Modeling Oliver’s Four Factors

Taaj Cheema

---

1. Loading and Exploring Data

1.1 Loading required libraries

```
library(car)
library(caret)
library(corrplot)
library(dplyr)
library(gbm)
library(GGally)
library(ggplot2)
library(plotly)
library(randomForest)
library(readxl)
library(rmarkdown)
library(scales)
library(sjstats)
library(tidyr)
library(viridis)
```

1.2 Loading our dataset of interest as a dataframe

```
four_factors <- read_excel("four_factors_all_seasons.xlsx")
```

1.3 Data exploration

```
head(four_factors)
```

```
## # A tibble: 6 x 13
##   Team  Season  WINS   MOV  NRTG `eFG%` `TOV%` `OREB%`   FTF `OPPeFG%` `OPPTOV%`
##   <chr> <chr>  <dbl> <dbl> <dbl>  <dbl>  <dbl>   <dbl> <dbl>     <dbl>     <dbl>
## 1 Milw… 2018 …    60  8.87   8.6  0.55    12      20.8 0.197     0.503      11.5
## 2 Toro… 2018 …    58  6.09   6    0.543   12.4    21.9 0.198     0.509      13.1
## 3 Gold… 2018 …    57  6.46   6.4  0.565   12.6    22.5 0.182     0.508      11.7
## 4 Denv… 2018 …    54  3.95   4.1  0.527   11.9    26.6 0.175     0.521      12.3
## 5 Hous… 2018 …    53  4.77   4.8  0.542   12      22.8 0.221     0.525      13.4
## 6 Port… 2018 …    53  4.2    4.2  0.528   12.1    26.6 0.21      0.516      11
## # … with 2 more variables: `DREB%` <dbl>, OPPFTF <dbl>
```

```
summary(four_factors)
```

```
##      Team              Season               WINS             MOV            
##  Length:330         Length:330         Min.   : 8.697   Min.   :-13.910000  
##  Class :character   Class :character   1st Qu.:31.295   1st Qu.: -3.260000  
##  Mode  :character   Mode  :character   Median :42.000   Median :  0.245000  
##                                        Mean   :40.997   Mean   :  0.000394  
##                                        3rd Qu.:50.000   3rd Qu.:  3.432500  
##                                        Max.   :73.000   Max.   : 11.630000  
##       NRTG                 eFG%             TOV%            OREB%      
##  Min.   :-15.200000   Min.   :0.4390   Min.   :10.70   Min.   :18.00  
##  1st Qu.: -3.475000   1st Qu.:0.4890   1st Qu.:12.60   1st Qu.:23.10  
##  Median :  0.250000   Median :0.5020   Median :13.20   Median :25.10  
##  Mean   :  0.009697   Mean   :0.5038   Mean   :13.24   Mean   :25.07  
##  3rd Qu.:  3.600000   3rd Qu.:0.5188   3rd Qu.:13.90   3rd Qu.:27.20  
##  Max.   : 11.600000   Max.   :0.5690   Max.   :16.00   Max.   :32.60  
##       FTF            OPPeFG%          OPPTOV%           DREB%      
##  Min.   :0.1430   Min.   :0.4500   Min.   :10.50   Min.   :68.10  
##  1st Qu.:0.1950   1st Qu.:0.4910   1st Qu.:12.60   1st Qu.:73.30  
##  Median :0.2100   Median :0.5050   Median :13.20   Median :74.80  
##  Mean   :0.2123   Mean   :0.5037   Mean   :13.24   Mean   :74.92  
##  3rd Qu.:0.2280   3rd Qu.:0.5160   3rd Qu.:13.90   3rd Qu.:76.40  
##  Max.   :0.2990   Max.   :0.5640   Max.   :16.30   Max.   :81.20  
##      OPPFTF      
##  Min.   :0.1000  
##  1st Qu.:0.1940  
##  Median :0.2110  
##  Mean   :0.2121  
##  3rd Qu.:0.2270  
##  Max.   :0.3150  
```

1.4 Renaming columns containing special characters

```
four_factors_new <- four_factors %>% dplyr::rename(eFG = `eFG%`,
                                TOV = `TOV%`,
                                OREB = `OREB%`,
                                OPPeFG = `OPPeFG%`,
                                OPPTOV = `OPPTOV%`,
                                DREB = `DREB%`)

names(four_factors_new)
```

```
##  [1] "Team"   "Season" "WINS"   "MOV"    "NRTG"   "eFG"    "TOV"    "OREB"
##  [9] "FTF"    "OPPeFG" "OPPTOV" "DREB"   "OPPFTF"
```

1.5 Creating a new dataframe

```
four_factors_scaled = four_factors_new
four_factors_scaled
```

```
## # A tibble: 330 x 13
##    Team  Season  WINS   MOV  NRTG   eFG   TOV  OREB   FTF OPPeFG OPPTOV  DREB
##    <chr> <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
##  1 Milw… 2018 …    60  8.87   8.6 0.55   12    20.8 0.197  0.503   11.5  80.3
##  2 Toro… 2018 …    58  6.09   6   0.543  12.4  21.9 0.198  0.509   13.1  77.1
##  3 Gold… 2018 …    57  6.46   6.4 0.565  12.6  22.5 0.182  0.508   11.7  77.1
##  4 Denv… 2018 …    54  3.95   4.1 0.527  11.9  26.6 0.175  0.521   12.3  78
##  5 Hous… 2018 …    53  4.77   4.8 0.542  12    22.8 0.221  0.525   13.4  74.4
##  6 Port… 2018 …    53  4.2    4.2 0.528  12.1  26.6 0.21   0.516   11    77.9
##  7 Phil… 2018 …    51  2.7    2.6 0.532  12.9  24.5 0.241  0.512   11.1  78.6
##  8 Utah… 2018 …    50  5.26   5.2 0.538  13.4  22.9 0.217  0.507   12.4  80.3
##  9 Bost… 2018 …    49  4.44   4.4 0.534  11.5  21.6 0.173  0.514   13.4  77
## 10 Okla… 2018 …    49  3.4    3.3 0.514  11.7  26   0.19   0.523   14.4  78.2
## # … with 320 more rows, and 1 more variable: OPPFTF <dbl>
```

1.6 Scaling the `eFG`, `FTF`, `OPPeFG` and `OPPFTF` features by a factor of 100

```
four_factors_scaled$eFG <- four_factors_new$eFG * 100
four_factors_scaled$FTF <- four_factors_new$FTF * 100
four_factors_scaled$OPPeFG <- four_factors_new$OPPeFG * 100
four_factors_scaled$OPPFTF <- four_factors_new$OPPFTF * 100

summary(four_factors_scaled)
```

```
##      Team              Season              WINS             MOV
##  Length:330         Length:330         Min.   : 8.697   Min.   :-13.910000
##  Class :character   Class :character   1st Qu.:31.295   1st Qu.: -3.260000
##  Mode  :character   Mode  :character   Median :42.000   Median :  0.245000
##                                        Mean   :40.997   Mean   :  0.000394
##                                        3rd Qu.:50.000   3rd Qu.:  3.432500
##                                        Max.   :73.000   Max.   : 11.630000
##       NRTG                eFG             TOV             OREB
##  Min.   :-15.200000   Min.   :43.90   Min.   :10.70   Min.   :18.00
##  1st Qu.: -3.475000   1st Qu.:48.90   1st Qu.:12.60   1st Qu.:23.10
##  Median :  0.250000   Median :50.20   Median :13.20   Median :25.10
##  Mean   :  0.009697   Mean   :50.38   Mean   :13.24   Mean   :25.07
##  3rd Qu.:  3.600000   3rd Qu.:51.88   3rd Qu.:13.90   3rd Qu.:27.20
##  Max.   : 11.600000   Max.   :56.90   Max.   :16.00   Max.   :32.60
##       FTF            OPPeFG          OPPTOV           DREB
##  Min.   :14.30   Min.   :45.00   Min.   :10.50   Min.   :68.10
##  1st Qu.:19.50   1st Qu.:49.10   1st Qu.:12.60   1st Qu.:73.30
##  Median :21.00   Median :50.50   Median :13.20   Median :74.80
##  Mean   :21.23   Mean   :50.37   Mean   :13.24   Mean   :74.92
##  3rd Qu.:22.80   3rd Qu.:51.60   3rd Qu.:13.90   3rd Qu.:76.40
##  Max.   :29.90   Max.   :56.40   Max.   :16.30   Max.   :81.20
##      OPPFTF
##  Min.   :10.00
##  1st Qu.:19.40
##  Median :21.10
##  Mean   :21.21
##  3rd Qu.:22.70
##  Max.   :31.50
```

```
str(four_factors_scaled)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    330 obs. of  13 variables:
##  $ Team  : chr  "Milwaukee Bucks" "Toronto Raptors" "Golden State Warriors" "Denver N
uggets" ...
##  $ Season: chr  "2018 - 2019" "2018 - 2019" "2018 - 2019" "2018 - 2019" ...
##  $ WINS  : num  60 58 57 54 53 53 51 50 49 49 ...
##  $ MOV   : num  8.87 6.09 6.46 3.95 4.77 4.2 2.7 5.26 4.44 3.4 ...
##  $ NRTG  : num  8.6 6 6.4 4.1 4.8 4.2 2.6 5.2 4.4 3.3 ...
##  $ eFG   : num  55 54.3 56.5 52.7 54.2 52.8 53.2 53.8 53.4 51.4 ...
##  $ TOV   : num  12 12.4 12.6 11.9 12 12.1 12.9 13.4 11.5 11.7 ...
##  $ OREB  : num  20.8 21.9 22.5 26.6 22.8 26.6 24.5 22.9 21.6 26 ...
##  $ FTF   : num  19.7 19.8 18.2 17.5 22.1 21 24.1 21.7 17.3 19 ...
##  $ OPPeFG: num  50.3 50.9 50.8 52.1 52.5 51.6 51.2 50.7 51.4 52.3 ...
##  $ OPPTOV: num  11.5 13.1 11.7 12.3 13.4 11 11.1 12.4 13.4 14.4 ...
##  $ DREB  : num  80.3 77.1 77.1 78 74.4 77.9 78.6 80.3 77 78.2 ...
##  $ OPPFTF: num  16.2 19 20.5 19.4 21 19.5 20.6 18.9 19.8 20.6 ...
```

1.7 Creating octiles for the `WINS` feature

```r
wins_octile <- ntile(four_factors_scaled$WINS, 8)
four_factors_scaled$octile <- as.factor(wins_octile)

str(four_factors_scaled)
```

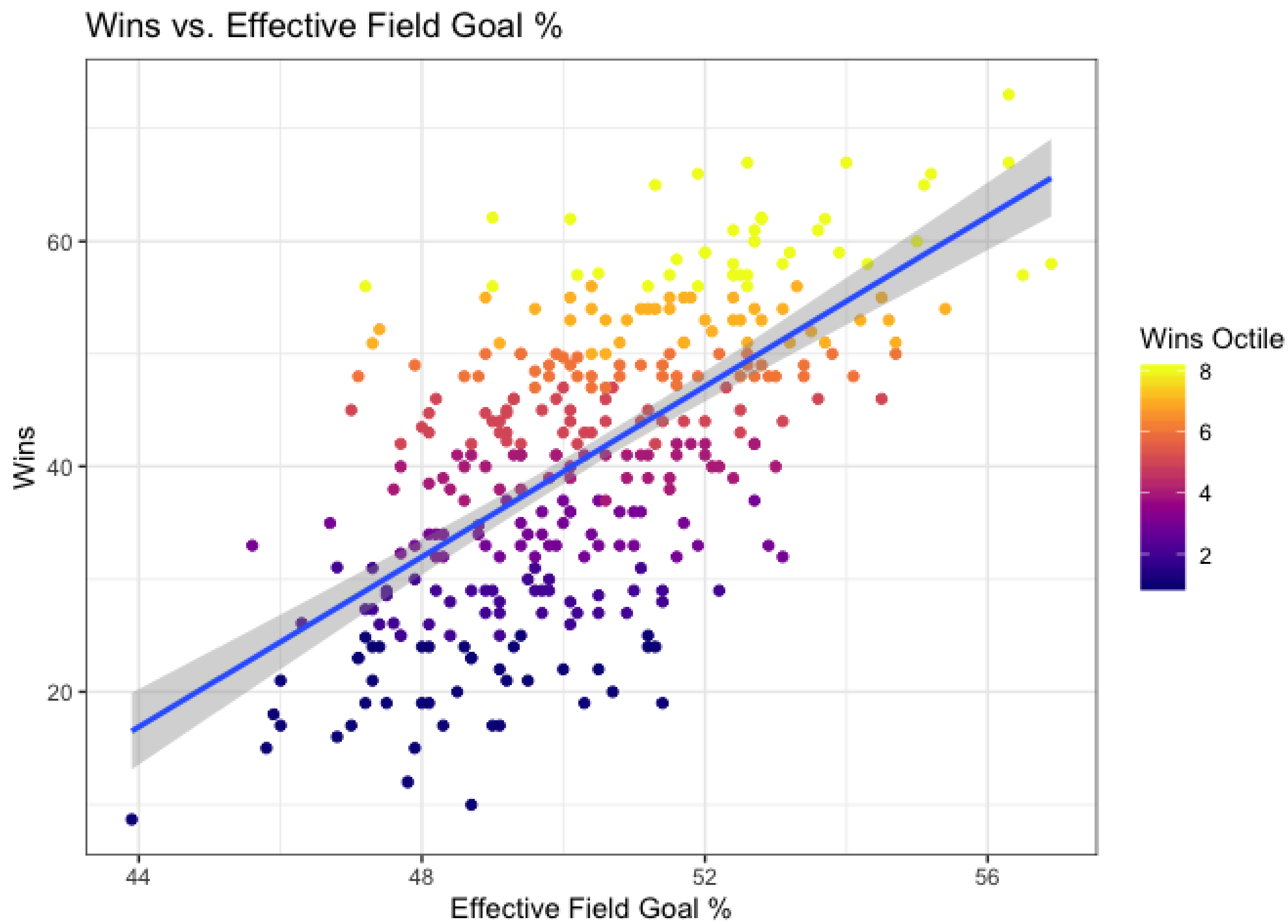
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    330 obs. of  14 variables:
##  $ Team  : chr  "Milwaukee Bucks" "Toronto Raptors" "Golden State Warriors" "Denver N
uggets" ...
##  $ Season: chr  "2018 - 2019" "2018 - 2019" "2018 - 2019" "2018 - 2019" ...
##  $ WINS  : num  60 58 57 54 53 53 51 50 49 49 ...
##  $ MOV   : num  8.87 6.09 6.46 3.95 4.77 4.2 2.7 5.26 4.44 3.4 ...
##  $ NRTG  : num  8.6 6 6.4 4.1 4.8 4.2 2.6 5.2 4.4 3.3 ...
##  $ eFG   : num  55 54.3 56.5 52.7 54.2 52.8 53.2 53.8 53.4 51.4 ...
##  $ TOV   : num  12 12.4 12.6 11.9 12 12.1 12.9 13.4 11.5 11.7 ...
##  $ OREB  : num  20.8 21.9 22.5 26.6 22.8 26.6 24.5 22.9 21.6 26 ...
##  $ FTF   : num  19.7 19.8 18.2 17.5 22.1 21 24.1 21.7 17.3 19 ...
##  $ OPPeFG: num  50.3 50.9 50.8 52.1 52.5 51.6 51.2 50.7 51.4 52.3 ...
##  $ OPPTOV: num  11.5 13.1 11.7 12.3 13.4 11 11.1 12.4 13.4 14.4 ...
##  $ DREB  : num  80.3 77.1 77.1 78 74.4 77.9 78.6 80.3 77 78.2 ...
##  $ OPPFTF: num  16.2 19 20.5 19.4 21 19.5 20.6 18.9 19.8 20.6 ...
##  $ octile: Factor w/ 8 levels "1","2","3","4",..: 8 8 8 7 7 7 7 6 6 6 ...
```

2. Graphical Analysis

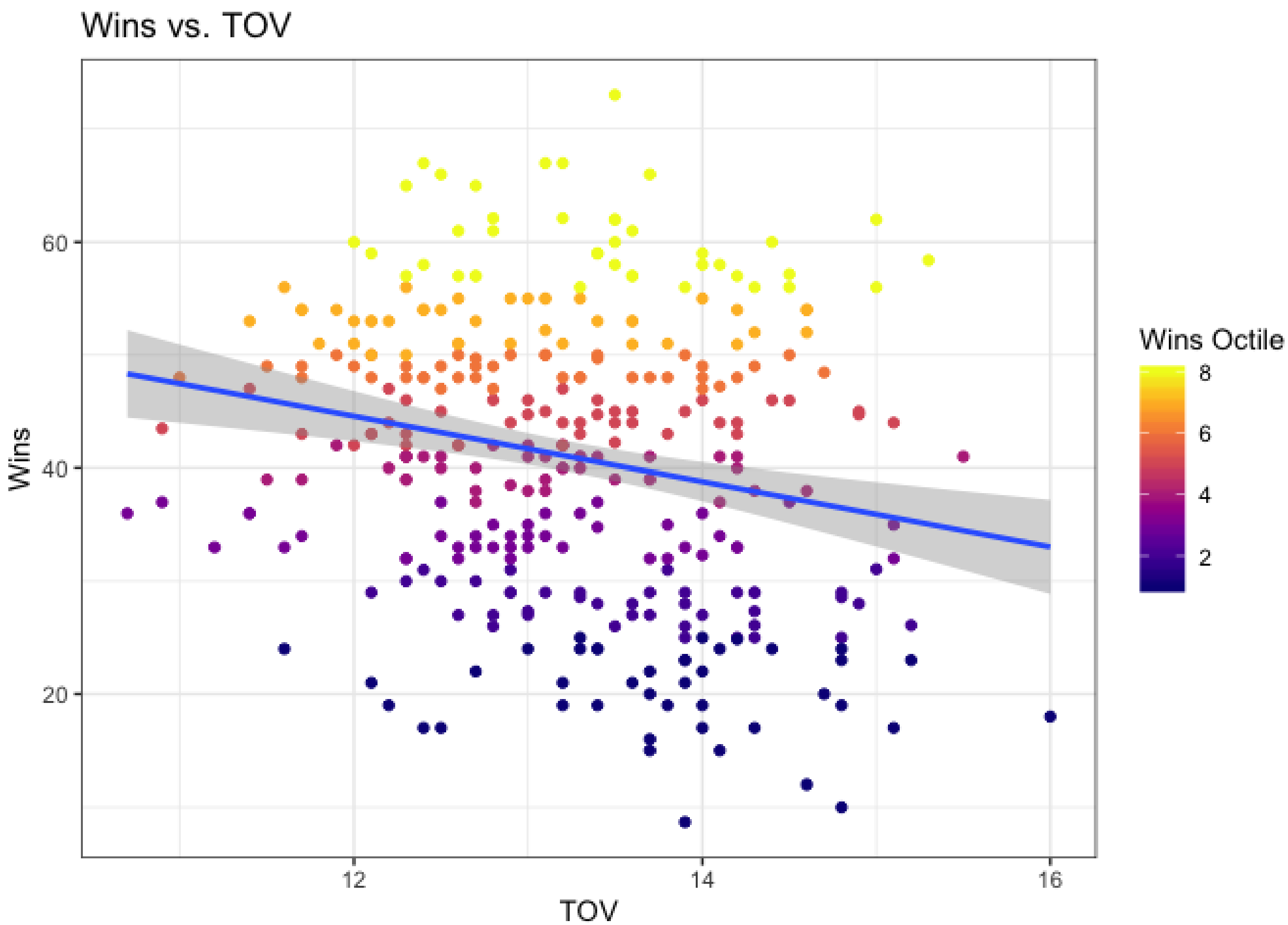
2.1 Scatter Plots

Checking graphically to see if there is a relationship between our response variable and predictor variables

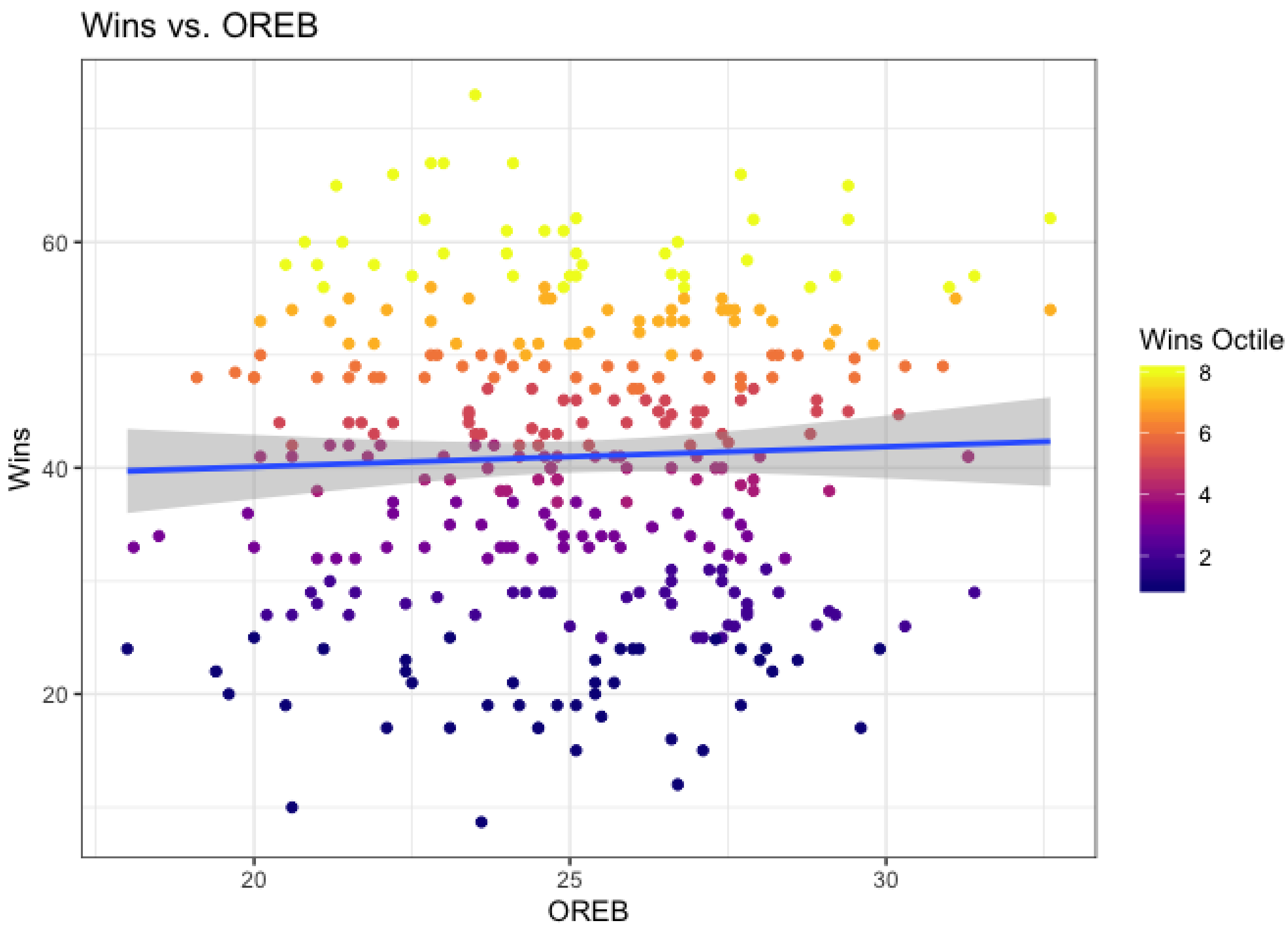
```r
ggplot(four_factors_scaled, aes(eFG, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. Effective Field Goal %") +
  xlab("Effective Field Goal %") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```



