything achieved from this stat given Lamelo Ball has only played 26 games? No.

So that brings up the question, at what point does a player's rPM/min become a valuable statistic at evaluating their skills. Well the first thing to look at would be what level of accuracy am I considering to be necessary to determine a player's skill? In this ranking of point guards most of the gaps between ranks tend to have a difference of roughly 0.01, so that is the interval size I will want for my confidence interval. That means that I need the z score times the standard error to be $\frac{0.01}{2}$. Plugging that in and solving for n I get:

$$0.005 = z \times \frac{\hat{\sigma}}{\sqrt{n}} = 1.959964 \times \frac{0.4293}{\sqrt{n}}$$

$$n = (1.959964 \times \frac{0.4293}{0.005})^2 = 283.19$$

This means that after 284 games, you would be able to get an acceptably accurate ranking as to where a point guard was relative to the rest of the league. That is more than three full seasons of playing every game! Needless to say that many basketball fans come to conclusions far too quickly about how good or bad a player is.