Untitled

GROUP: Group 4 MEMBERS: Vignesh Mahalingam, Brandon Wang, Arjie Nanda TITLE: Predicting Player Minutes based on Box Score Statistics PURPOSE: We plan on predicting minutes played through the other types of basketball statistics. Minutes played is a valuable stat because it keeps track of how often a player is on the court. More minutes played gives players more time to contribute to the game. We will use a combination of both counting stats and rate stats. We hypothesize that the counting stats will be positively correlated with minutes played, while the rate stats should have no relationship. It is certainly possible that players who have higher rate stats, like three point percentage, are more valuable and will play more minutes as a result. On the other hand, players who play fewer minutes could have better rates due to the increased variability in a smaller sample. DATA: We are using the per 36 player data from the NBA's 2018-19 season, which can be found at https://www.basketball-reference.com/leagues/NBA_2019_per_minute.html . This data is taken from the NBA's own statistics page, and is a widely used industry source. Per 36 refers to adjusting the player's statistics to project what their stats would be if the player played 36 minutes per game. We will be using the per 36 data as it provides a good insight into a player's productivity without interference from the number of minutes played by the player.
POPULATION: The individual population units are players with a minimum of 41 games played, which is half the season. We're trying to generalize this to future NBA seasons, as a method of estimating the average number of minutes played by a given player. We believe there are around 240 such players in the league.

RESPONSE VARIABLE(S): The number of minutes played per game. Self-evidently, this statistic is recorded in minutes. We know the range for this value stretches from 1 to around 45, although both extremes will be rare. This is because a regulation NBA game lasts for 48 minutes, but overtime can also play a role in inflating player minutes. The dataset provided does not have minutes played per game, instead having minutes played for the total season. We will create our response variable by dividing the number of minutes played by the number of games played.

EXPLANATORY VARIABLES: All variables are measured by official NBA scorekeepers at the games. Total rebounds in a 36 minute time period: This is the total offensive and defensive rebounds in a 36 minute time period. The unit is rebounds. Total blocks per 36 minutes, the unit is blocks. Turnovers per 36 mins, the unit is turnovers. Free throw attempts per 36 min time period, the unit is free throw attempts. Points per 36 minutes, the unit is points. Field goal percentage per game, the unit is percentage.

EXPLORATORY ANALYSIS:

```
nbaper36 <- read_csv("nbaper36.csv") #reads in csv
```

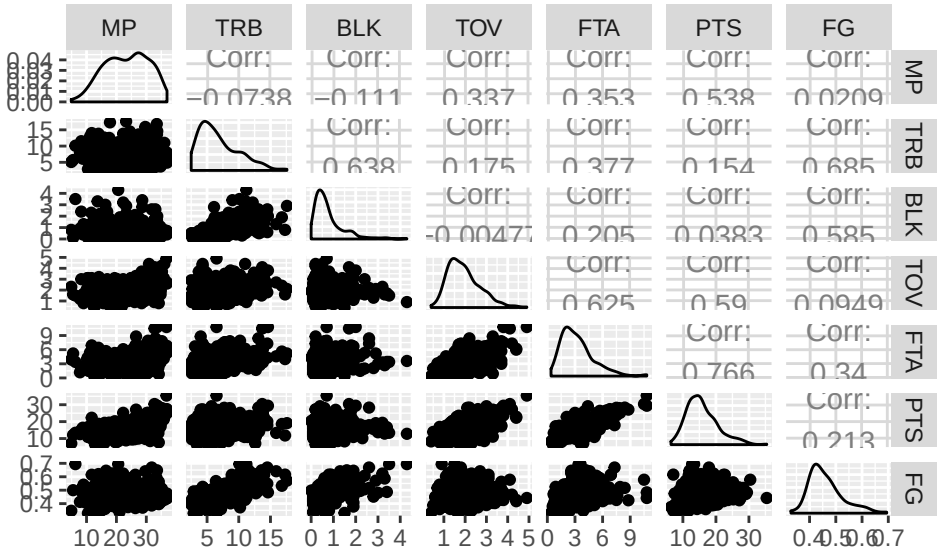
```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Player = col_character(),
##   Pos = col_character(),
##   Tm = col_character()
## )
```