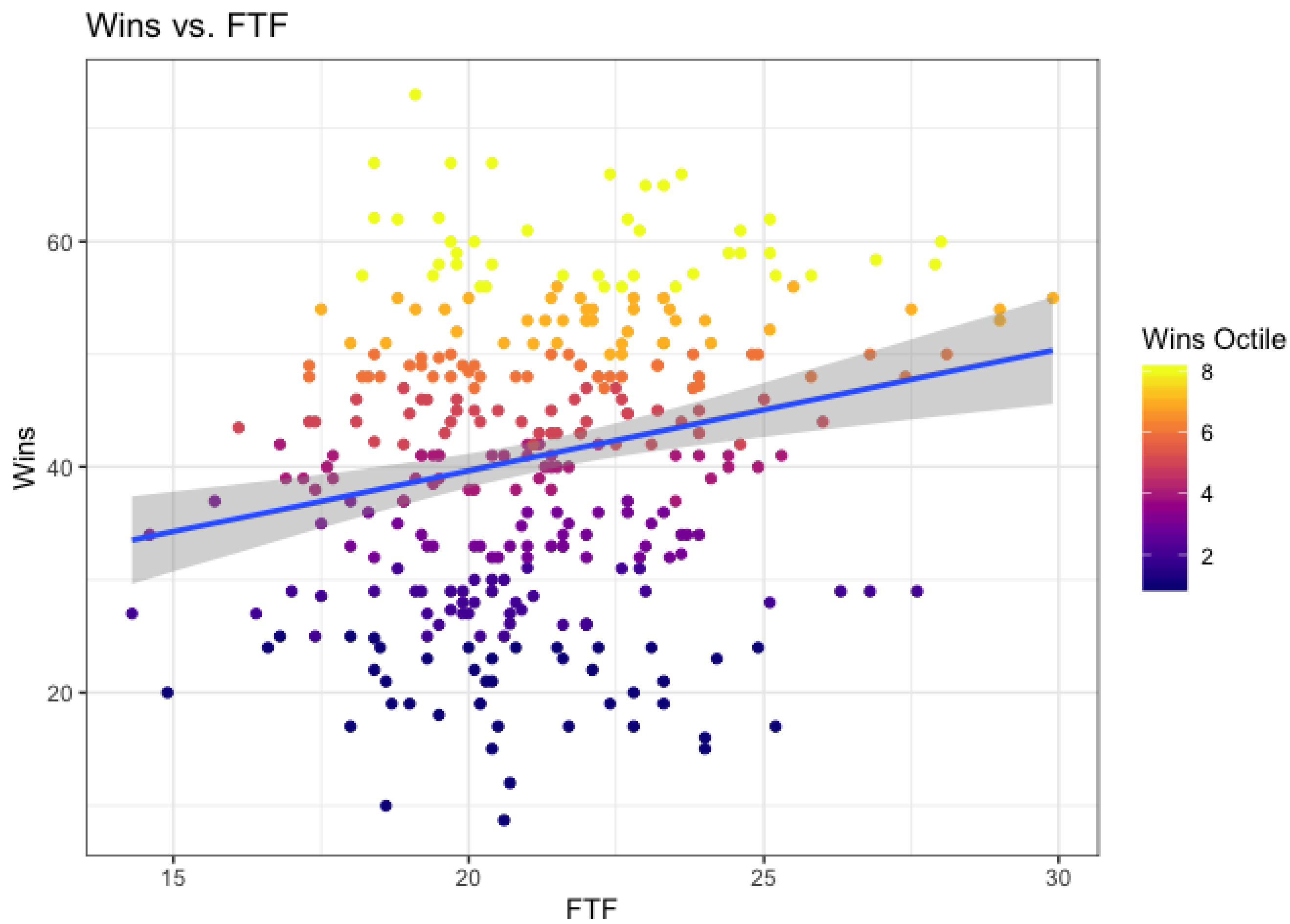

```
ggplot(four_factors_scaled, aes(TOV, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. TOV") +
  xlab("TOV") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```



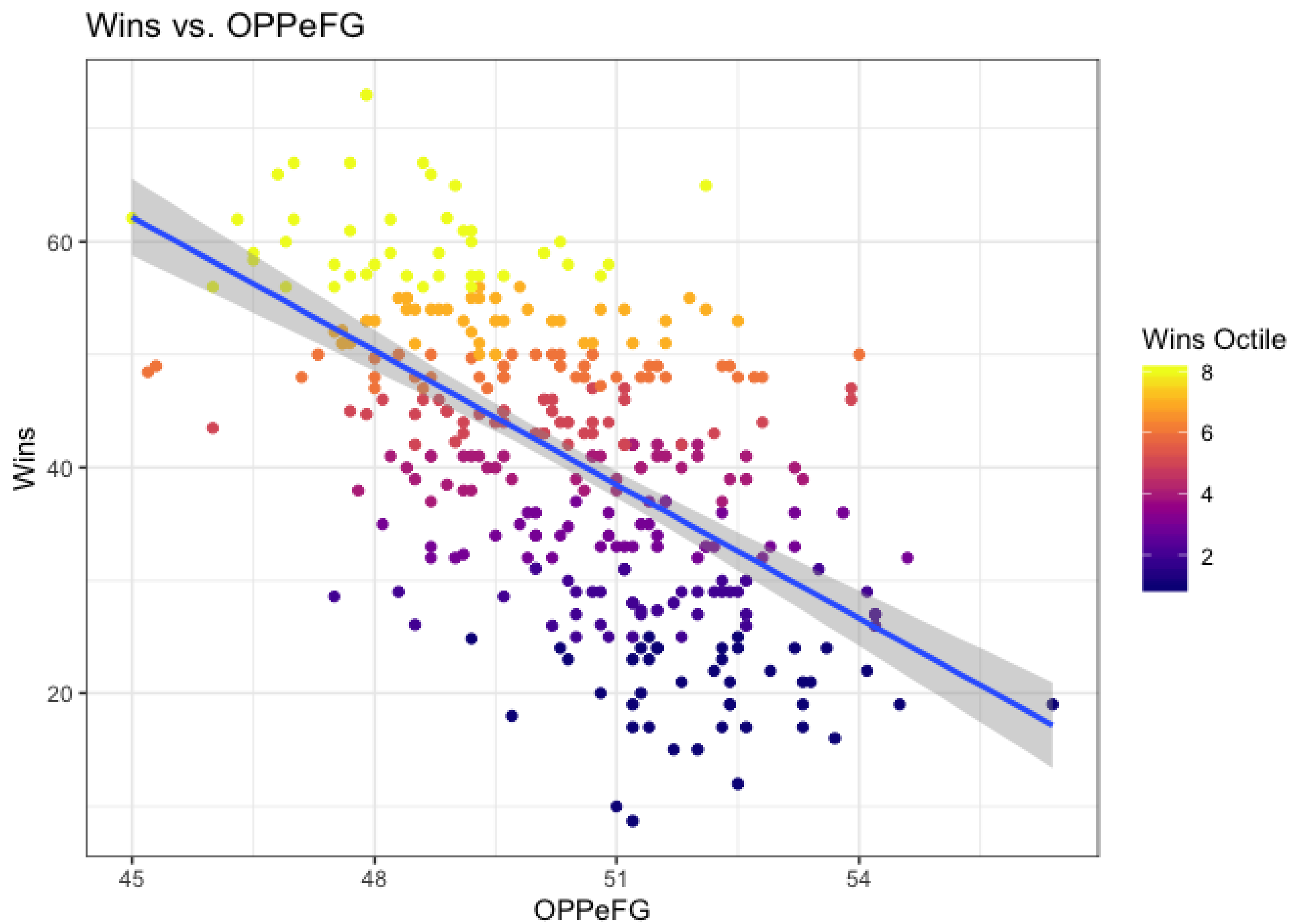

```
ggplot(four_factors_scaled, aes(OREB, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. OREB") +
  xlab("OREB") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```



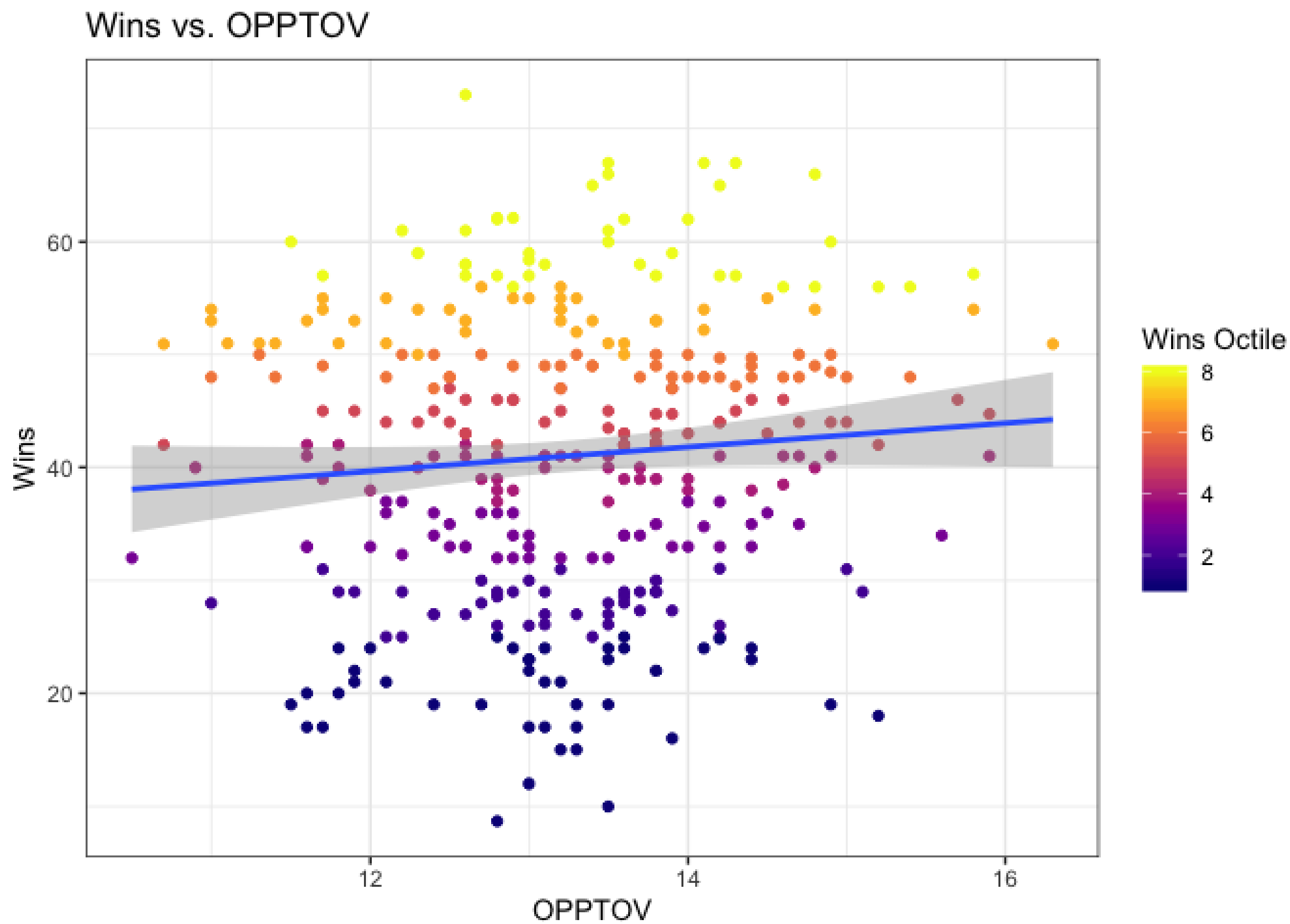

```
ggplot(four_factors_scaled, aes(FTF, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. FTF") +
  xlab("FTF") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```



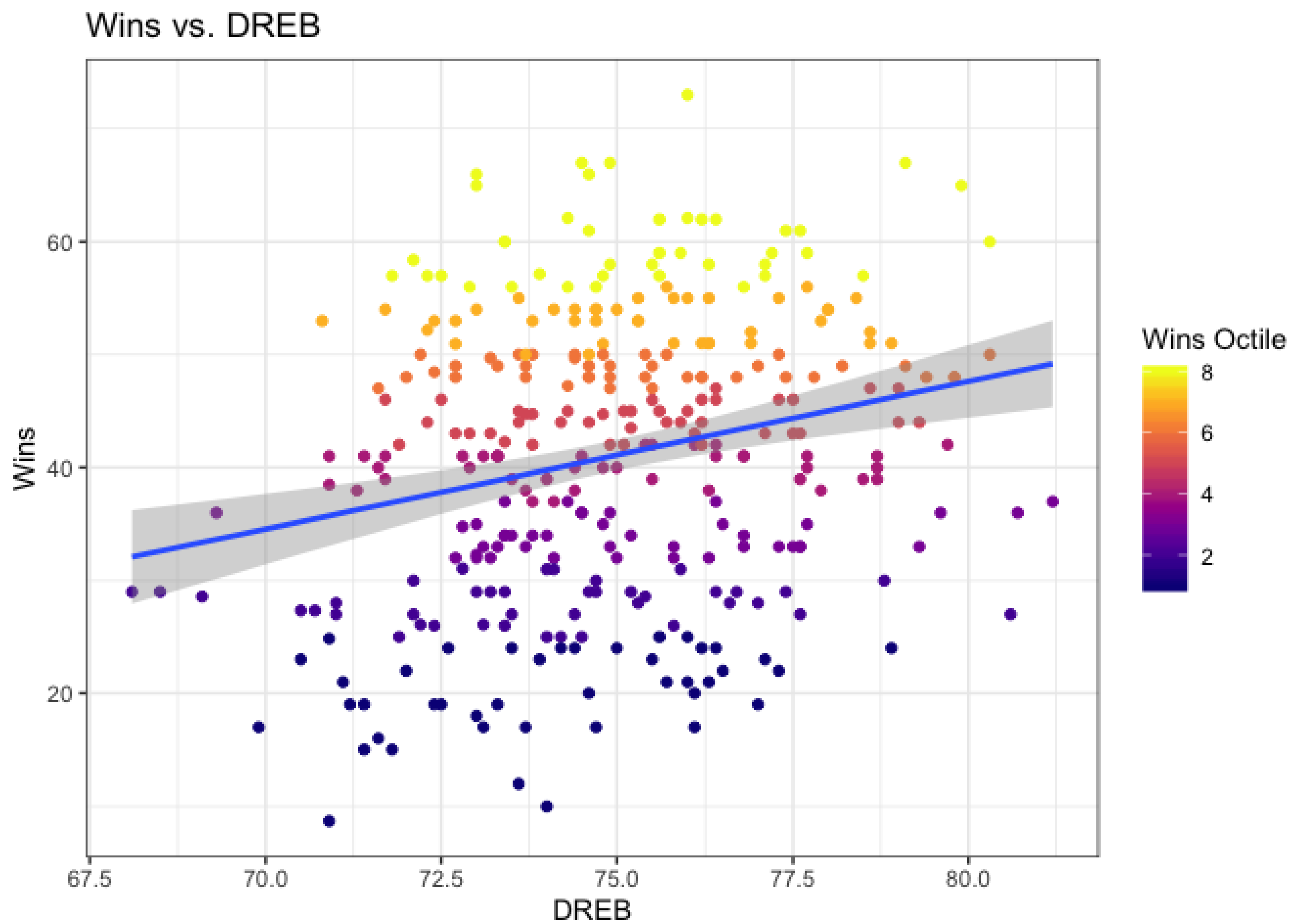

```
ggplot(four_factors_scaled, aes(OPPeFG, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. OPPeFG") +
  xlab("OPPeFG") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```



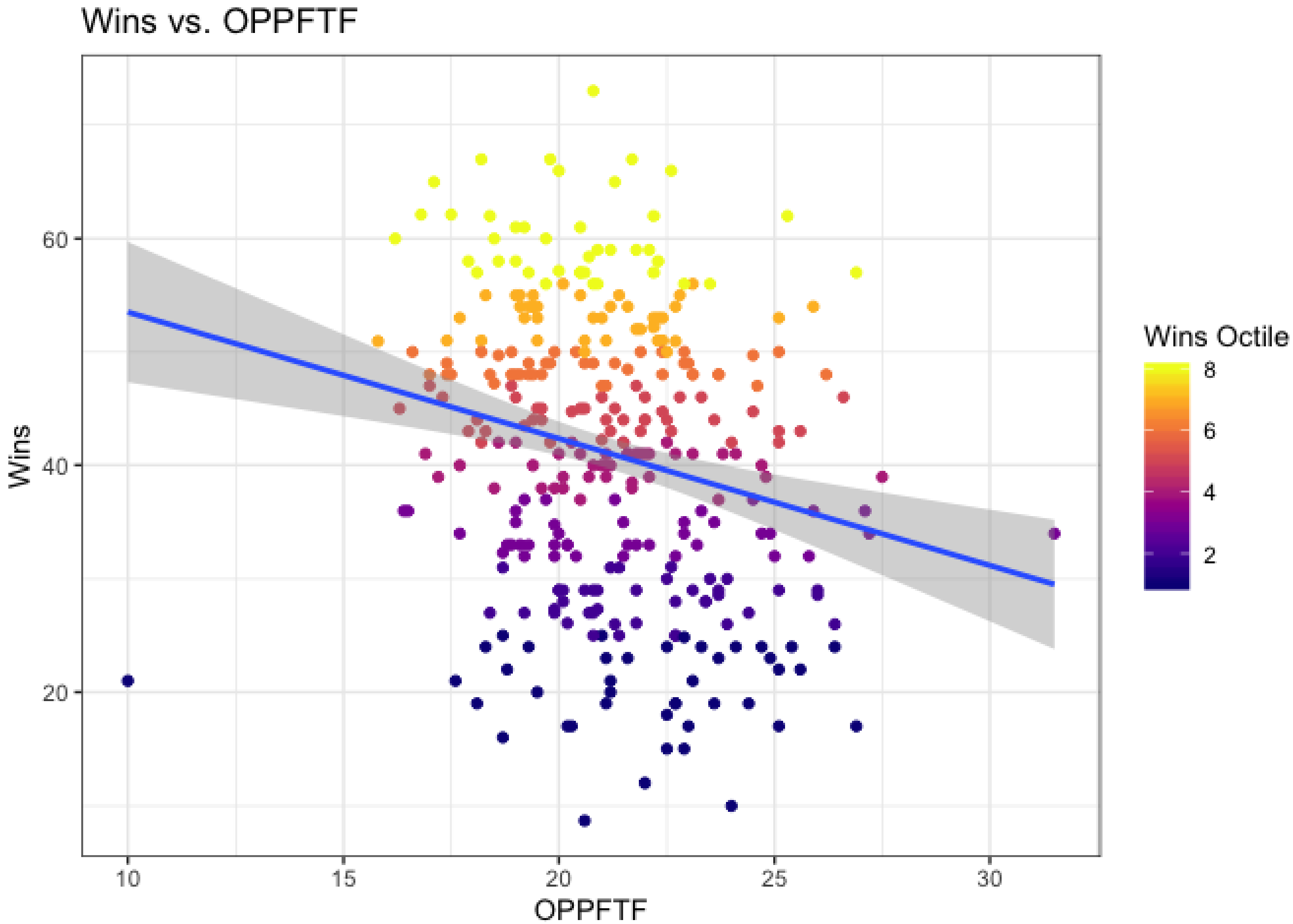

```
ggplot(four_factors_scaled, aes(OPPTOV, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. OPPTOV") +
  xlab("OPPTOV") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```




```
ggplot(four_factors_scaled, aes(DREB, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. DREB") +
  xlab("DREB") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```




```
ggplot(four_factors_scaled, aes(OPPFTF, WINS)) +
  geom_point(aes(color = wins_octile)) +
  geom_smooth(method ="lm") +
  ggtitle("Wins vs. OPPFTF") +
  xlab("OPPFTF") +
  ylab("Wins") +
  labs(color = "Wins Octile") +
  coord_cartesian() +
  scale_color_viridis(option = "C") +
  theme_bw()
```