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

```
totplayers = as.list(nbaper36 %>% filter(Tm == "TOT") %>% select(Player))#list of players on multiple t
nbaper36 = nbaper36 %>% mutate(MP = MP/G) %>% filter(G > 41) %>% filter(!(Player %in% totplayers$Player
```

```
nbaper36select = select(nbaper36, Player, MP, TRB, BLK, TOV, FTA, PTS, 'FG%') %>% rename(FG = 'FG%')

GGally::ggpairs(select(nbaper36select, -Player))
```

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

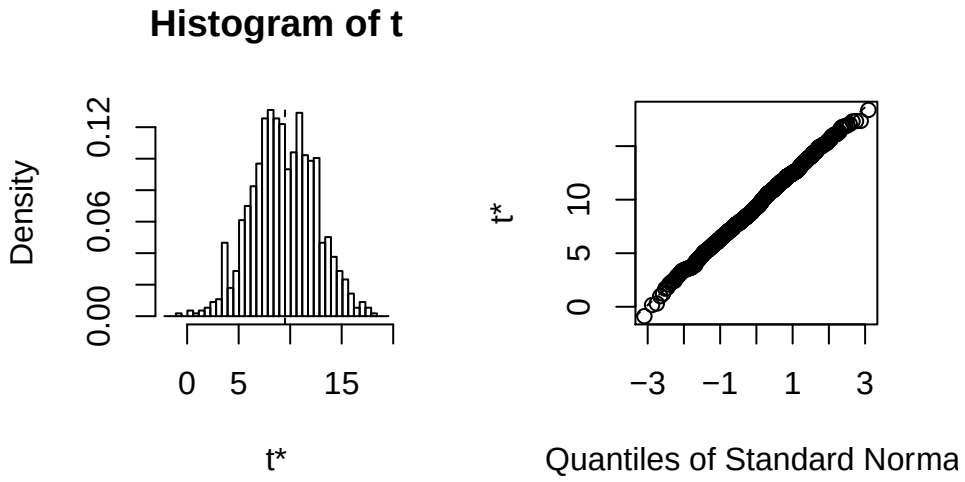



```
set.seed(1109)


# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data = nbaper36select, statistic=bs,
   R=1000, formula = MP ~ TRB + BLK + TOV + FTA + PTS + FG)

# view results
results
```

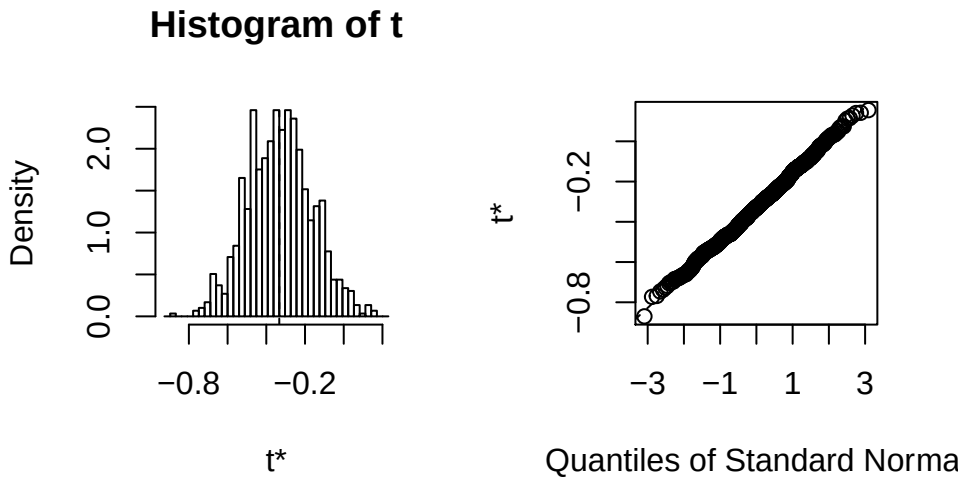
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = nbaper36select, statistic = bs, R = 1000, formula = MP ~
##     TRB + BLK + TOV + FTA + PTS + FG)
##
##
```