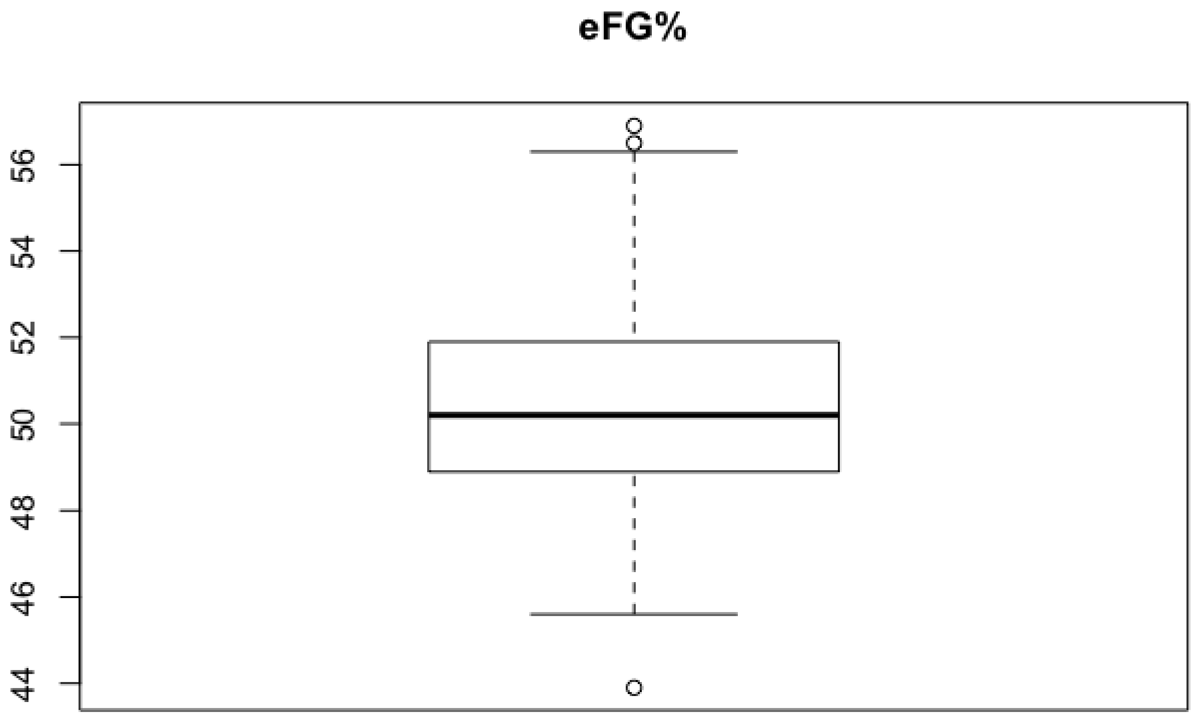



```
new = t(four_factors_scaled)
```

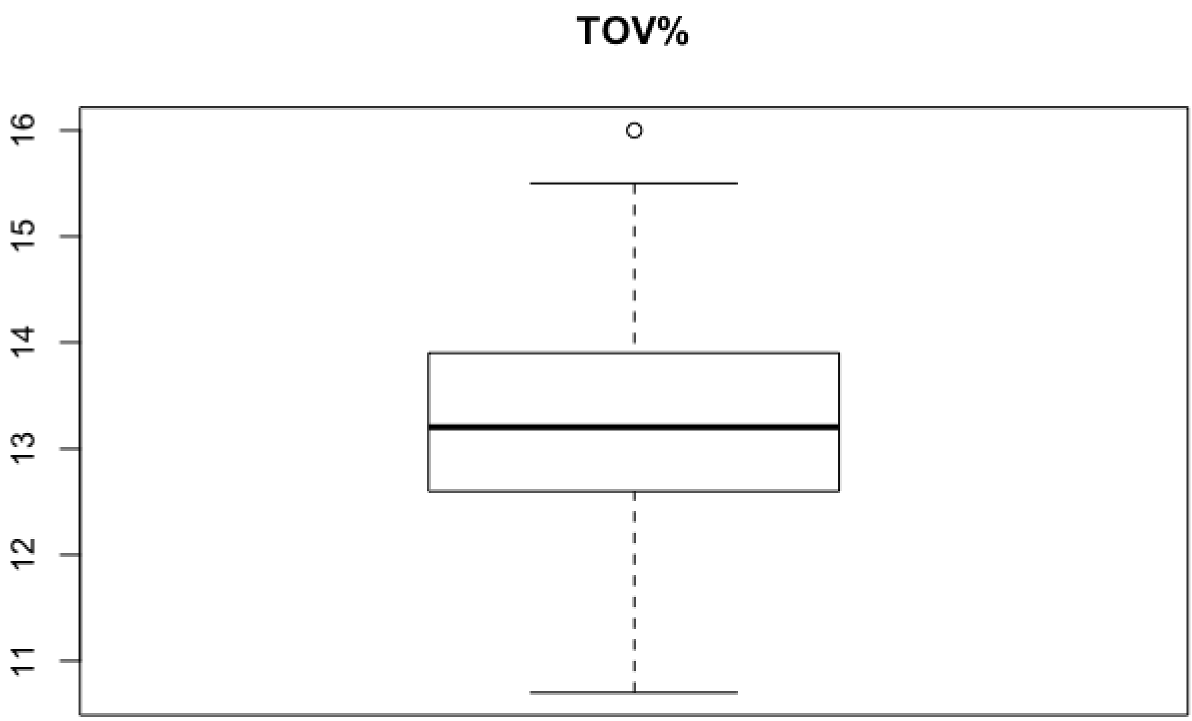
2.2 Box Plots

Checking graphically to see if there are outliers in our predictor variables

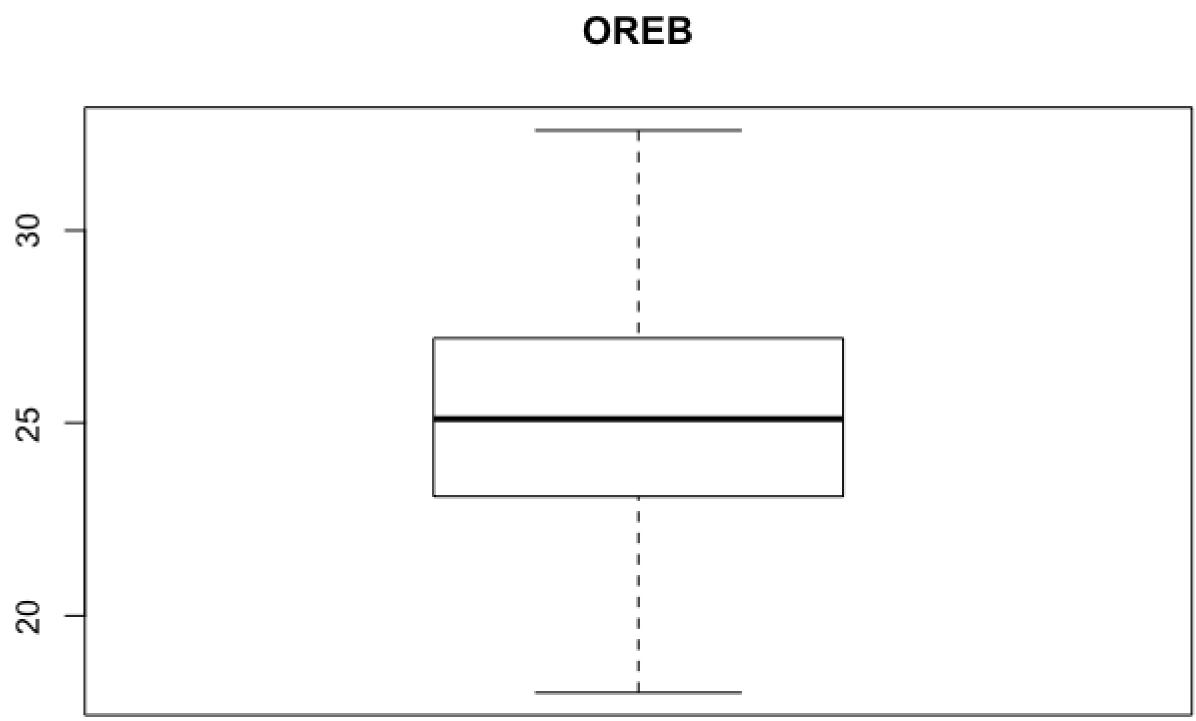
```
boxplot(four_factors_scaled$eFG, main="eFG%")
```



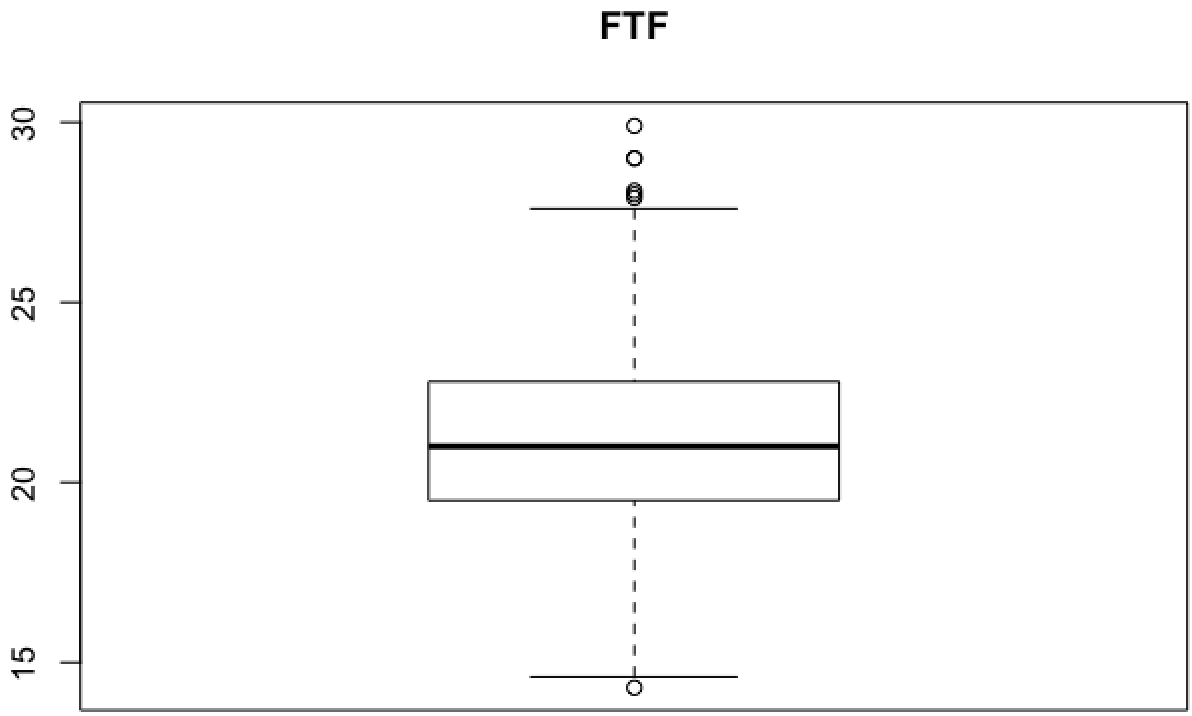

```
boxplot(four_factors_scaled$TOV, main="TOV%")
```



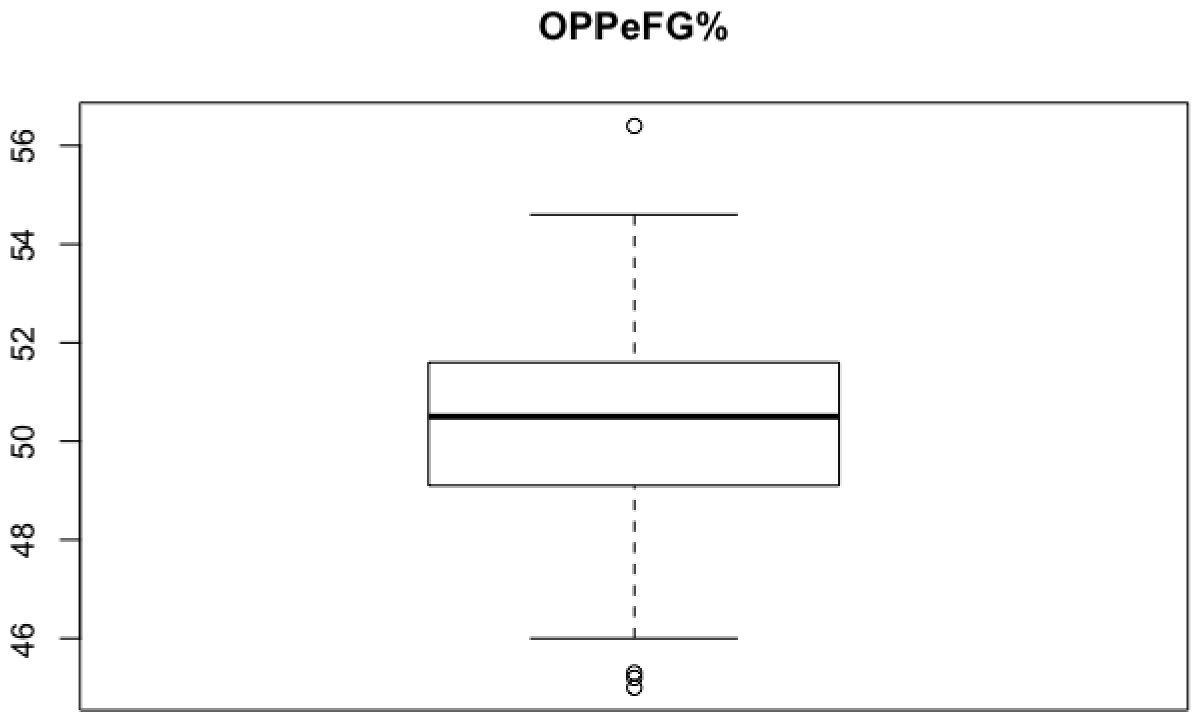

```
boxplot(four_factors_scaled$OREB, main="OREB")
```



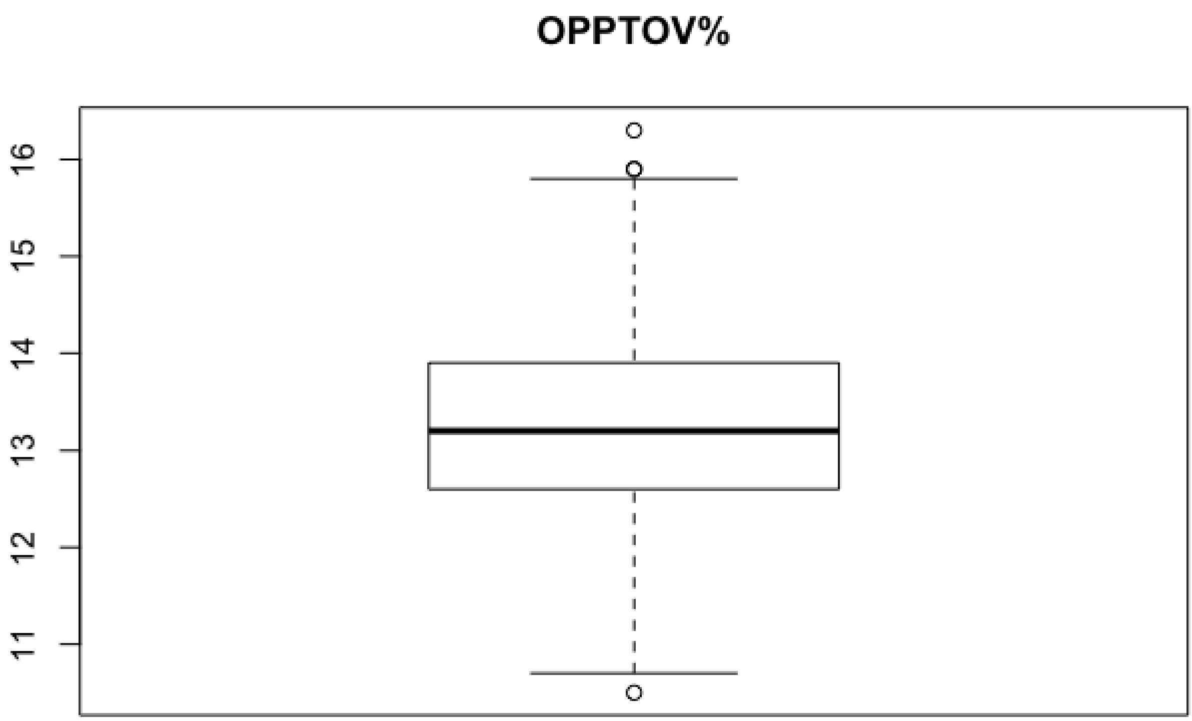

```
boxplot(four_factors_scaled$FTF, main="FTF")
```



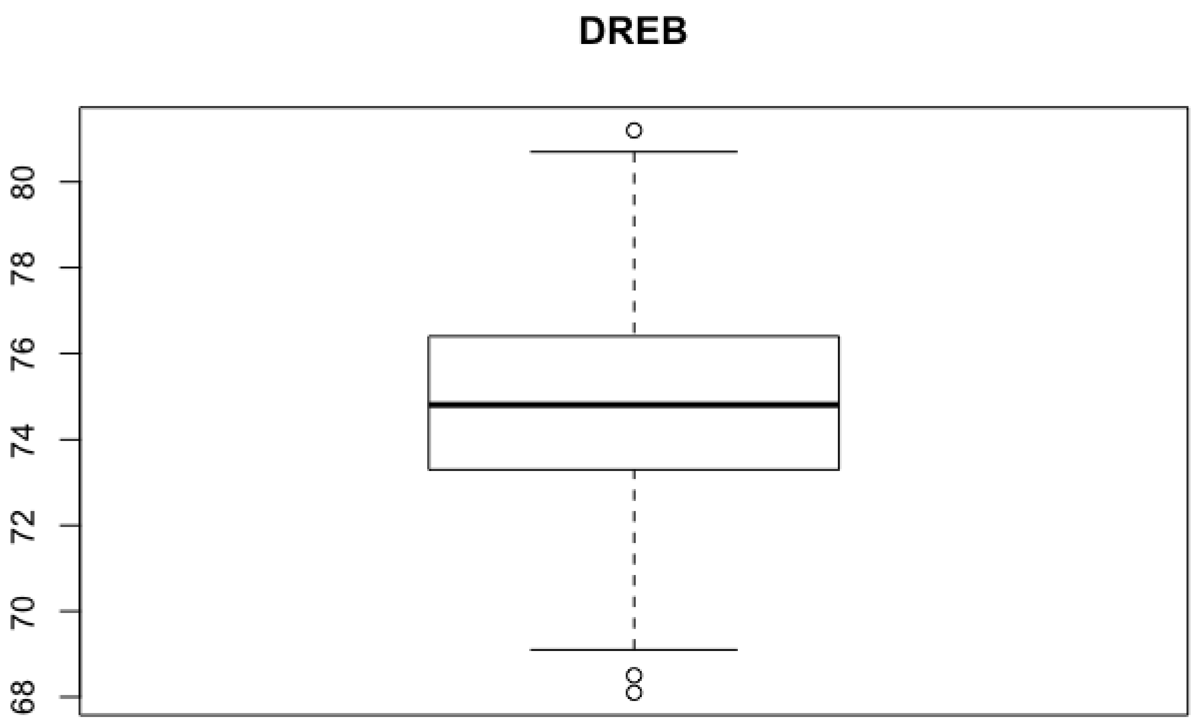

```
boxplot(four_factors_scaled$OPPeFG, main="OPPeFG%")
```



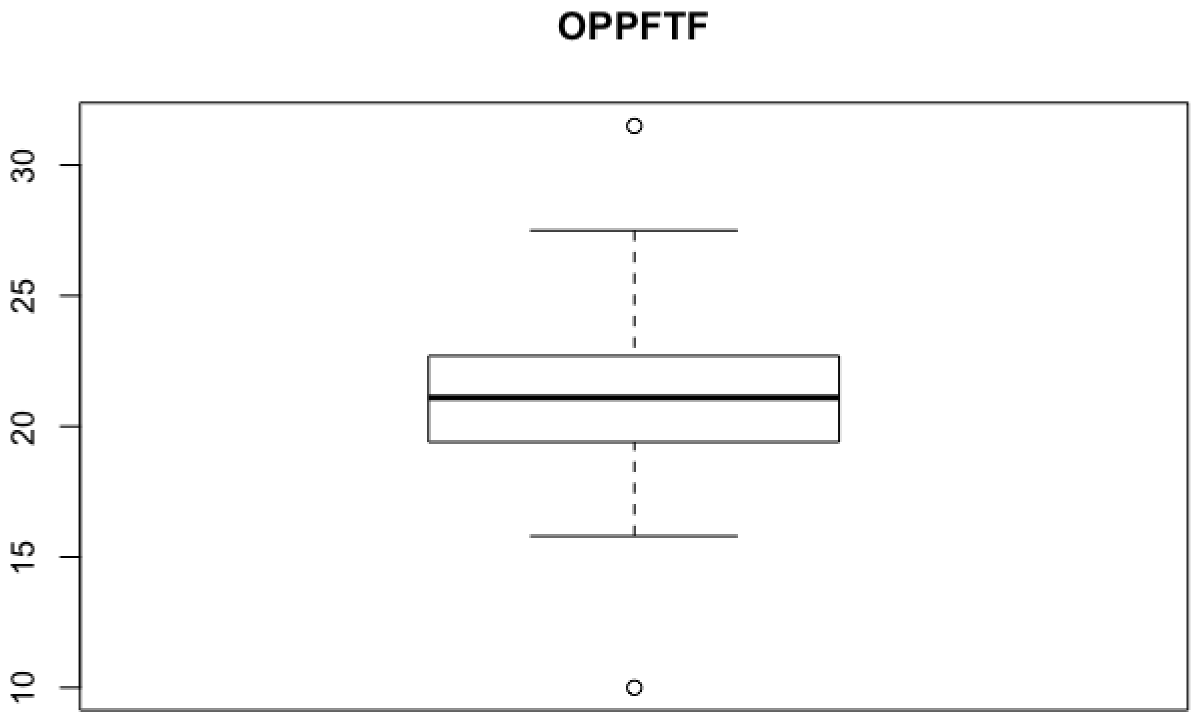

```
boxplot(four_factors_scaled$OPPTOV, main="OPPTOV%")
```



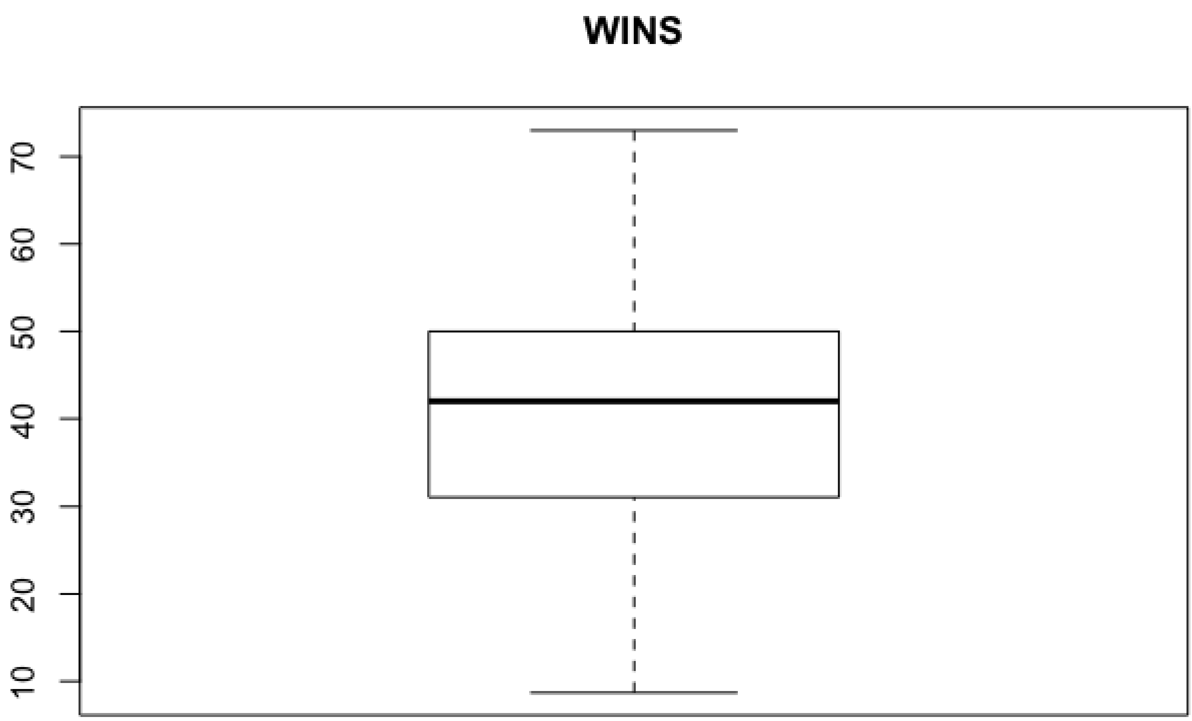

```
boxplot(four_factors_scaled$DREB, main="DREB")
```




```
boxplot(four_factors_scaled$OPPFTF, main="OPPFTF")
```

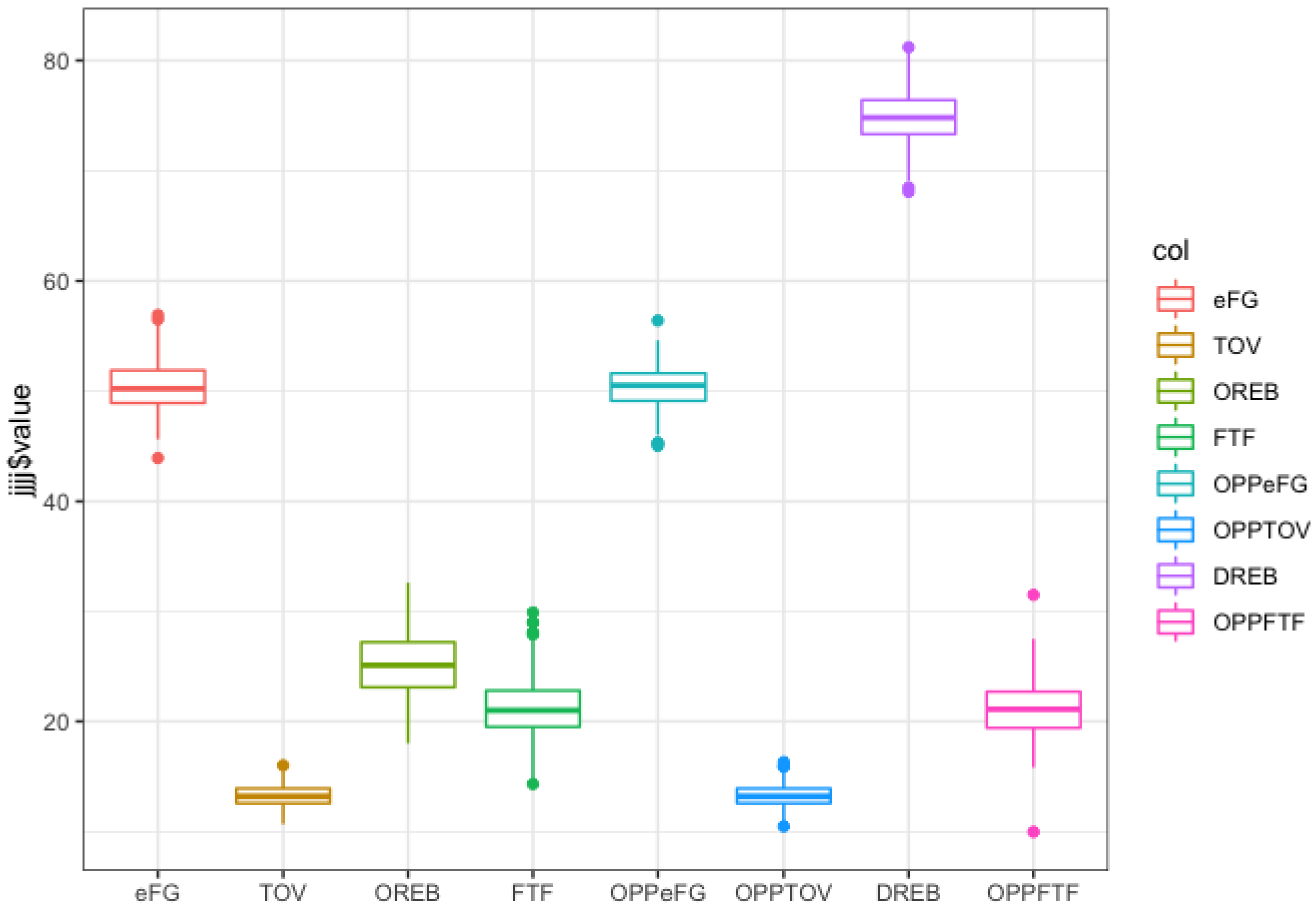



```
boxplot(four_factors_scaled$WINS, main="WINS")
```