```
## Bootstrap Statistics :
##       original       bias    std. error
## t1*  9.4991783 -0.103459103   3.0811775
## t2* -0.3330206  0.003760774   0.1661718
## t3* -0.5885414  0.034877910   0.7164654
## t4*  0.6692768  0.026903330   0.6128558
## t5* -0.3295594 -0.002304849   0.3567487
## t6*  0.8790656 -0.001046784   0.1126845
## t7*  5.4924707  0.068660510   7.6696190
```

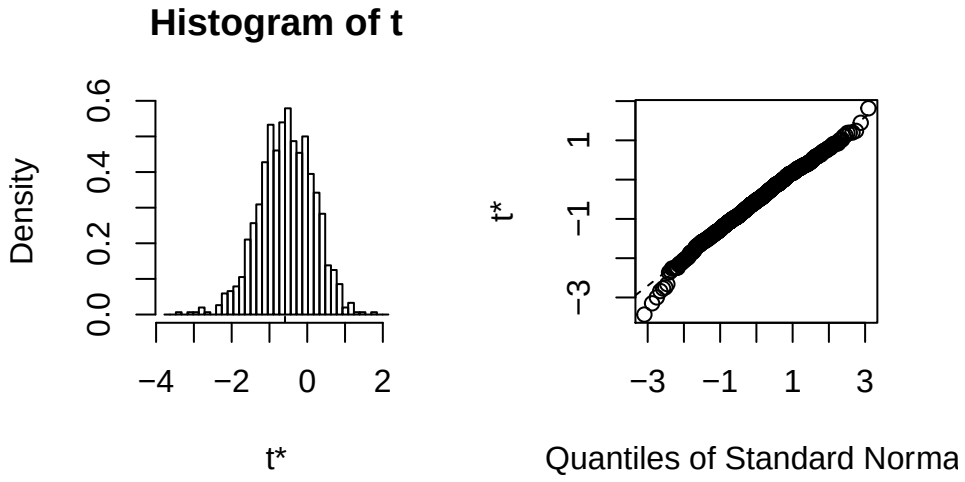
```r
plot(results, index=1) # intercept
```



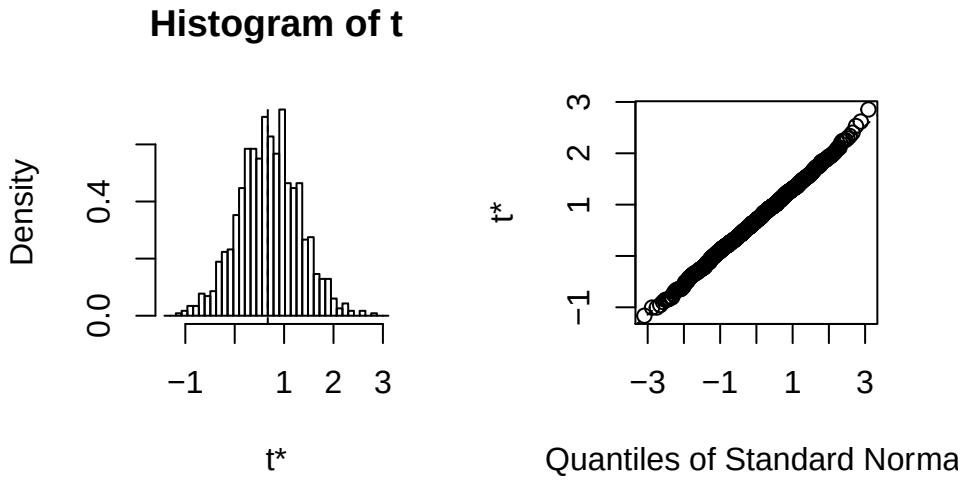