Boxplot for all predictors

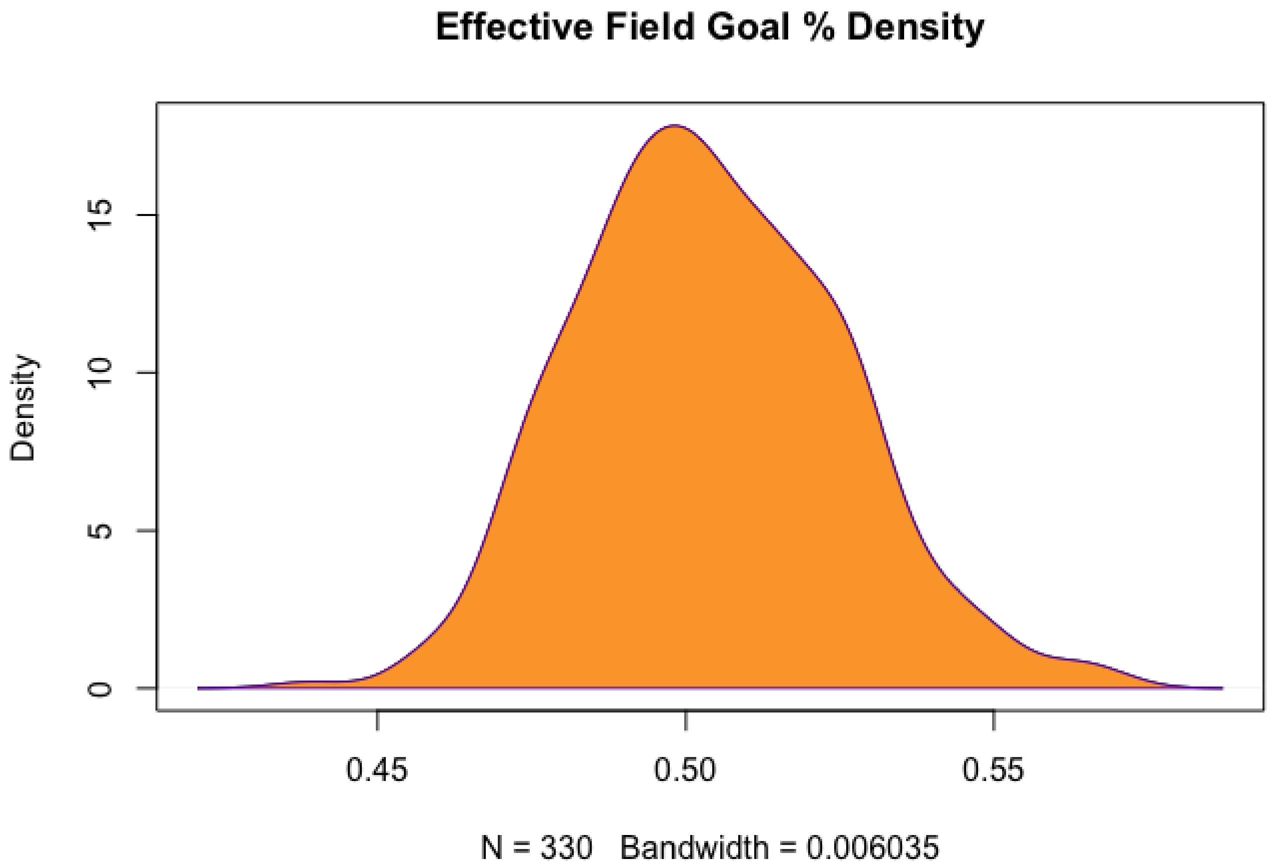
```
fact_eFG = data.frame("col" = rep("eFG", 330))
fact_TOV = data.frame("col" = rep("TOV", 330))
fact_OREB = data.frame("col" = rep("OREB", 330))
fact_FTF = data.frame("col" = rep("FTF", 330))
fact_OPPeFG = data.frame("col" = rep("OPPeFG", 330))
fact_OPPTOV = data.frame("col" = rep("OPPTOV", 330))
fact_DREB = data.frame("col" = rep("DREB", 330))
fact_OPPFTF = data.frame("col" = rep("OPPFTF", 330))

jjjj <- rbind(fact_eFG, fact_TOV, fact_OREB, fact_FTF, fact_OPPeFG, fact_OPPTOV, fact_DR
EB, fact_OPPFTF)

jjjj$value[1:330] <- four_factors_scaled$eFG
jjjj$value[331:660] <- four_factors_scaled$TOV
jjjj$value[661:990] <- four_factors_scaled$OREB
jjjj$value[991:1320] <- four_factors_scaled$FTF
jjjj$value[1321:1650] <- four_factors_scaled$OPPeFG
jjjj$value[1651:1980] <- four_factors_scaled$OPPTOV
jjjj$value[1981:2310] <- four_factors_scaled$DREB
jjjj$value[2311:2640] <- four_factors_scaled$OPPFTF

#jjjj

h <- ggplot(data=jjjj, aes(x=jjjj$col, y=jjjj$value, color = col)) +
geom_boxplot() +
xlab("Factor") +
theme(legend.position="none") +
xlab("") +
theme_bw()

h
```



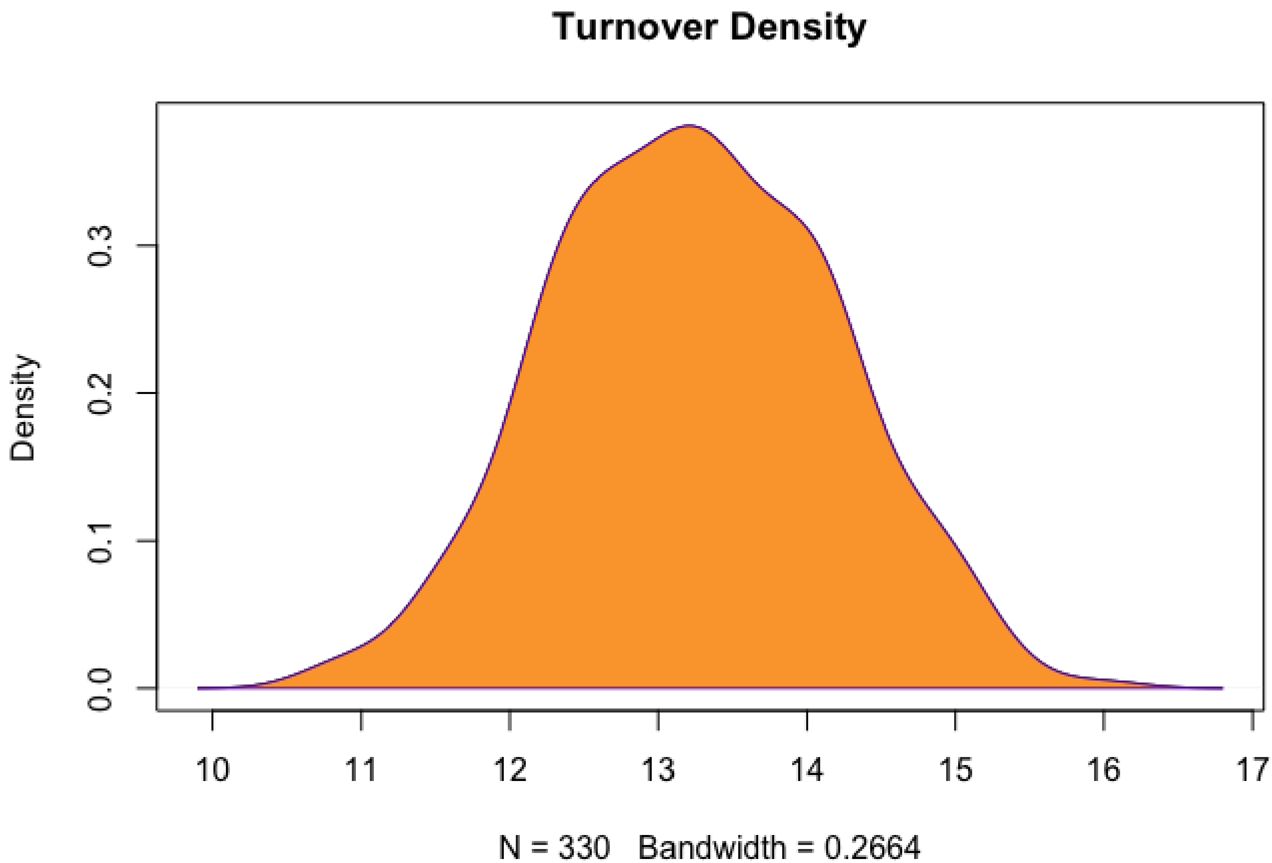
2.3 Density Plots

Checking graphically to see if our feature variables have a normal distribution

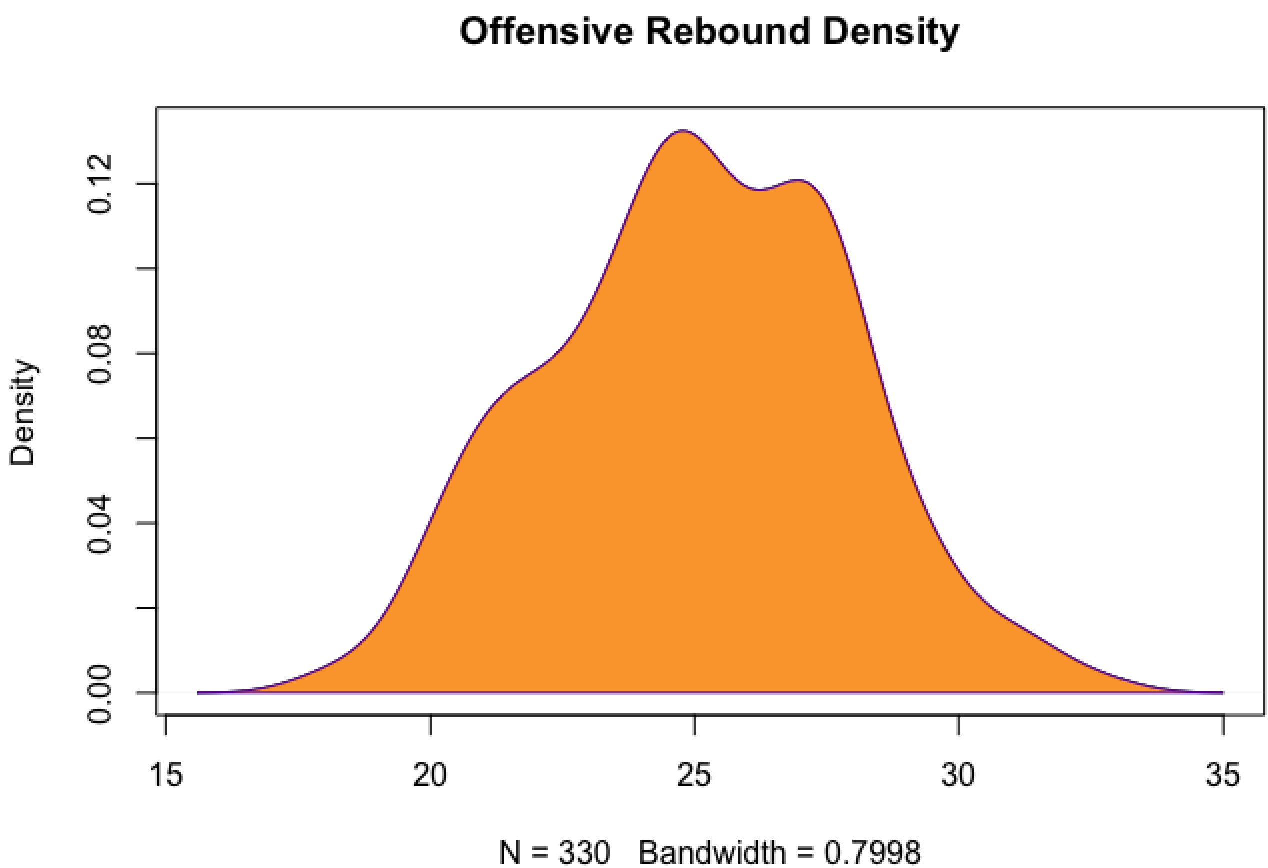
```
dens_eFG <- density(four_factors_new$eFG)
plot(dens_eFG, main = "Effective Field Goal % Density")
polygon(dens_eFG, col="#fca538", border="#6721a7")
```



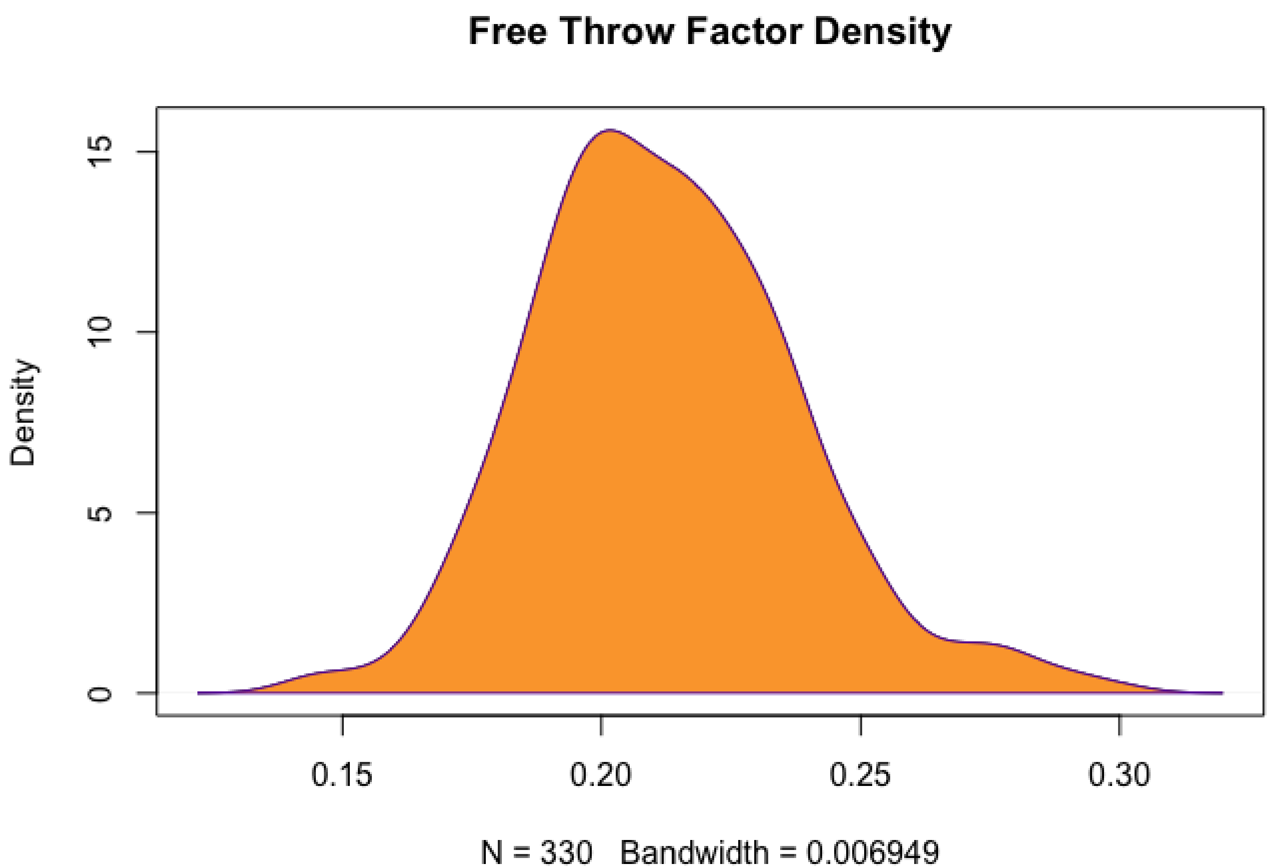

```
dens_TOV <- density(four_factors_new$TOV)
plot(dens_TOV, main = "Turnover Density")
polygon(dens_TOV, col="#fca538", border="#6721a7")
```



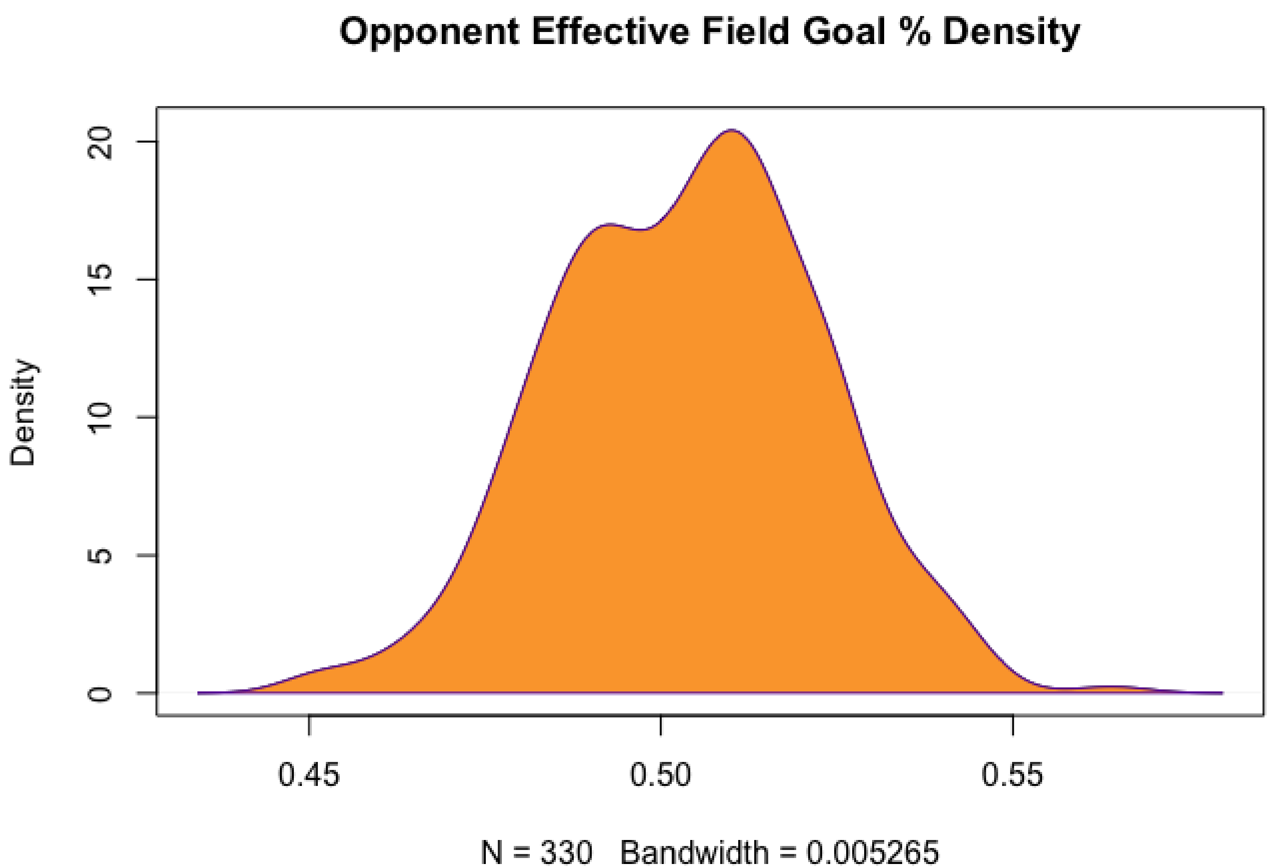

```
dens_OREB <- density(four_factors_new$OREB)
plot(dens_OREB, main = "Offensive Rebound Density")
polygon(dens_OREB, col="#fca538", border="#6721a7")
```



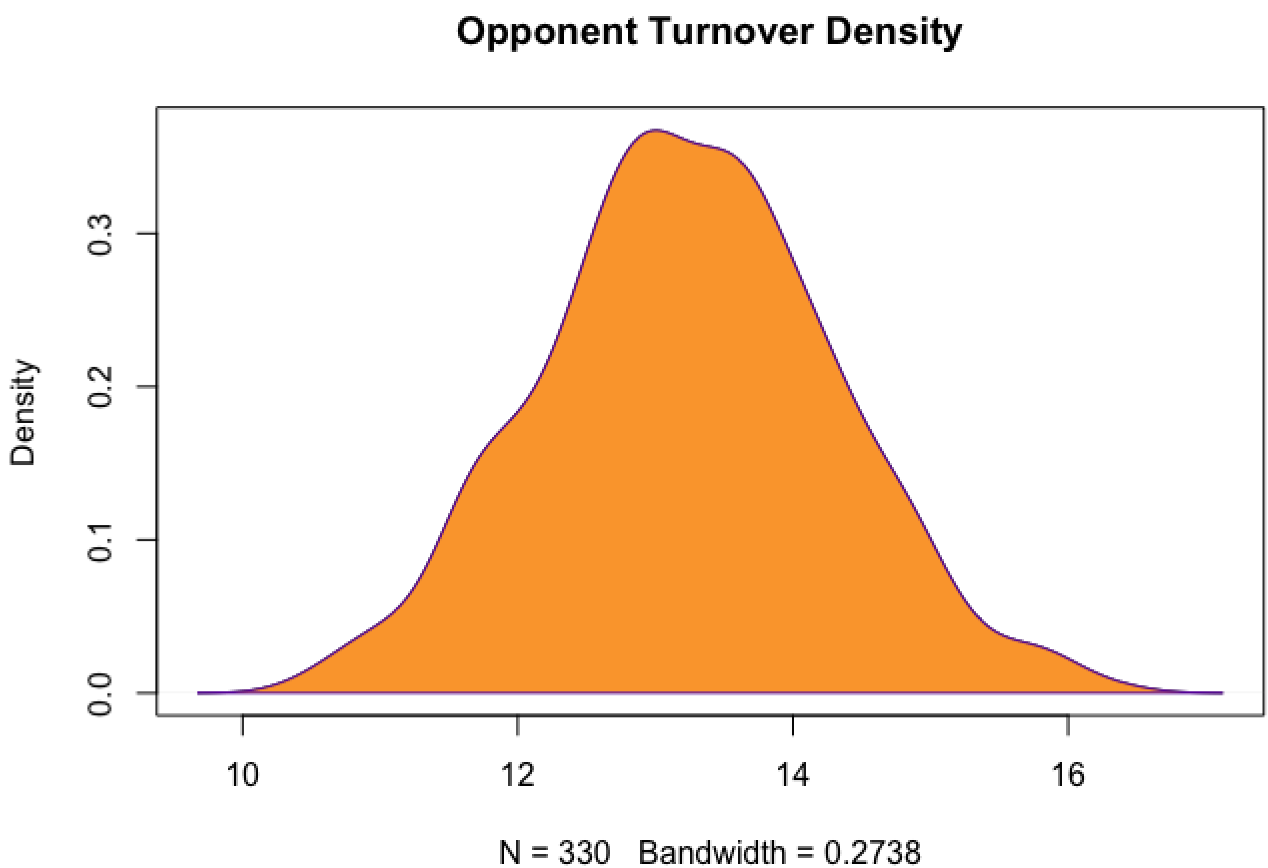

```
dens_FTF <- density(four_factors_new$FTF)
plot(dens_FTF, main = "Free Throw Factor Density")
polygon(dens_FTF, col="#fca538", border="#6721a7")
```