```r
plot(results, index=2) # TRB
```



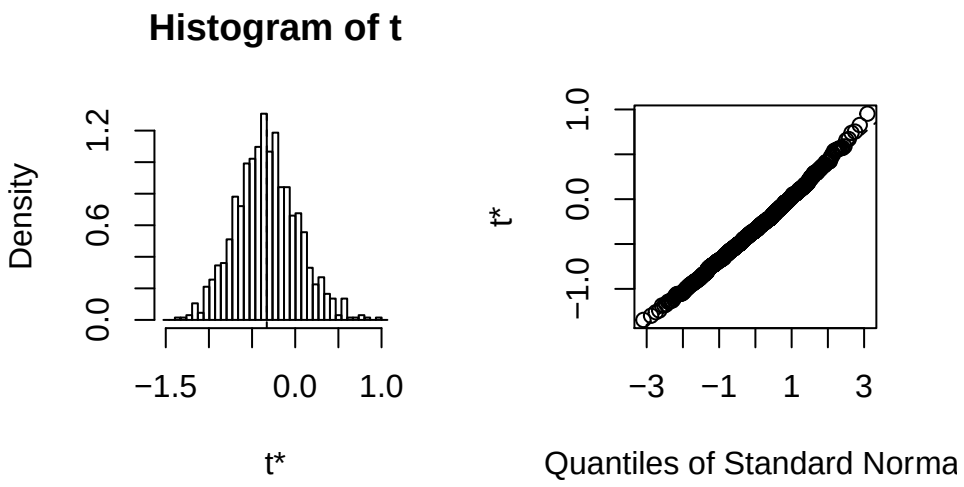
```r
plot(results, index=3) # BLK
```



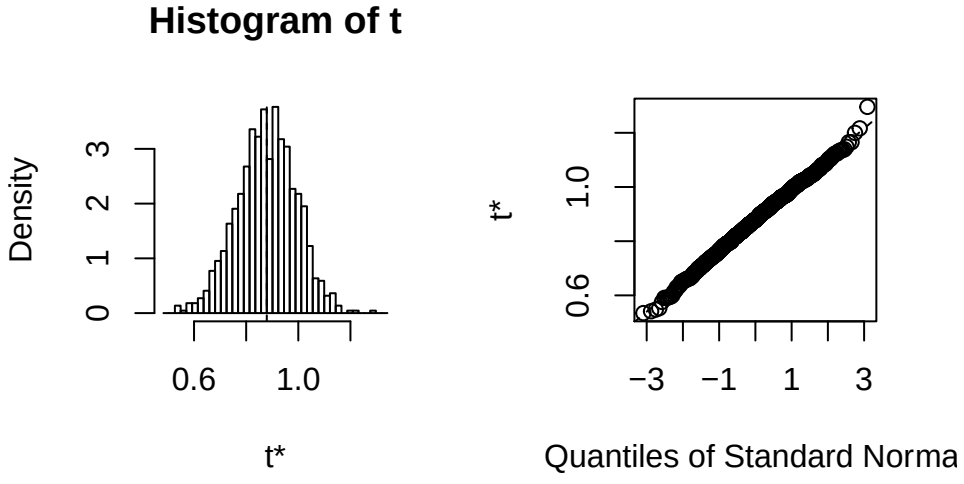

```r
plot(results, index=4) # TOV
```



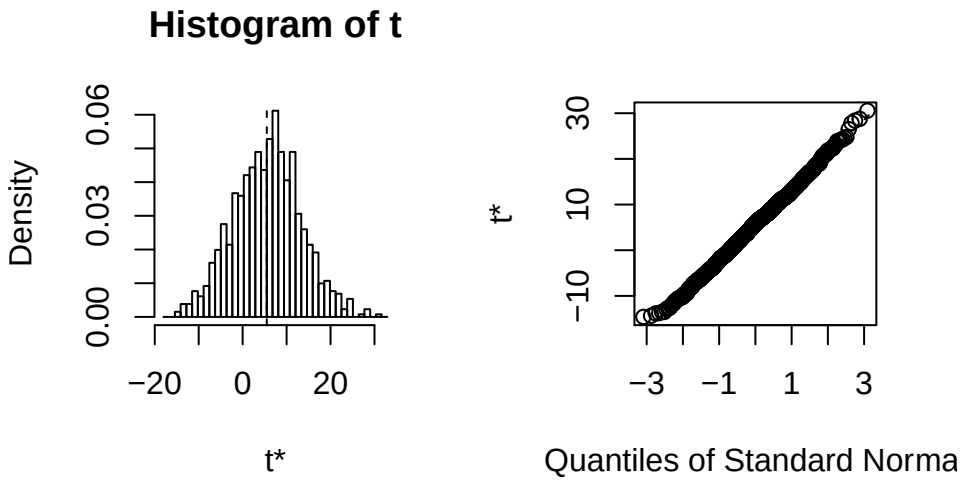