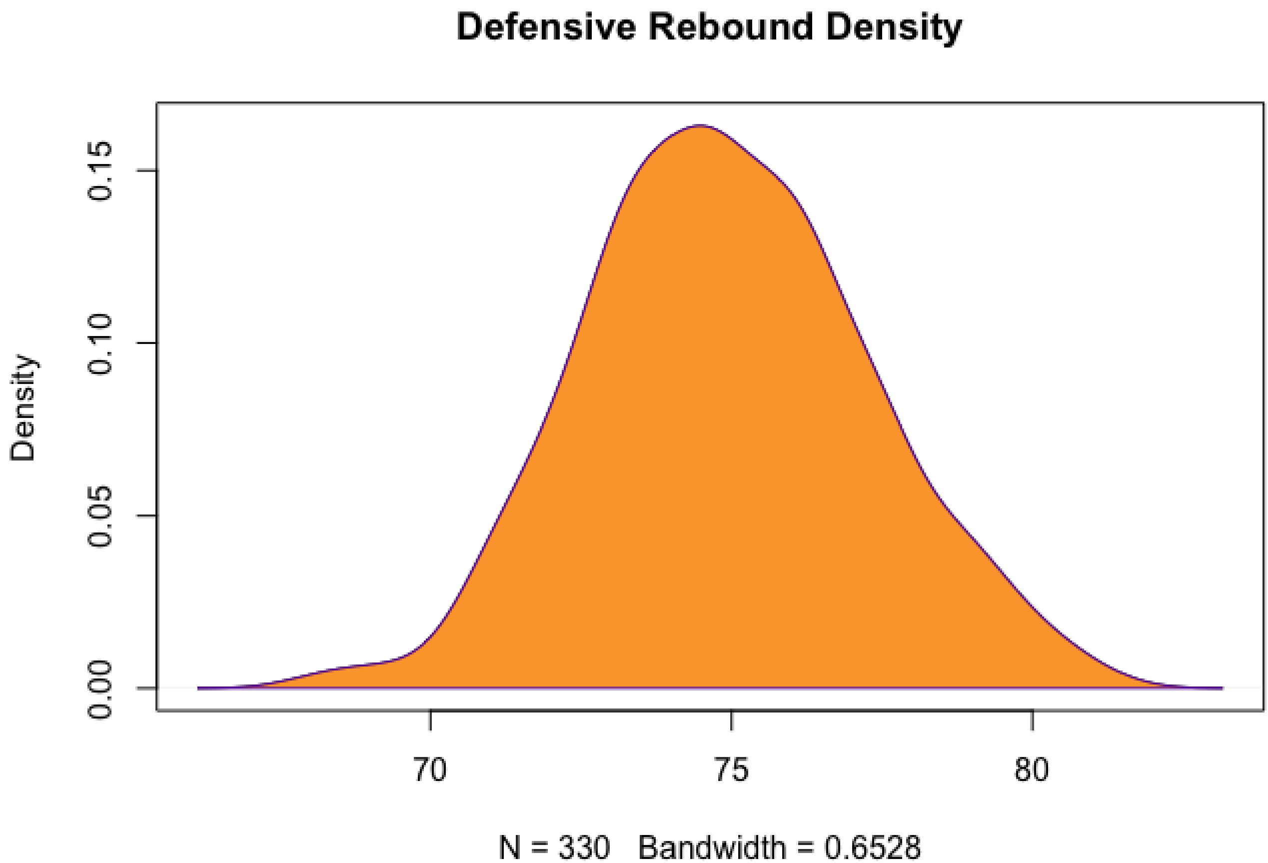

```
dens_OPPeFG <- density(four_factors_new$OPPeFG)
plot(dens_OPPeFG, main = "Opponent Effective Field Goal % Density")
polygon(dens_OPPeFG, col="#fca538", border="#6721a7")
```



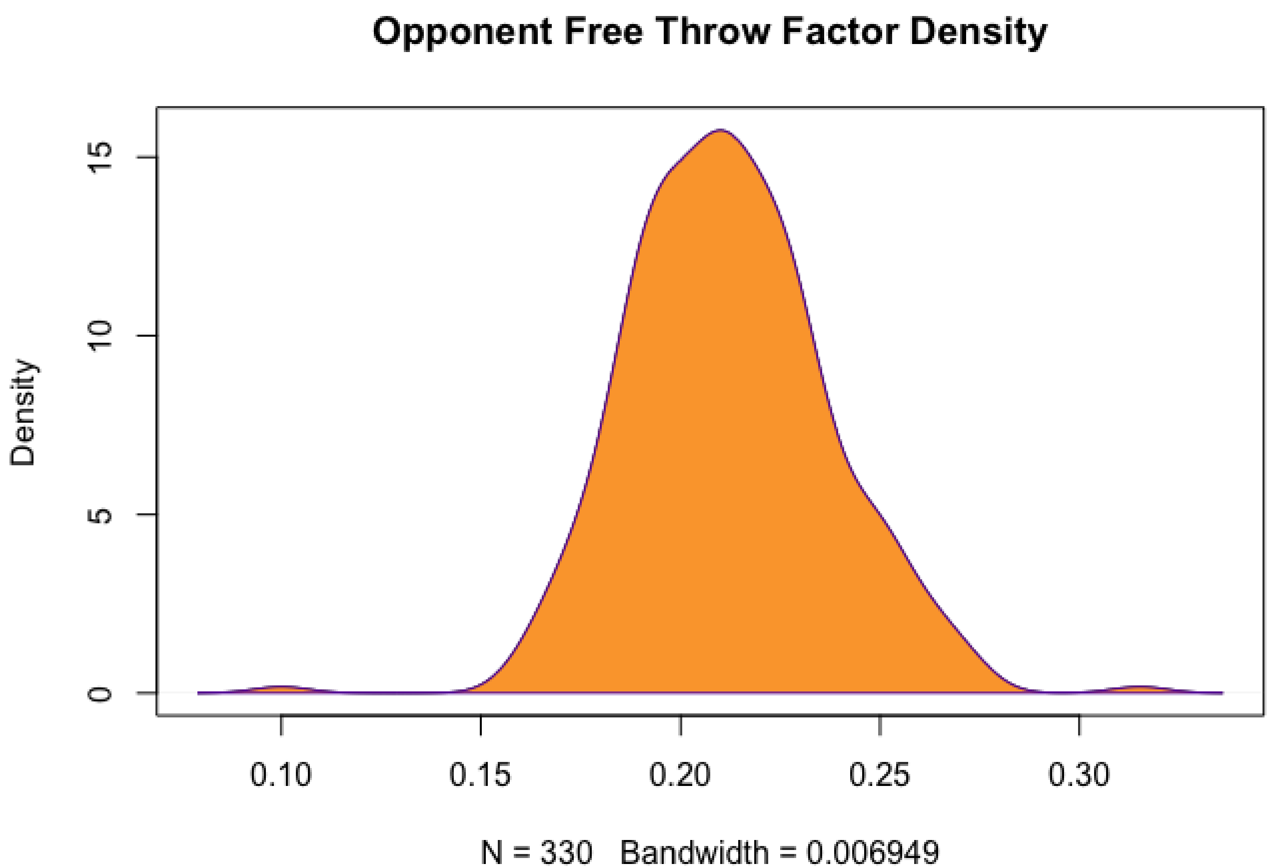

```
dens_OPPTOV <- density(four_factors_new$OPPTOV)
plot(dens_OPPTOV, main = "Opponent Turnover Density")
polygon(dens_OPPTOV, col="#fca538", border="#6721a7")
```



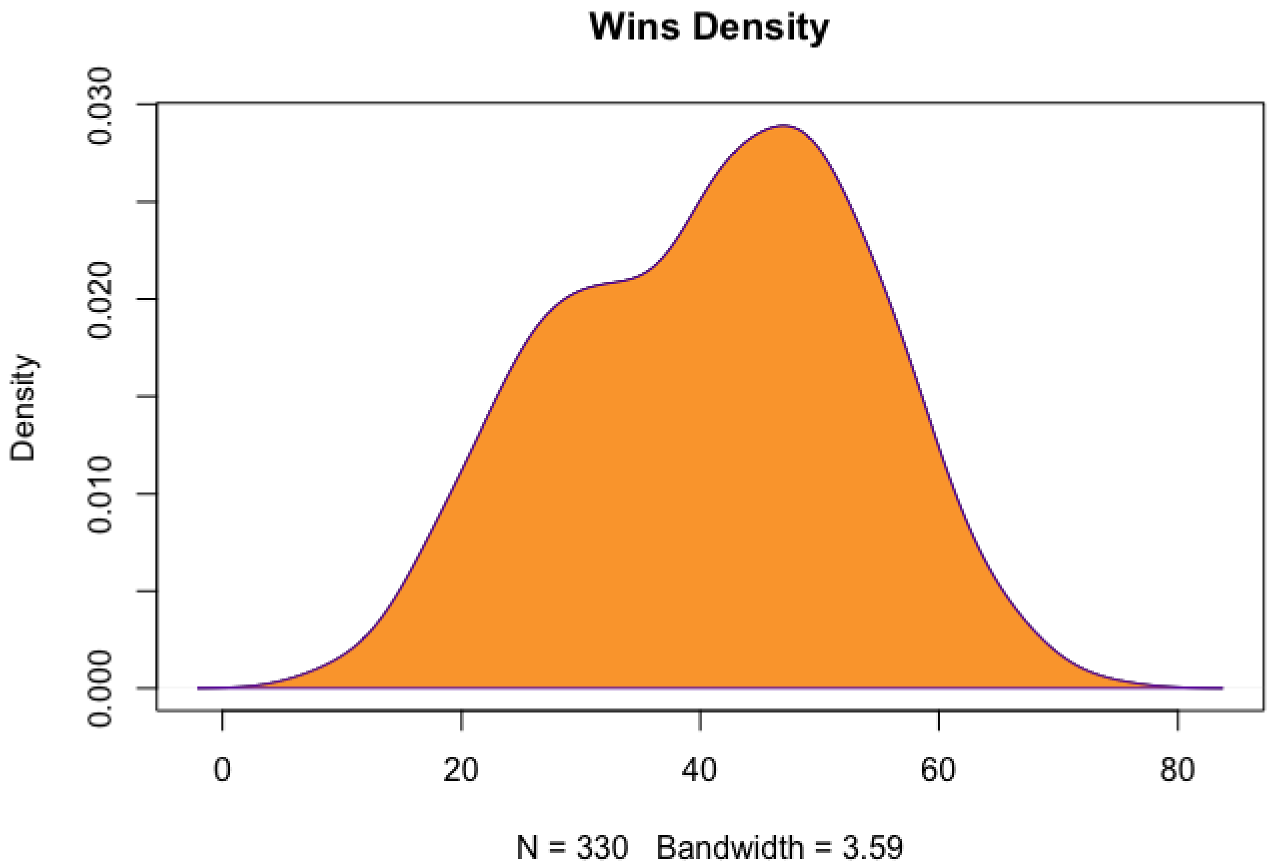

```
dens_DREB <- density(four_factors_new$DREB)
plot(dens_DREB, main = "Defensive Rebound Density")
polygon(dens_DREB, col="#fca538", border="#6721a7")
```




```
dens_OPPFTF <- density(four_factors_new$OPPFTF)
plot(dens_OPPFTF, main = "Opponent Free Throw Factor Density")
polygon(dens_OPPFTF, col="#fca538", border="#6721a7")
```

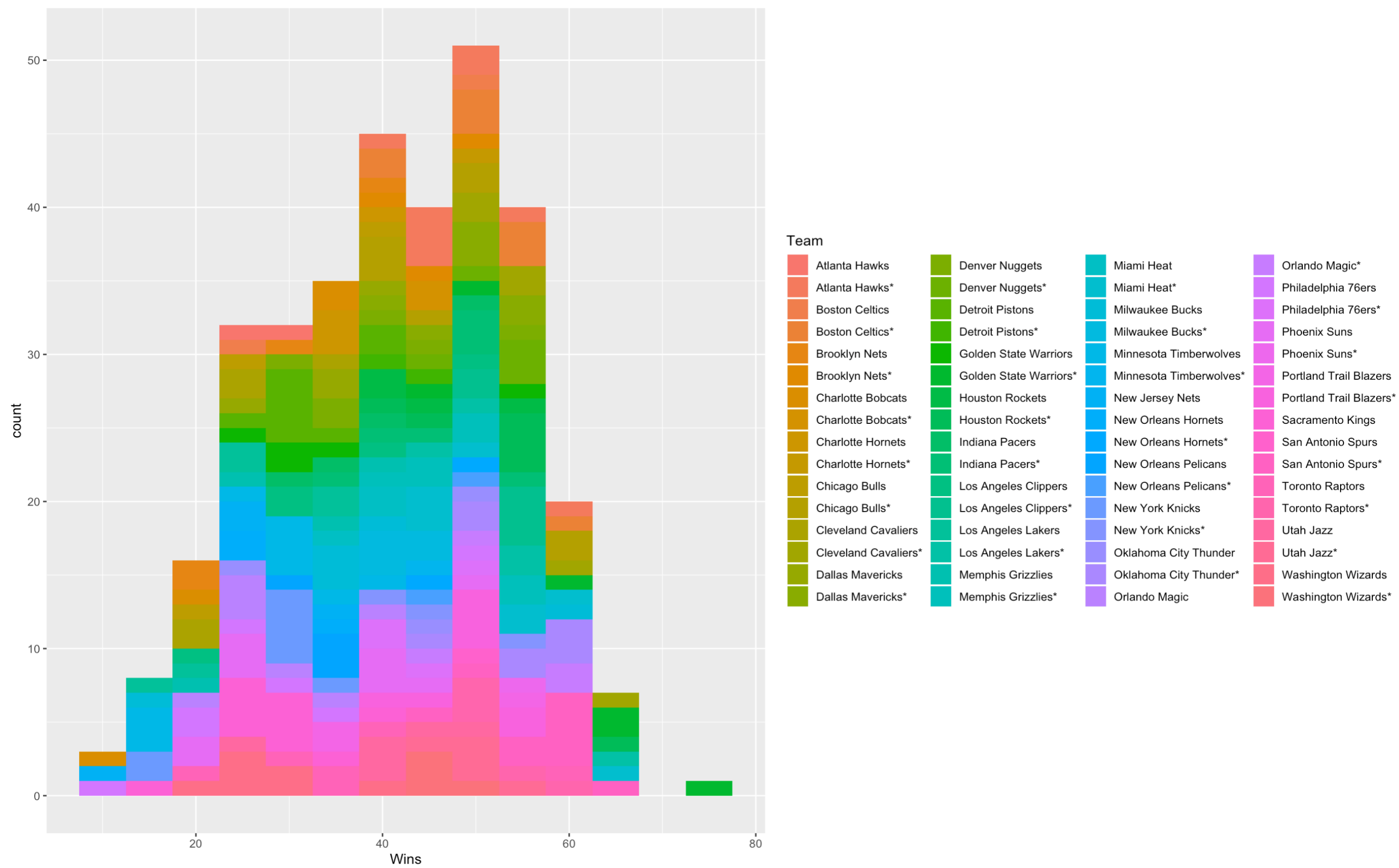



```
dens_WINS <- density(four_factors_new$WINS)
plot(dens_WINS, main = "Wins Density")
polygon(dens_WINS, col="#fca538", border="#6721a7")
```



2.4 Histogram of wins

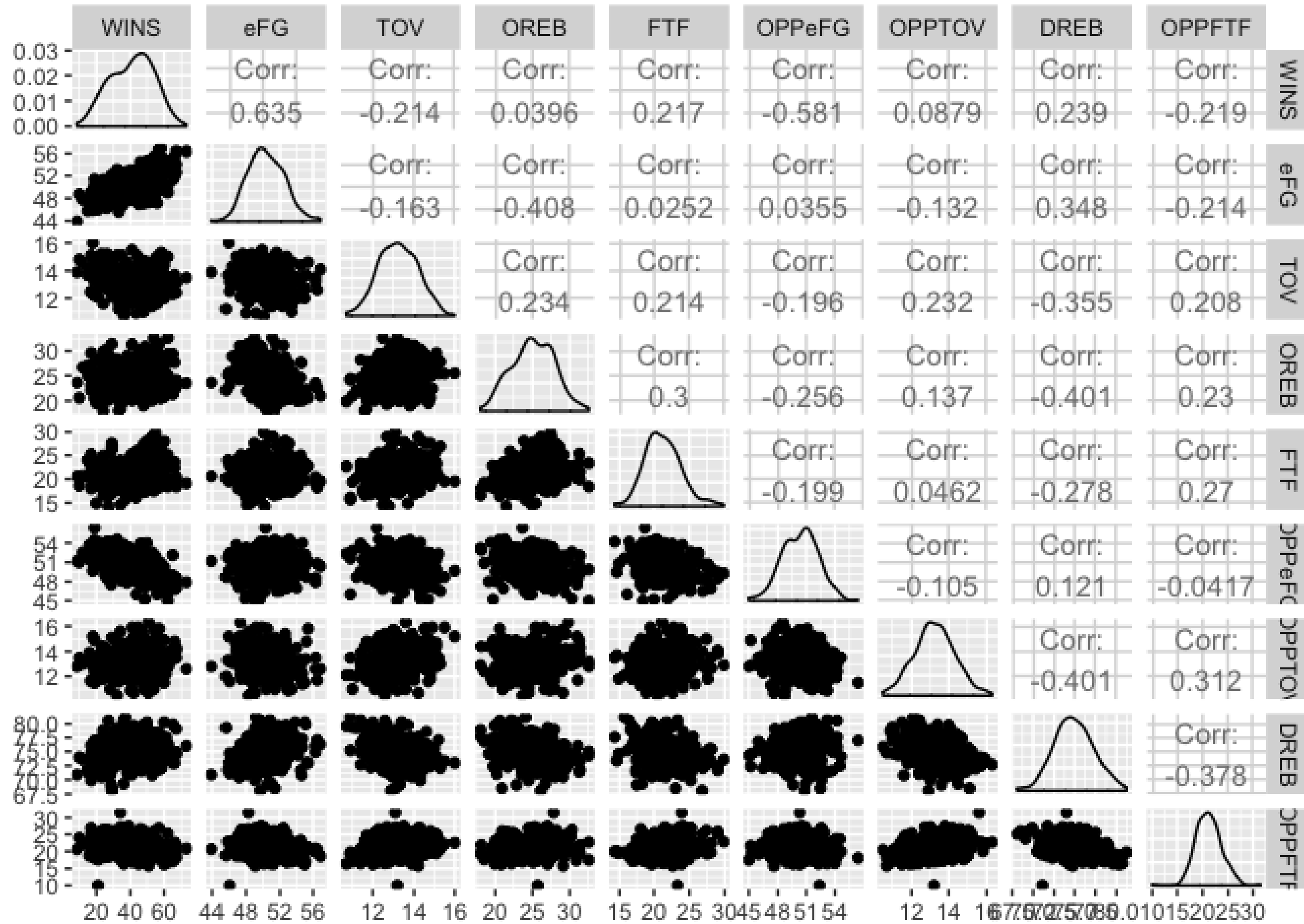
```
fig2 = ggplot(four_factors_scaled, aes(x=WINS, fill=Team)) + geom_histogram(binwidth=5)
 + labs(x="Wins")
```



2.5 Pairplot for all variables

Checking graphically pairwise relationships in our variables

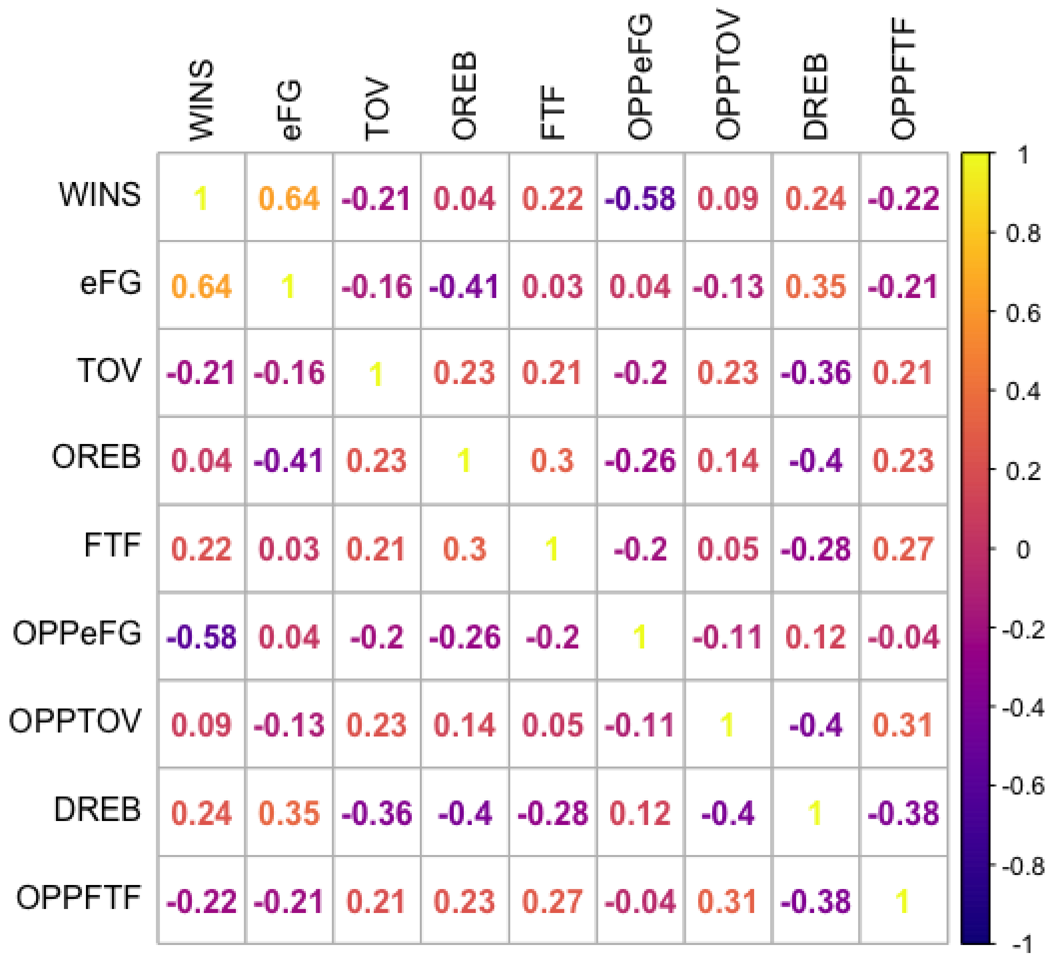
```
ggpairs(four_factors_scaled[c("WINS", "eFG", "TOV", "OREB", "FTF", "OPPeFG", "OPPTOV",
"DREB", "OPPFTF")])
```



2.6 Correlation plot for all variables

Checking graphically for multicolinearity between our different feature variables

```
? corrplot

C <- cor(four_factors_scaled[c("WINS", "eFG", "TOV", "OREB", "FTF", "OPPeFG", "OPPTOV",
"DREB", "OPPFTF")])

corrplot(C, method = "number", col = plasma(256), tl.col = "black")
```



There doesn't seem to be any moderate or strong multicolinarity to be aware of in the matrix

3. Modeling and Analysis

3.1 Splitting our dataset into train and test subsets

```
set.seed(8)
rows <- sample(1:nrow(four_factors_scaled), 0.7*nrow(four_factors_scaled))
ff_train = four_factors_scaled[rows,]
ff_test = four_factors_scaled[-rows,]
dim(ff_train)
```

```
## [1] 230  14
```

```
dim(ff_test)
```

```
## [1] 100  14
```

3.2 Creating a Multiple Linear Regression model regressing the Four Factors on `WINS` using the test dataset

```
lmWINS <- lm(WINS ~ eFG + TOV + OREB + FTF + OPPeFG + OPPTOV + DREB + OPPFTF, data = ff_
train)

summary(lmWINS)
```

```
##
## Call:
## lm(formula = WINS ~ eFG + TOV + OREB + FTF + OPPeFG + OPPTOV +
##     DREB + OPPFTF, data = ff_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.6982  -2.1067  -0.1126   2.0078   9.0728
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -53.18468   14.54796  -3.656  0.00032 ***
## eFG           3.89934    0.11042  35.314  < 2e-16 ***
## TOV          -3.39168    0.24879 -13.633  < 2e-16 ***
## OREB          1.11070    0.09013  12.323  < 2e-16 ***
## FTF           0.70791    0.09304   7.609 7.89e-13 ***
## OPPeFG       -3.76483    0.11439 -32.913  < 2e-16 ***
## OPPTOV        2.90168    0.21229  13.668  < 2e-16 ***
## DREB          0.88666    0.11764   7.537 1.22e-12 ***
## OPPFTF       -0.72897    0.10377  -7.025 2.62e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.158 on 221 degrees of freedom
## Multiple R-squared:  0.9435, Adjusted R-squared:  0.9414
## F-statistic: 461.2 on 8 and 221 DF,  p-value: < 2.2e-16
```

3.2.1 Interpretation of F-statistic, P-value and R-squared value for the **lmWINS** model

From the F-statistic and p-value of our **lmWINS** model, we can reject the null hypothesis that our predictor variables have no effect on `WINS`. There is strong evidence to conclude there is a relationship between our predictor and response variables.

Our model explains approximately 94.4% of the variation in `WINS` using `eFG`, `TOV`, `OREB`, `FTF`, `OPPeFG`, `OPPTOV`, `DREB` and `OPPFTF` as predictors.

3.2.2 Interpretation of coefficients for the **lmWINS** model

- An increase of 1 percentage point of `eFG` is associated with an average increase of 3.81 `WINS` , holding all else equal
- An increase of 1 percentage point of `TOV` is associated with an average decrease of 3.64 `WINS` , holding all else equal
- An increase of 1 percentage point of `OREB` is associated with an average increase of 1.11 `WINS` , holding all else equal
- An increase of 1 percentage point of `FTF` is associated with an average increase of 0.69 `WINS` , holding all else equal
- An increase of 1 percentage point of `OPPeFG` is associated with an average decrease of 3.86 `WINS` , holding all else equal
- An increase of 1 percentage point of `OPPTOV` is associated with an average increase of 2.99 `WINS` , holding all else equal
- An increase of 1 percentage point of `DREB` is associated with an average increase of 0.88 `WINS` , holding all else equal
- An increase of 1 percentage point of `OPPFTF` is associated with an average decrease of 0.81 `WINS` , holding all else equal

3.2.3 ANOVA table and confidence interval for the **lmWINS** model

```
anova(lmWINS)
```

```
## Analysis of Variance Table
##
## Response: WINS
##            Df  Sum Sq Mean Sq  F value    Pr(>F)
## eFG         1 16737.8 16737.8 1678.045 < 2.2e-16 ***
## TOV         1   264.3   264.3   26.502 5.813e-07 ***
## OREB        1  4722.3  4722.3  473.437 < 2.2e-16 ***
## FTF         1   519.4   519.4   52.076 8.457e-12 ***
## OPPeFG      1 12167.9 12167.9 1219.886 < 2.2e-16 ***
## OPPTOV      1  1037.7  1037.7  104.037 < 2.2e-16 ***
## DREB        1   860.4   860.4   86.260 < 2.2e-16 ***
## OPPFTF      1   492.2   492.2   49.349 2.618e-11 ***
## Residuals 221  2204.4    10.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(lmWINS)
```

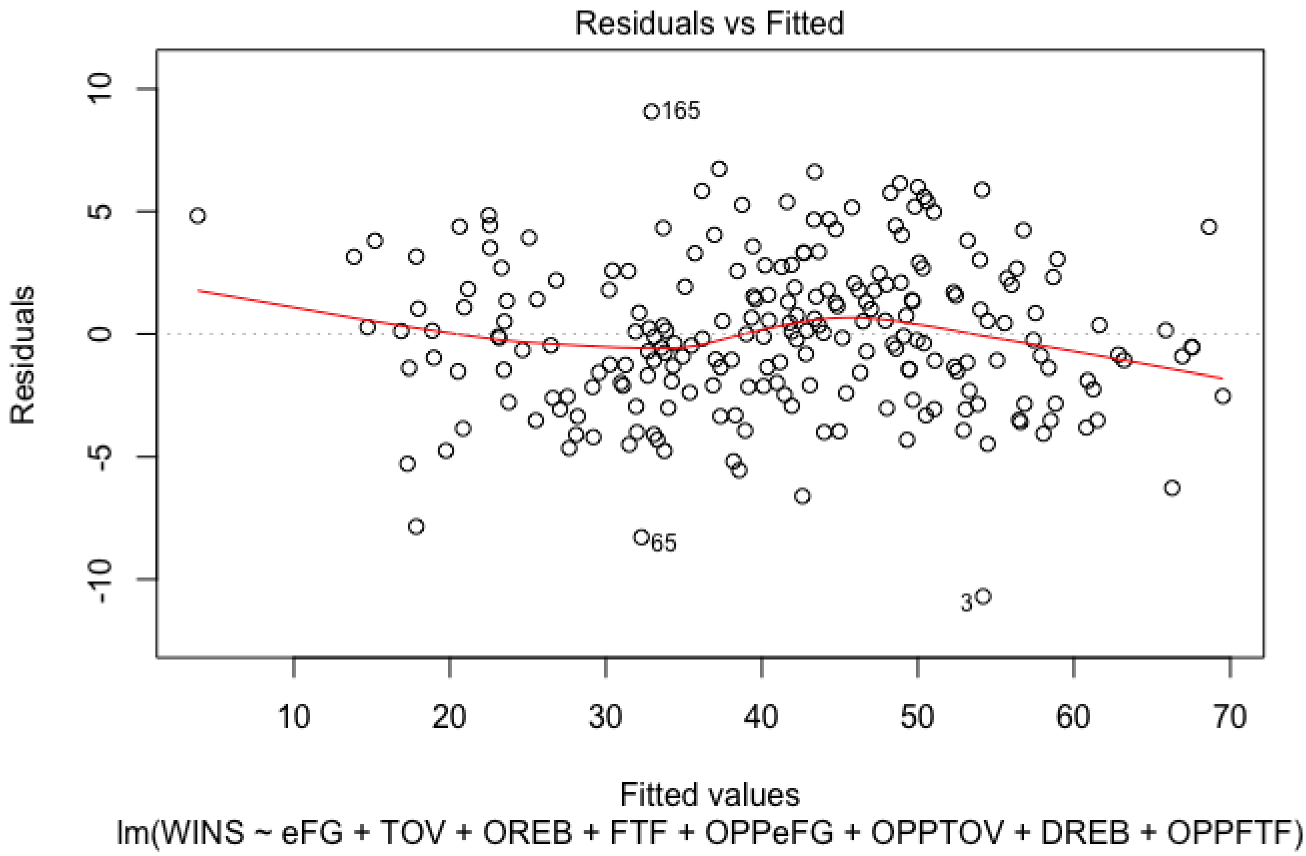
```
##                   2.5 %      97.5 %
## (Intercept) -81.8551642 -24.5141932
## eFG           3.6817271   4.1169430
## TOV          -3.8819723  -2.9013793
## OREB          0.9330770   1.2883264
## FTF           0.5245595   0.8912601
## OPPeFG       -3.9902637  -3.5394044
## OPPTOV        2.4833084   3.3200553
## DREB          0.6548226   1.1184874
## OPPFTF       -0.9334806  -0.5244679
```

From the ANOVA table, we can see that the F-statistic for each predictor variable is significant and adds prediction power to our model. All included variables are relevant to our model.

3.2.4 Checking to see if assumptions of linear regression are reasonably met for the **lmWINS** model

1. The relationship is linear

```
plot(lmWINS, 1)
```



Our residual plot is mostly flat. There are some points that skew the line, such as observation 116, 138 and 188, but overall the relationship is linear.

2. Independence of error terms

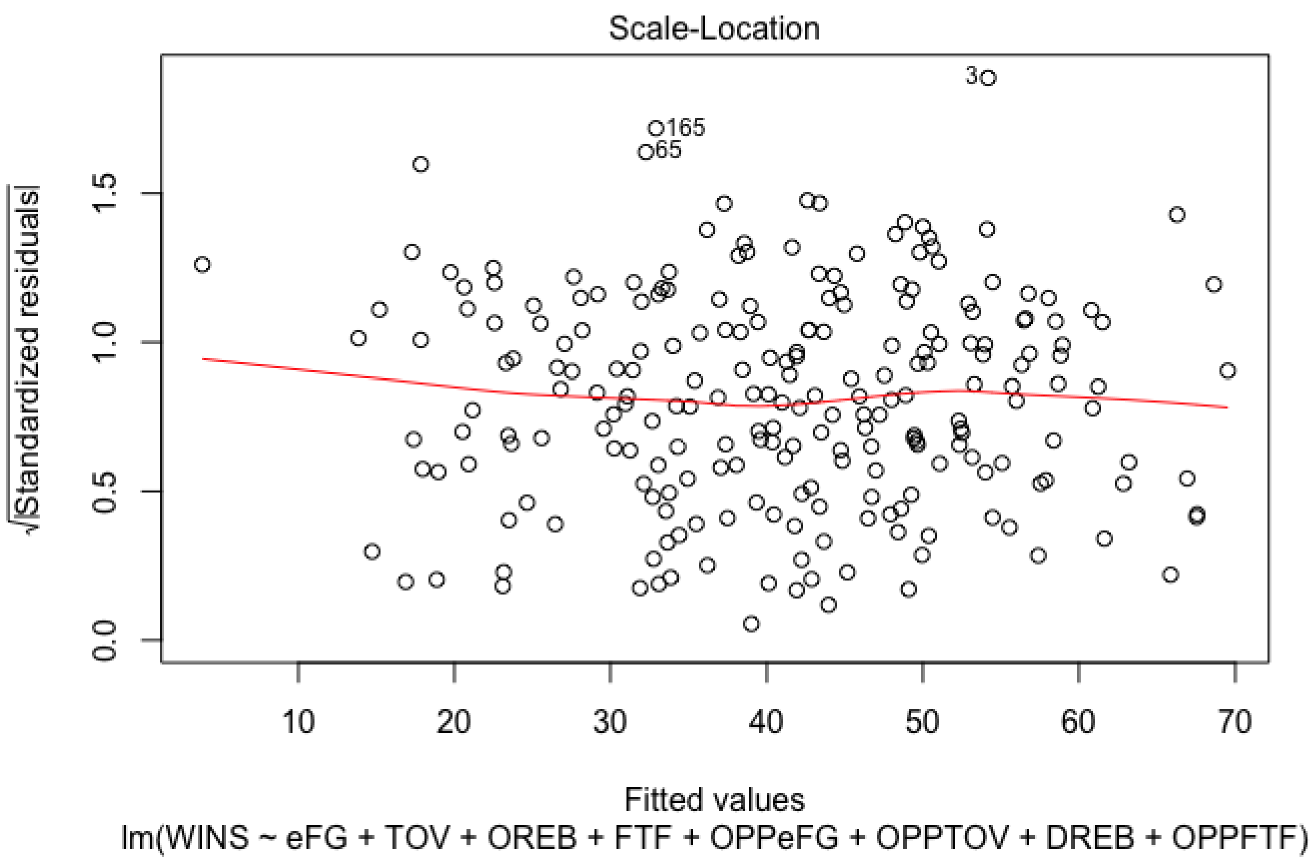
```
durbinWatsonTest(lmWINS)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.01605434       2.02369   0.836
##  Alternative hypothesis: rho != 0
```

We fail to reject the null hypothesis that the error terms are not autocorrelated. We have met the independence assumption.

3. Variation of observations’ error terms is constant

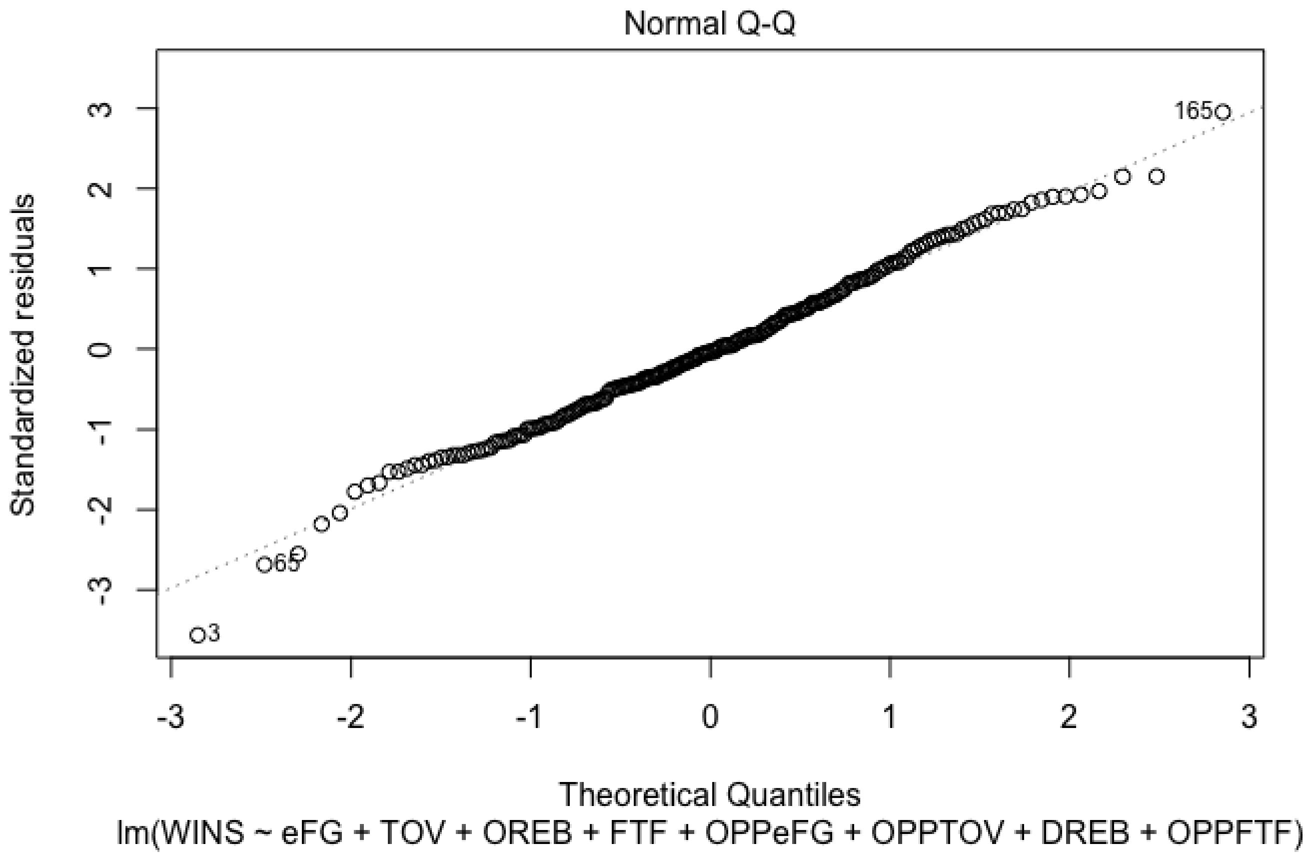
```
plot(lmWINS, 3)
```



The homoskedasticity assumption is met. In the scale-location plot we see points that spread in a normally pattern, there is no evidence of heteroskedasticity.

4. Values are normally distributed

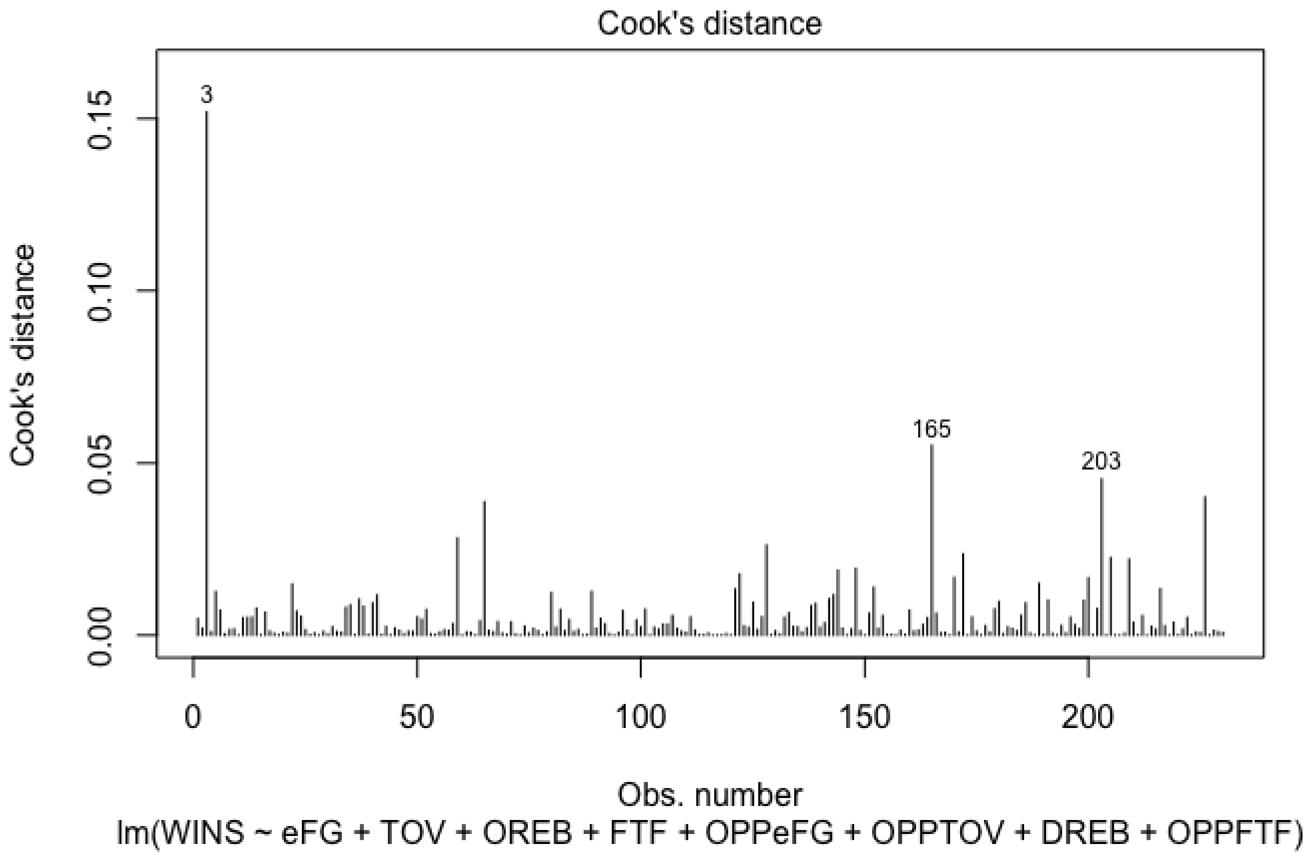
```
plot(lmWINS, 2)
```



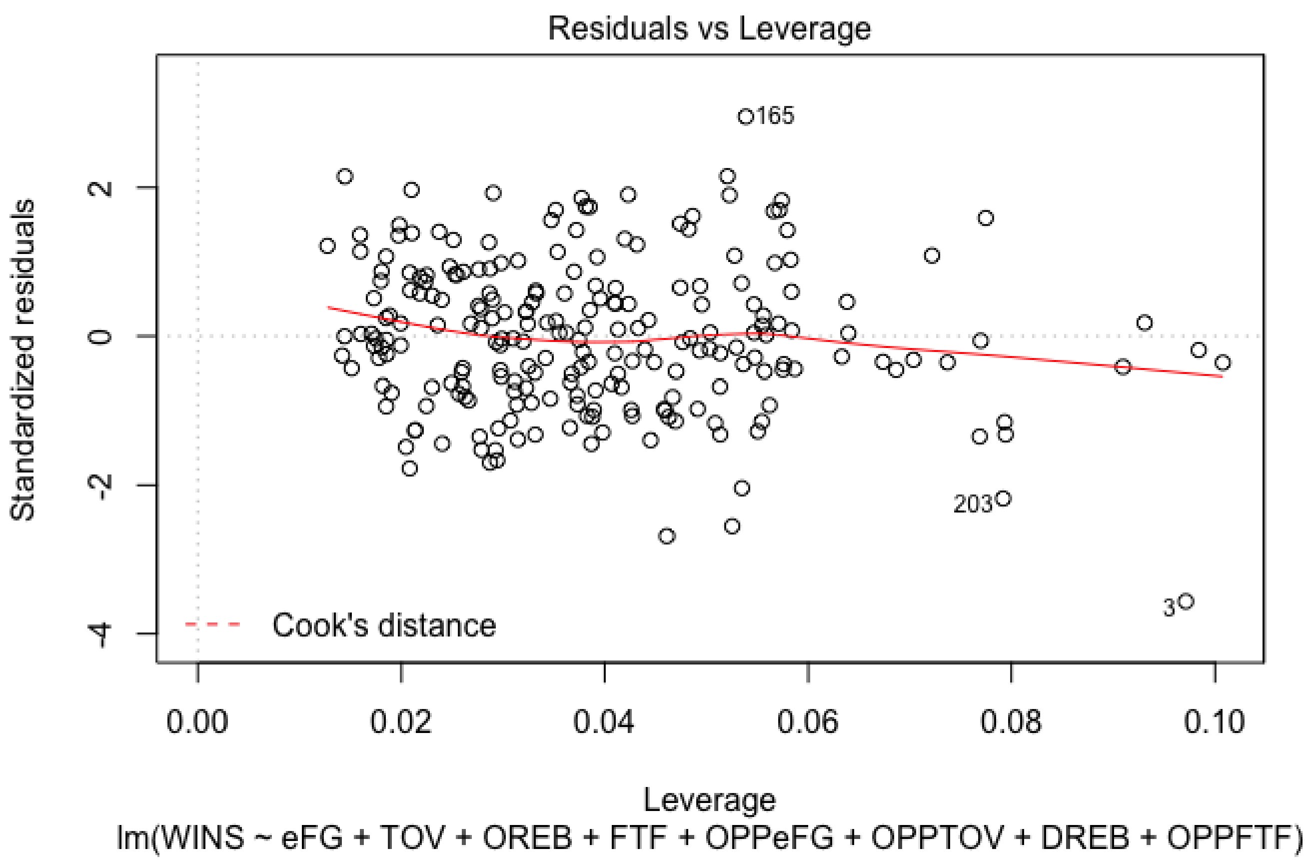
From the Q-Q plot, we can see that points are very close to the diagonal line and are normally distributed.

5. Outliers and Influential Points

```
plot(lmWINS, 4)
```




```
plot(lmWINS, 5)
```



We can look for outliers in our data using Cook's distance. Points 48, 138 and 188 are identified as outliers. These three points also appear in our residual vs. leverage graph as potential points of interest.

6. Multicolinearity

```
vif(lmWINS)
```

```
##      eFG      TOV     OREB      FTF   OPPeFG   OPPTOV     DREB   OPPFTF
## 1.339503 1.229970 1.470495 1.343472 1.120709 1.252527 1.632149 1.341740
```

All of our VIF values are below 4. There is no multicolinearity in our data.