```r
plot(results, index=5) # FTA
```




```
plot(results, index=6) # PTS
```




```
plot(results, index=7) # FG%
```




```
# get 95% confidence intervals
boot.ci(results, type="bca", index=1) # intercept
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   ( 3.641, 15.905 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=2) # TRB
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 2)
##
## Intervals :
## Level       BCa
## 95%   (-0.6604,  0.0115 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=3) # BLK
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
```

```
## boot.ci(boot.out = results, type = "bca", index = 3)
##
## Intervals :
## Level       BCa
## 95%   (-2.1458,  0.6804 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=4) # TOV
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 4)
##
## Intervals :
## Level       BCa
## 95%   (-0.5889,  1.8713 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=5) # FTA
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 5)
##
## Intervals :
## Level       BCa
## 95%   (-0.9850,  0.4208 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=6) # PTS
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 6)
##
## Intervals :
## Level       BCa
## 95%   ( 0.6582,  1.0986 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(results, type="bca", index=7) # FG%
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 7)
##
## Intervals :
## Level       BCa
## 95%   (-10.35,  20.64 )
## Calculations and Intervals on Original Scale
```

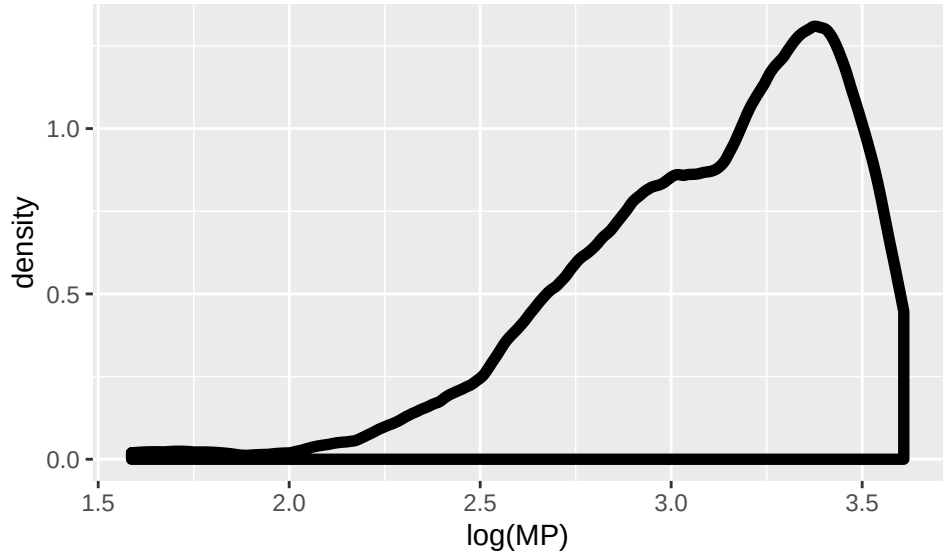
```
#OLM

model_olm = lm(data = nbaper36, MP ~ TRB + BLK + TOV + FTA + PTS + FG)
summary(model_olm) # Only PTS is significant s
```

```
##
## Call:
## lm(formula = MP ~ TRB + BLK + TOV + FTA + PTS + FG, data = nbaper36)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.9754  -3.9438   0.3711   4.1442  17.4790
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.2326     1.5476   7.258 2.66e-12 ***
## TRB          -0.2459     0.1610  -1.527  0.12762
## BLK          -0.4072     0.6590  -0.618  0.53704
## TOV           0.6613     0.5788   1.142  0.25408
## FTA          -0.3994     0.3859  -1.035  0.30133
## PTS           1.0317     0.3650   2.827  0.00498 **
## FG           -0.3579     0.8810  -0.406  0.68485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.022 on 342 degrees of freedom
## Multiple R-squared:  0.3197, Adjusted R-squared:  0.3077
## F-statistic: 26.78 on 6 and 342 DF,  p-value: < 2.2e-16
```

```
# JHM
model_f = rfit(data = nbaper36select, MP ~ TRB + BLK + TOV + FTA + PTS + FG)
summary(model_f)$coefficients
```