3.3 Creating a Multiple Linear Regression model regressing the Four Factors on `MOV` using the train dataset

```
lmMOV = lm(MOV ~ eFG + TOV + OREB + FTF + OPPeFG + OPPTOV + DREB + OPPFTF, data = ff_tra
in)

summary(lmMOV)
```

```
##
## Call:
## lm(formula = MOV ~ eFG + TOV + OREB + FTF + OPPeFG + OPPTOV +
##     DREB + OPPFTF, data = ff_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.87201 -0.41263 -0.00396  0.46326  1.89270
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -38.85131    3.16528  -12.27   <2e-16 ***
## eFG           1.43555    0.02402   59.75   <2e-16 ***
## TOV          -1.32334    0.05413  -24.45   <2e-16 ***
## OREB          0.43792    0.01961   22.33   <2e-16 ***
## FTF           0.29640    0.02024   14.64   <2e-16 ***
## OPPeFG       -1.42002    0.02489  -57.06   <2e-16 ***
## OPPTOV        1.17049    0.04619   25.34   <2e-16 ***
## DREB          0.38560    0.02559   15.07   <2e-16 ***
## OPPFTF       -0.28811    0.02258  -12.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6872 on 221 degrees of freedom
## Multiple R-squared:  0.9806, Adjusted R-squared:  0.9799
## F-statistic:  1398 on 8 and 221 DF,  p-value: < 2.2e-16
```

3.3.1 Interpretation of F-statistic, P-value and R-squared value

From the F-statistic and p-value of our **lmMOV** model, we can reject the null hypothesis that our predictor variables have no effect on `MOV`. There is strong evidence to conclude there is a relationship between our predictor and response variables.

Our model explains approximately 97.9% of the variation in `WINS` using `eFG`, `TOV`, `OREB`, `FTF`, `OPPeFG`, `OPPTOV`, `DREB` and `OPPFTF` as predictors.

3.3.2 Interpretation of coefficients

- An increase of 1 percentage point of `eFG` is associated with an average increase of 1.42 `MOV`, holding all else equal
- An increase of 1 percentage point of `TOV` is associated with an average decrease of 1.35 `MOV`, holding all else equal

- An increase of 1 percentage point of `OREB` is associated with an average increase of 0.44 `MOV`, holding all else equal
- An increase of 1 percentage point of `FTF` is associated with an average increase of 0.28 `MOV`, holding all else equal
- An increase of 1 percentage point of `OPPeFG` is associated with an average decrease of 1.42 `MOV`, holding all else equal
- An increase of 1 percentage point of `OPPTOV` is associated with an average increase of 1.19 `MOV`, holding all else equal
- An increase of 1 percentage point of `DREB` is associated with an average increase of 0.42 `MOV`, holding all else equal
- An increase of 1 percentage point of `OPPFTF` is associated with an average decrease of 0.28 `MOV`, holding all else equal

3.3.3 ANOVA table and confidence interval

```
anova(lmMOV)
```

```
## Analysis of Variance Table
##
## Response: MOV
##            Df  Sum Sq Mean Sq  F value    Pr(>F)
## eFG         1 2301.63 2301.63 4874.378 < 2.2e-16 ***
## TOV         1   44.78   44.78   94.836 < 2.2e-16 ***
## OREB        1  709.96  709.96 1503.552 < 2.2e-16 ***
## FTF         1   84.40   84.40  178.739 < 2.2e-16 ***
## OPPeFG      1 1742.09 1742.09 3689.399 < 2.2e-16 ***
## OPPTOV      1  164.26  164.26  347.865 < 2.2e-16 ***
## DREB        1  157.77  157.77  334.123 < 2.2e-16 ***
## OPPFTF      1   76.89   76.89  162.835 < 2.2e-16 ***
## Residuals 221  104.35    0.47
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(lmMOV)
```

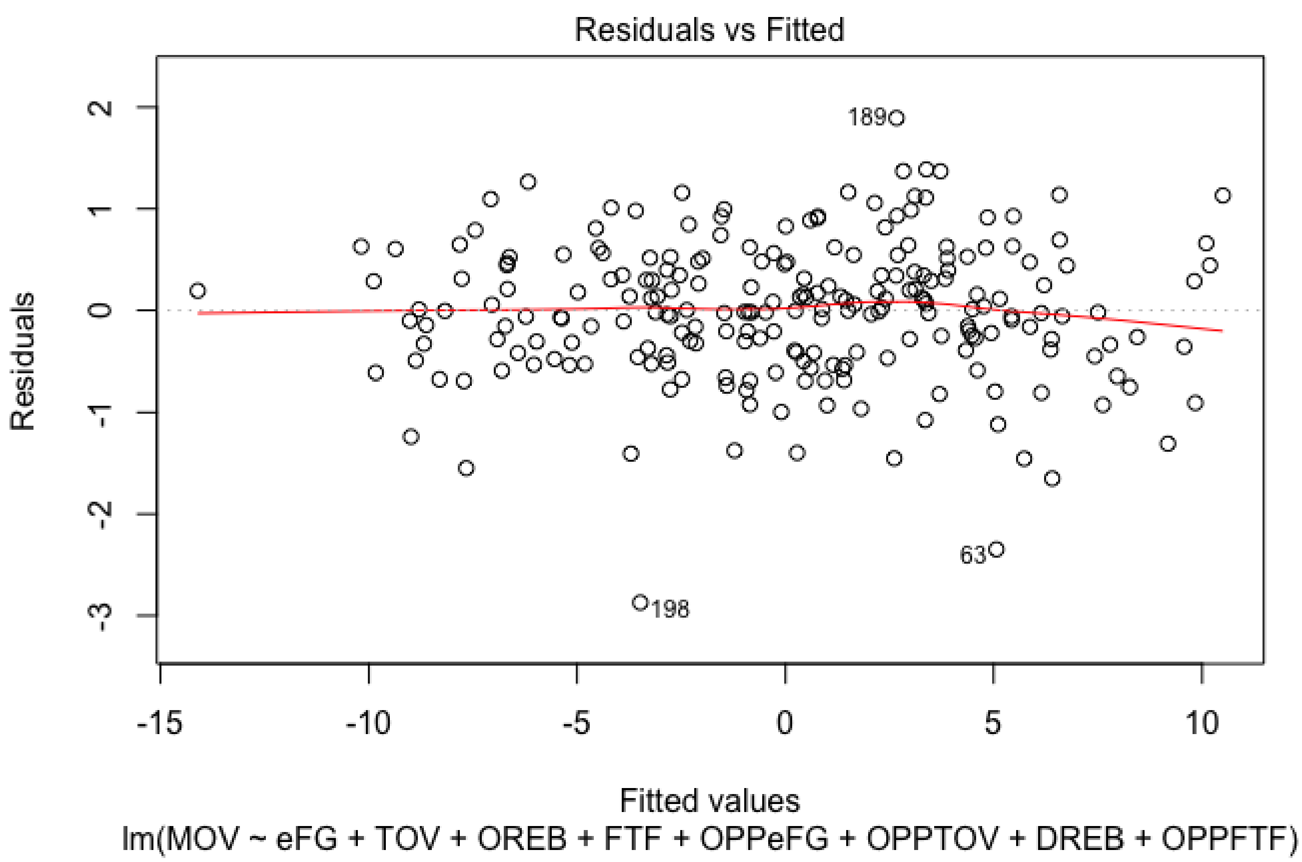
```
##                   2.5 %      97.5 %
## (Intercept) -45.0893055 -32.6133072
## eFG           1.3881997   1.4828921
## TOV          -1.4300169  -1.2166637
## OREB          0.3992758   0.4765695
## FTF           0.2565094   0.3362945
## OPPeFG       -1.4690670  -1.3709711
## OPPTOV        1.0794625   1.2615183
## DREB          0.3351639   0.4360461
## OPPFTF       -0.3326050  -0.2436138
```

From the ANOVA table, we can see that the F-statistic for each predictor variable is significant and adds prediction power to our model. All included variables are relevant to our model.

3.3.4 Checking to see if assumptions of linear regression for the **lmMOV** model are reasonably met

1. The relationship is linear

```
plot(lmMOV, 1)
```



***

Our residual plot is mostly flat. There are some points that skew the line, such as observation 116, 138 and 188, but overall the relationship is linear.

2. Independence of error terms

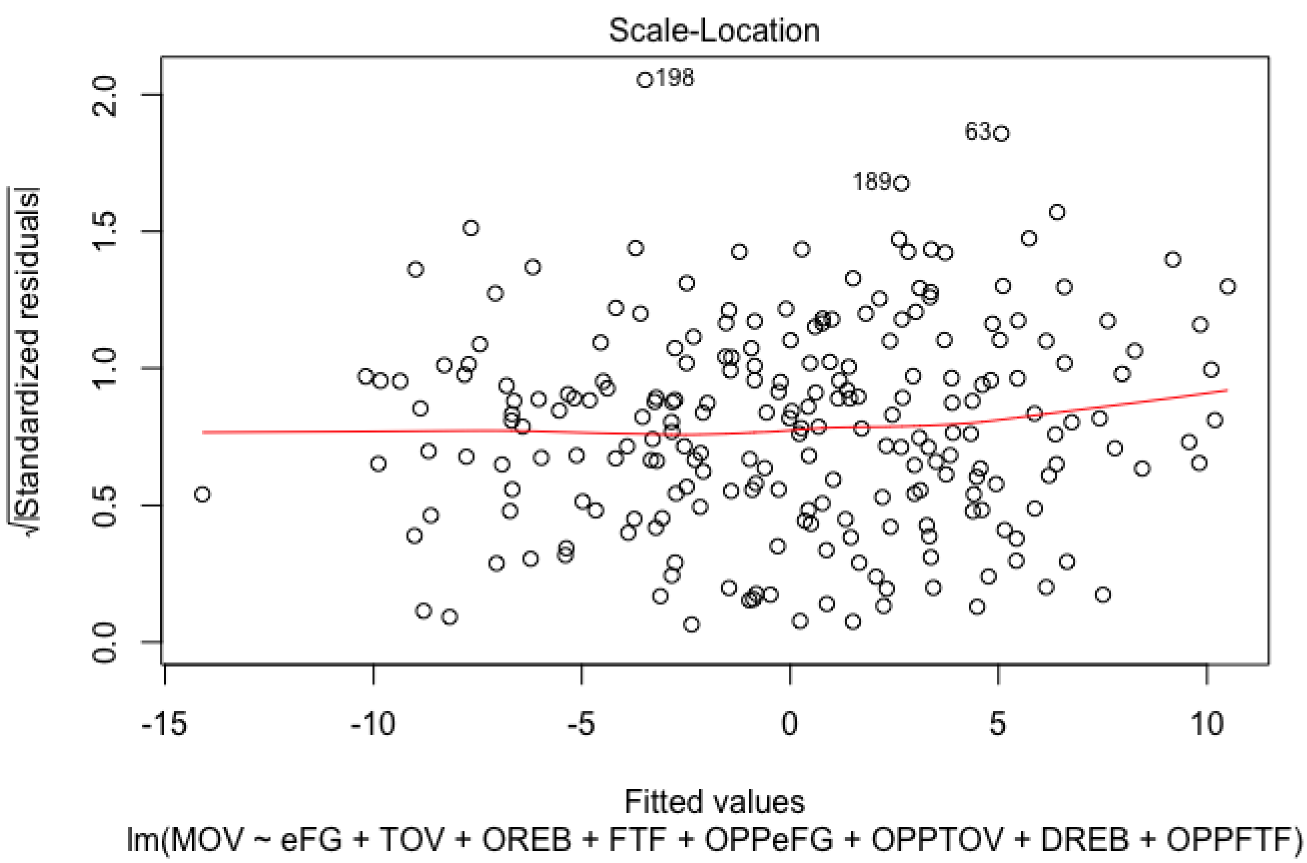
```
durbinWatsonTest(lmMOV)
```

```
##   lag Autocorrelation D-W Statistic p-value
##     1     -0.01259298      2.014365   0.886
##   Alternative hypothesis: rho != 0
```

We fail to reject the null hypothesis that the error terms are not autocorrelated. We have met the independence assumption.

3. Variation of observations' error terms is constant

```
plot(lmMOV, 3)
```

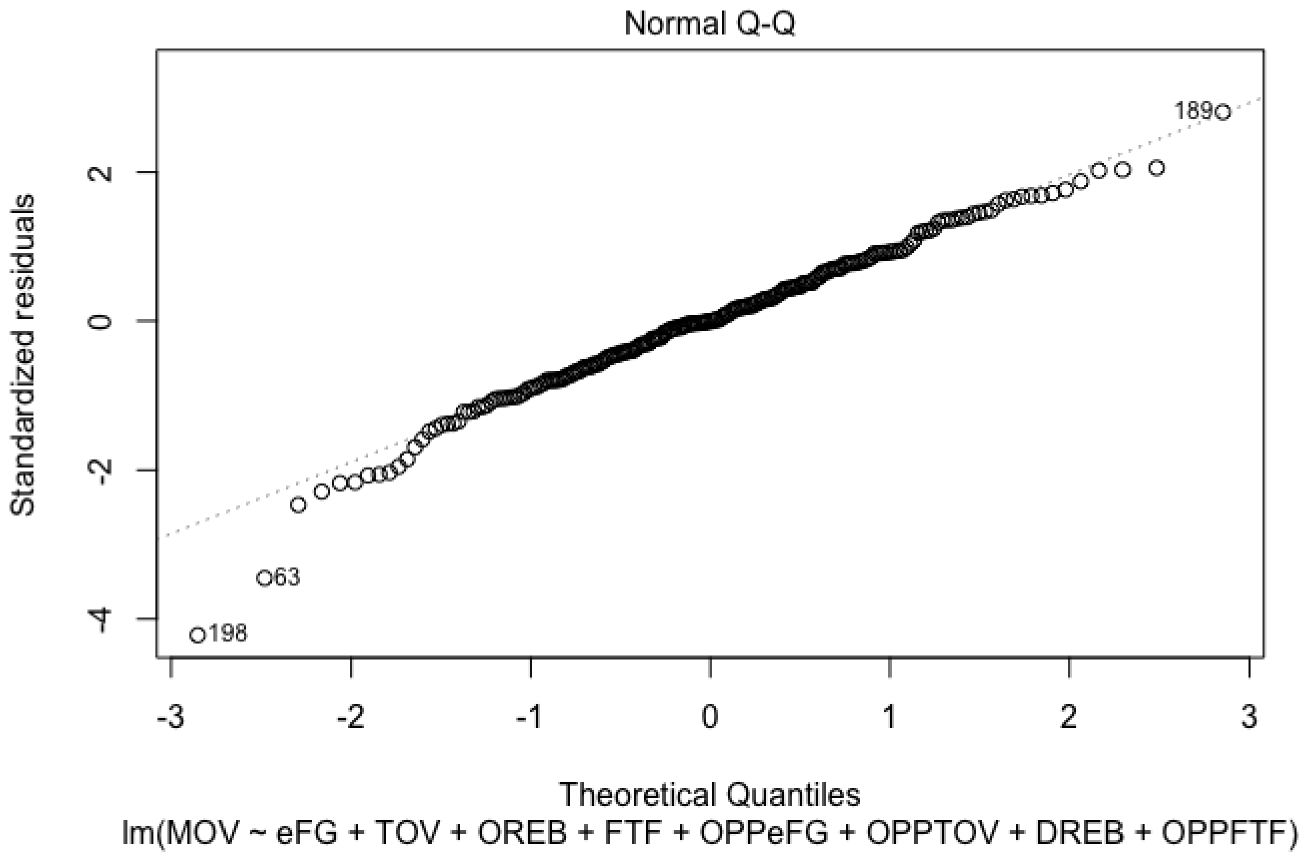


***

The homoskedasticity assumption is met. In the scale-location plot we see see points that are normally spread, there is no evidence of heteroskedasticity.

4. Values are normally distributed

```
plot(lmMOV, 2)
```

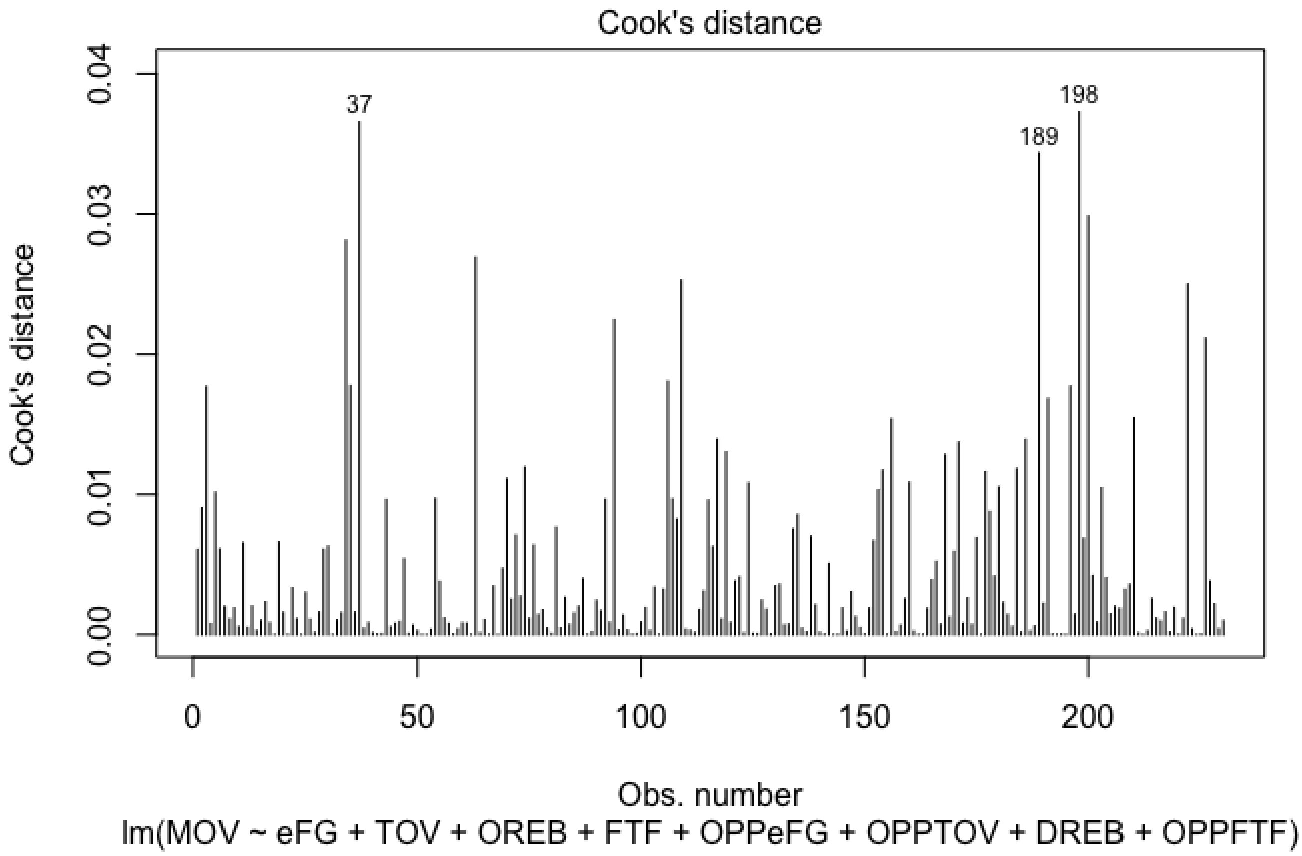


***

From the Q-Q plot, we can see that points are very close to the diagonal line.

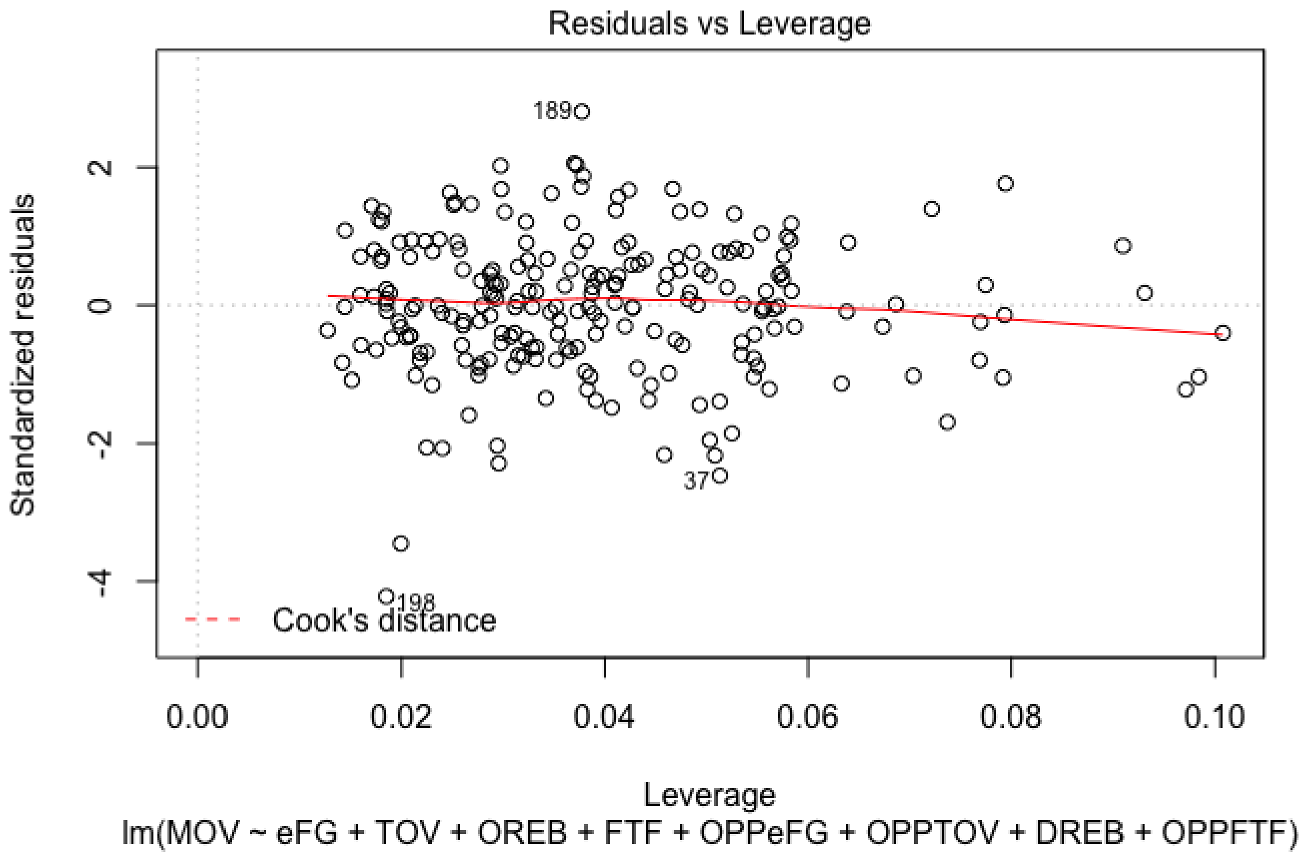
5. Outliers and Influential Points

```
plot(lmMOV, 4)
```



Cook's distance
Cook's distance
0.00
0.01
0.02
0.03
0.04
37
189
198
0
50
100
150
200
Obs. number
lm(MOV ~ eFG + TOV + OREB + FTF + OPPeFG + OPPTOV + DREB + OPPFTF)


```
plot(lmMOV, 5)
```



***

We can look for outliers in our data using Cook's distance. Points 48, 138 and 188 are identified as outliers. These three points also appear in our residual vs leverage graph as potential points of interest.

6. Multicolinearity

```
vif(lmMOV)
```

```
##      eFG      TOV     OREB      FTF   OPPeFG   OPPTOV     DREB   OPPFTF
## 1.339503 1.229970 1.470495 1.343472 1.120709 1.252527 1.632149 1.341740
```

All of our VIF values are below 4. There is no multicolinearity in our data.