```
##               Estimate Std. Error    t.value      p.value
## (Intercept)  9.7100041  3.0890344  3.1433784 1.816214e-03
## TRB         -0.3663835  0.1706285 -2.1472579 3.247528e-02
## BLK         -0.5389857  0.7050211 -0.7644958 4.450990e-01
## TOV          0.6767090  0.5999443  1.1279529 2.601305e-01
## FTA         -0.3255316  0.3466229 -0.9391522 3.483154e-01
## PTS          0.9027934  0.1186668  7.6077979 2.728345e-13
## FG           5.8233237  7.5897603  0.7672606 4.434560e-01
```

```
ggplot(data = nbaper36select, aes(log(MP))) +
  geom_density(bw = "SJ", kernel = "epanechnikov", size = 2)
```




```
#ols
ols_mod <- lm(MP ~ TRB + BLK + TOV + FTA + PTS + FG, data = nbaper36select)
summary(ols_mod)
```

```
##
## Call:
## lm(formula = MP ~ TRB + BLK + TOV + FTA + PTS + FG, data = nbaper36select)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.0228  -4.0451   0.5064   4.1991  17.8150
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4992     2.9329   3.239  0.00132 **
## TRB          -0.3330     0.1625  -2.049  0.04119 *
## BLK          -0.5885     0.6715  -0.877  0.38136
## TOV           0.6693     0.5714   1.171  0.24228
## FTA          -0.3296     0.3301  -0.998  0.31884
## PTS           0.8791     0.1130   7.778 8.76e-14 ***
## FG            5.4925     7.2284   0.760  0.44787
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.019 on 342 degrees of freedom
## Multiple R-squared:  0.3205, Adjusted R-squared:  0.3086
## F-statistic: 26.88 on 6 and 342 DF,  p-value: < 2.2e-16
```

```
#reb spline
simp_reb <- lm(MP ~ TRB, data = nbaper36select)
s_reb <- gam(MP ~ s(TRB), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
AIC(simp_reb)
```

```
## [1] 2375.105
```

```
AIC(s_reb) #slightly lower aic
```

```
## [1] 2373.432
```

```
#blk spline
simp_blk <- lm(MP ~ BLK, data = nbaper36select)
s_blk <- gam(MP ~ s(BLK), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
AIC(simp_blk)
```

```
## [1] 2372.685
```

```
AIC(s_blk) #slightly lower aic
```

```
## [1] 2372.517
```

```
#tov spline
simp_tov <- lm(MP ~ TOV, data = nbaper36select)
s_tov <- gam(MP ~ s(TOV), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
AIC(simp_tov) #slightly lower aic
```

```
## [1] 2334.933
```

```
AIC(s_tov)
```

```
## [1] 2335.894
```

```
#fta spline
simp_fta <- lm(MP ~ FTA, data = nbaper36select)
s_fta <- gam(MP ~ s(FTA), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
AIC(simp_fta) #slightly lower aic
```

```
## [1] 2330.47
```

```r
AIC(s_fta)
```

```
## [1] 2332.292
```

```r
#pts spline
simp_pts <- lm(MP ~ PTS, data = nbaper36select)
s_pts <- gam(MP ~ s(PTS), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
AIC(simp_pts)
```

```
## [1] 2257.746
```

```r
AIC(s_pts) #slightly lower aic
```

```
## [1] 2255.181
```

```r
#fg spline
simp_fg <- lm(MP ~ FG, data = nbaper36select)
s_fg <- gam(MP ~ s(FG), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
AIC(simp_fg)
```

```
## [1] 2376.857
```

```r
AIC(s_fg) #lower aic
```

```
## [1] 2354.373
```

```r
#gam
full_gam <- gam(MP ~ s(TRB) + s(BLK) + TOV + FTA + s(PTS) + s(FG), data = nbaper36select)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
full_gam_c = predict(full_gam, type = "terms")
colnames(full_gam_c) = c("trb_pred", "blk_pred", "tov_pred", "fta_pred", "pts_pred", "fg_pred")
mp_m = mean(nbaper36select$MP)
```

```r
#graphing gam

plot1 <- ggplot(data = nbaper36select, aes(x = TRB, y = MP)) +
  geom_point() +
  geom_hline(yintercept = mp_m, linetype = 2, color = "blue") +
  geom_line(inherit.aes = F, size = 2, color = "gold", data = )
```