3.3.5 Linear Regression Models Summary

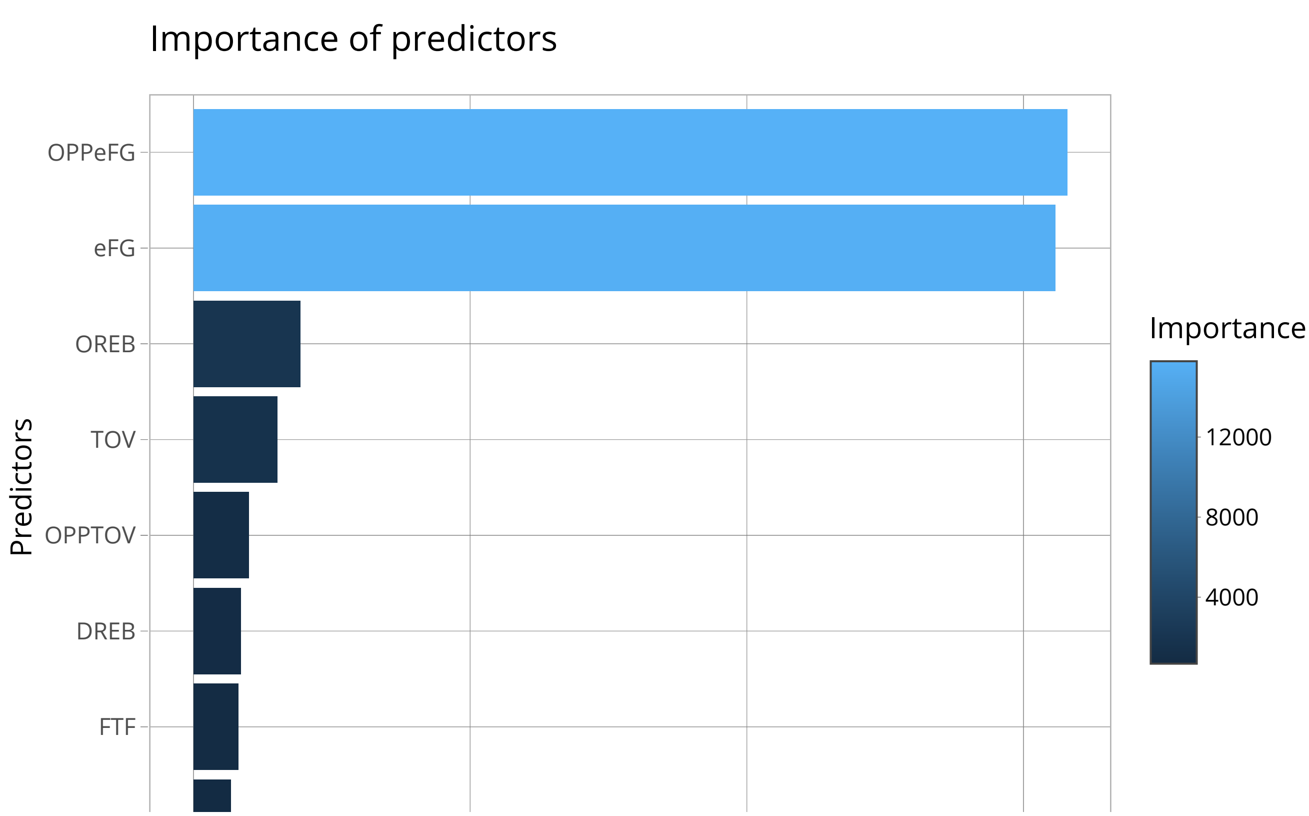
From the ANOVA tables in 3.2.3 and 3.3.3, we can see that all four factors that Dean proposed are significant at threshold of p-value <0.01 for predicted both `WINS` and `MOV`

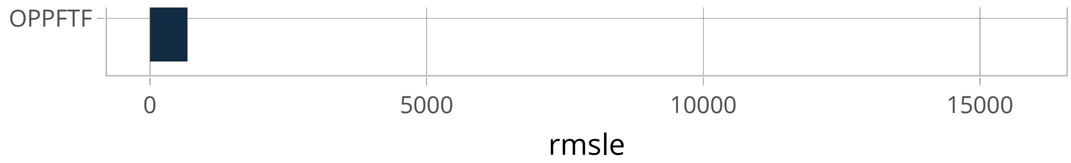
3.4 Random Forest Model for `WINS`

```
set.seed(10)
rfWINS <- randomForest(formula = WINS ~ eFG + TOV + OREB +
  FTF + OPPeFG + OPPTOV + DREB +
  OPPFTF, type = prob, mtry = 8, ntree = 100,  data = ff_train)

rfWINS
```

```
##
## Call:
##  randomForest(formula = WINS ~ eFG + TOV + OREB + FTF + OPPeFG +      OPPTOV + DREB +
OPPFTF, data = ff_train, type = prob, mtry = 8,      ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 33.0514
##                     % Var explained: 80.51
```

```
importance_rfWINS <- importance(rfWINS)

varImportance_rfWINS <- data.frame(Variables = row.names(importance_rfWINS),
                            Importance = round(importance_rfWINS[,'IncNodePurity'], 0))

ggplotly(ggplot(varImportance_rfWINS, aes(x = reorder(Variables, Importance),
                           y = Importance, fill = Importance)) +
       geom_bar(stat='identity') +
       labs(title = 'Importance of predictors', x = 'Predictors', y = 'rmsle') +
       coord_flip() +
       theme_light())
```

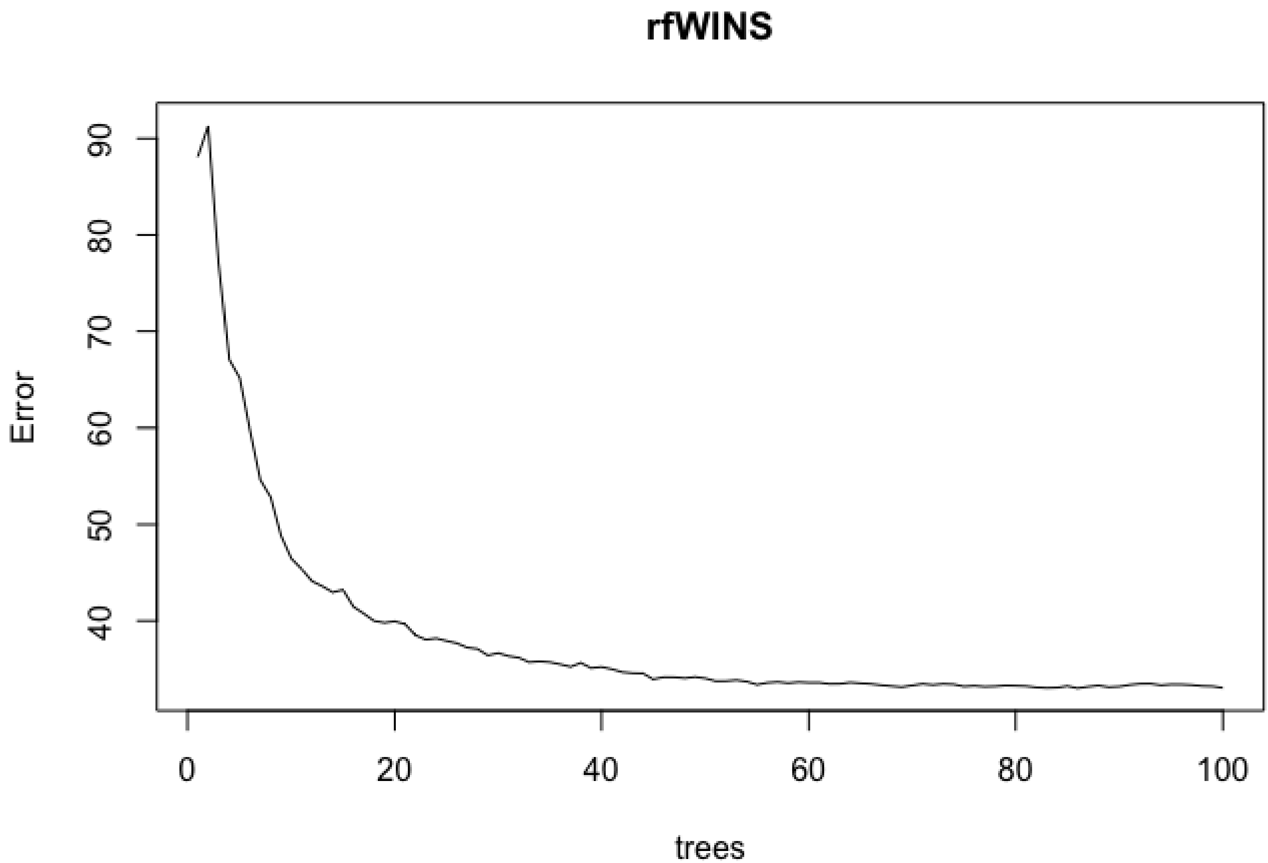





```
predicted_rfWINS = predict(rfWINS, ff_test)
predicted_rfWINS
```

```
##        1        2        3        4        5        6        7        8
## 52.99617 43.78906 49.08317 38.65268 37.72000 45.11800 28.95717 29.78150
##        9       10       11       12       13       14       15       16
## 51.47067 53.47717 51.63918 43.44083 42.54587 42.30806 45.12500 28.14433
##       17       18       19       20       21       22       23       24
## 47.06783 37.83912 47.22656 44.46980 33.08417 33.71457 29.72049 28.43883
##       25       26       27       28       29       30       31       32
## 30.16586 52.91377 49.44632 48.92538 40.72462 41.91182 46.92298 33.05839
##       33       34       35       36       37       38       39       40
## 33.57583 26.20969 54.45130 44.35616 45.35495 43.04947 46.37922 43.84762
##       41       42       43       44       45       46       47       48
## 40.67612 47.07255 39.06687 31.40869 33.75418 18.75129 43.79155 45.90477
##       49       50       51       52       53       54       55       56
## 41.04936 34.39777 29.19262 26.48208 52.36279 55.21718 40.48489 45.21512
##       57       58       59       60       61       62       63       64
## 46.49743 46.92894 36.20323 26.10243 24.61030 21.00752 29.63551 58.56314
##       65       66       67       68       69       70       71       72
## 47.80291 52.17222 47.25449 45.79075 42.71789 40.44831 28.14403 42.04495
##       73       74       75       76       77       78       79       80
## 40.24382 26.79997 35.26717 57.66292 34.47036 37.69035 39.97372 23.62953
##       81       82       83       84       85       86       87       88
## 59.27665 47.52023 47.67037 48.13347 33.84547 37.82014 32.72050 40.61161
##       89       90       91       92       93       94       95       96
## 30.91467 31.51412 28.12411 53.57304 49.35518 52.68319 48.66120 51.65824
##       97       98       99      100
## 42.09069 42.29389 40.42742 38.33552
```

```
plot(rfWINS)
```

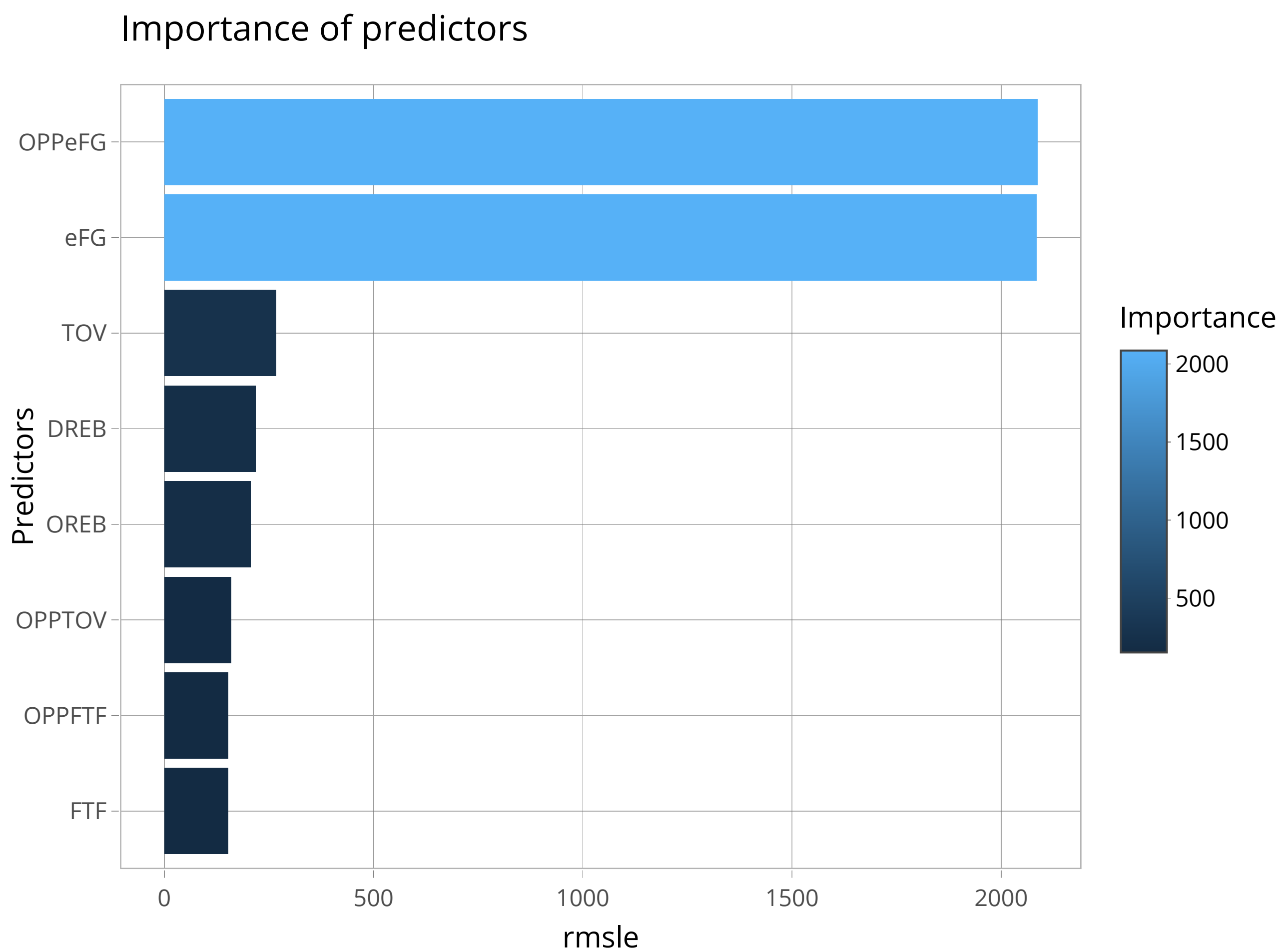


3.5 Random Forest Model for `MOV`

```
set.seed(12)
rfMOV <- randomForest(
  formula = MOV ~ eFG + TOV + OREB +
  FTF + OPPeFG + OPPTOV + DREB +
  OPPFTF, type = prob, mtry = 5, ntree = 100,  data = ff_train)

rfMOV
```

```
##
## Call:
##  randomForest(formula = MOV ~ eFG + TOV + OREB + FTF + OPPeFG +       OPPTOV + DREB +
OPPFTF, data = ff_train, type = prob, mtry = 5,      ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 4.322187
##                     % Var explained: 81.54
```

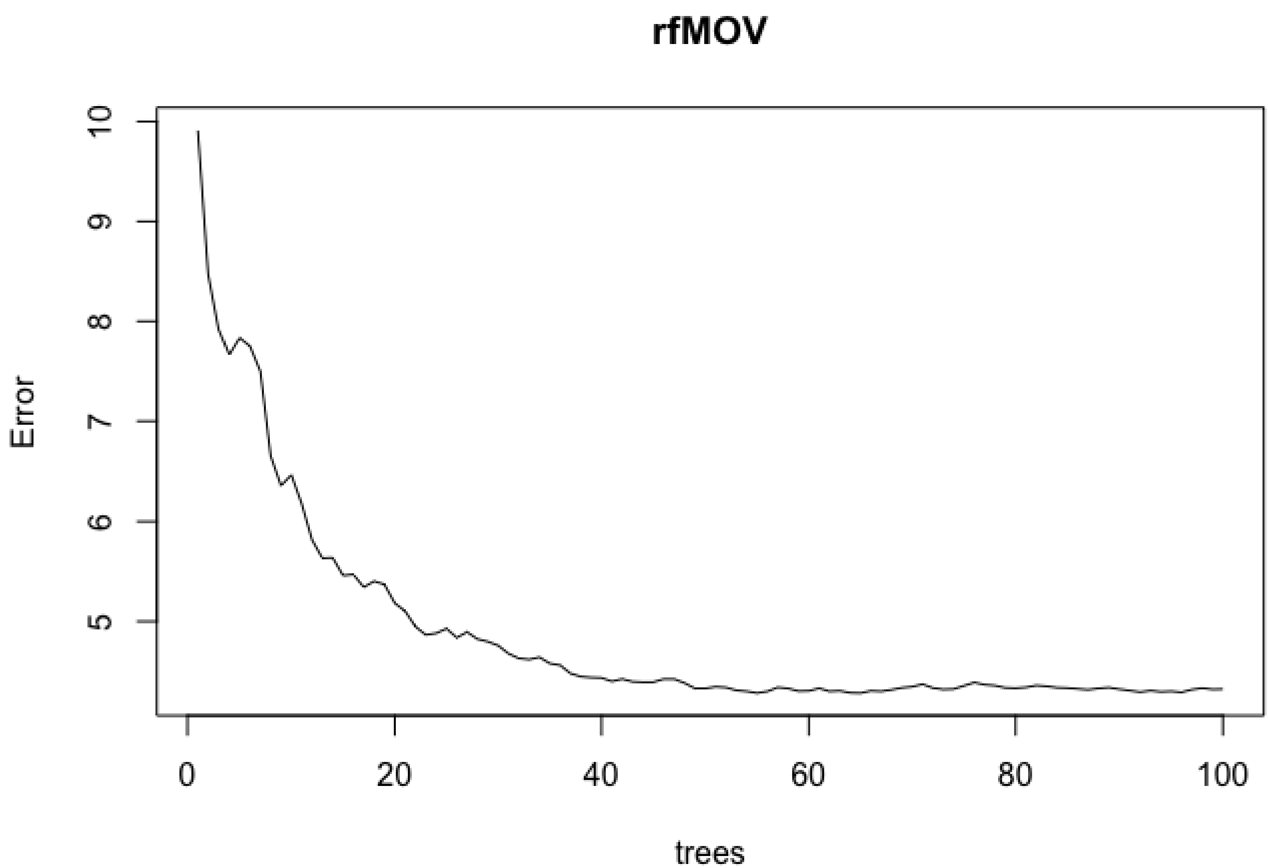
```
importance_rfMOV <- importance(rfMOV)

varImportance_rfMOV <- data.frame(Variables = row.names(importance_rfMOV),
                                  Importance = round(importance_rfMOV[,'IncNodePurity'], 0))

ggplotly(ggplot(varImportance_rfMOV, aes(x = reorder(Variables, Importance),
                                 y = Importance, fill = Importance)) +
       geom_bar(stat='identity') +
       labs(title = 'Importance of predictors', x = 'Predictors', y = 'rmsle') +
       coord_flip() +
       theme_light())
```




```
predicted_rfMOV = predict(rfMOV, ff_test)
predicted_rfMOV
```

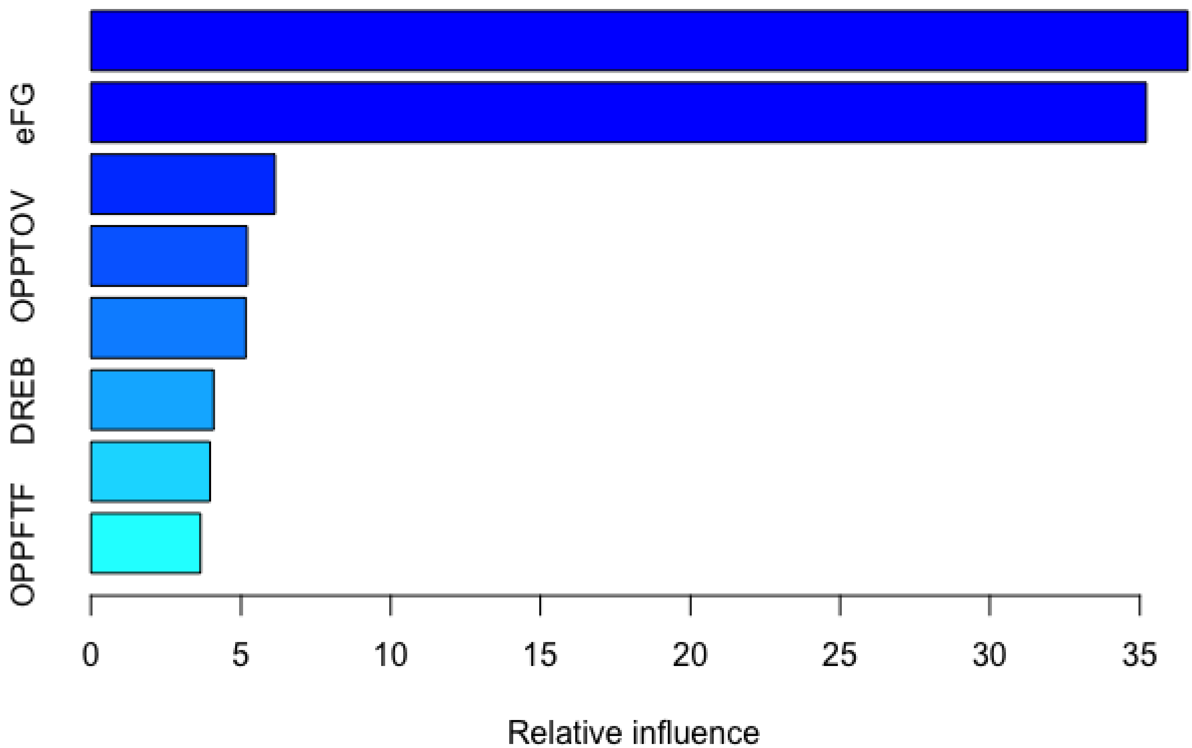
```
##          1          2          3          4          5          6 
##  4.70411333  0.44492000  1.75897667 -0.39023667 -0.65458833  1.04610167 
##          7          8          9         10         11         12 
## -5.13784333 -4.64947000  3.01612000  5.37060667  4.40723667  0.43265667 
##         13         14         15         16         17         18 
##  0.47341500  0.22988500  1.07280833 -4.98806500  3.65733833 -1.29342000 
##         19         20         21         22         23         24 
##  1.10023500  3.13646333 -2.43330500 -2.40070500 -4.84224167 -4.69837167 
##         25         26         27         28         29         30 
## -3.50574167  4.81272833  1.17967500  1.60443667 -0.22607167 -0.08688167 
##         31         32         33         34         35         36 
##  0.83038833 -3.54648833 -2.97943667 -5.30107000  4.92724000  1.06191667 
##         37         38         39         40         41         42 
##  1.81877000  0.84503667  1.74553667  1.06317167  0.26851667  1.85895500 
##         43         44         45         46         47         48 
## -1.31293167 -3.61436833 -3.26125667 -8.15337167  0.39171667  1.58442000 
##         49         50         51         52         53         54 
##  1.53426167 -1.87378167 -4.02905333 -5.35198167  2.85475333  5.20183333 
##         55         56         57         58         59         60 
## -0.58595167  1.51368000  2.33989333  1.82947500 -1.72388833 -4.91215667 
##         61         62         63         64         65         66 
## -5.93165167 -6.72481333 -4.62892333  5.73980333  2.55964000  4.38946333 
##         67         68         69         70         71         72 
##  1.96638333  2.08326333  0.36124667 -0.63832833 -4.68654167 -0.27848833 
##         73         74         75         76         77         78 
##  1.12000667 -5.47748000 -0.94587833  6.69601333 -2.18986333 -1.58901667 
##         79         80         81         82         83         84 
## -0.38305167 -5.47842833  7.41262333  1.97604167  2.18967167  1.75231333 
##         85         86         87         88         89         90 
## -2.43702333  0.16829167 -3.36714667  0.41332833 -3.02051667 -4.18188333 
##         91         92         93         94         95         96 
## -4.99639167  3.64797500  2.23762167  2.39703500  1.67150667  1.58936167 
##         97         98         99        100 
##  0.67753167 -0.05850833 -0.20965833 -0.78329167
```

```
plot(rfMOV)
```



3.6 Gradient Boosting Machine Model for `WINS`

```
set.seed (14)
gbWINS = gbm(formula = WINS ~ eFG + TOV + OREB +
  FTF + OPPeFG + OPPTOV + DREB +
  OPPFTF, data = ff_train, distribution =
"gaussian", n.trees =  5000, interaction.depth = 5)

summary(gbWINS)
```



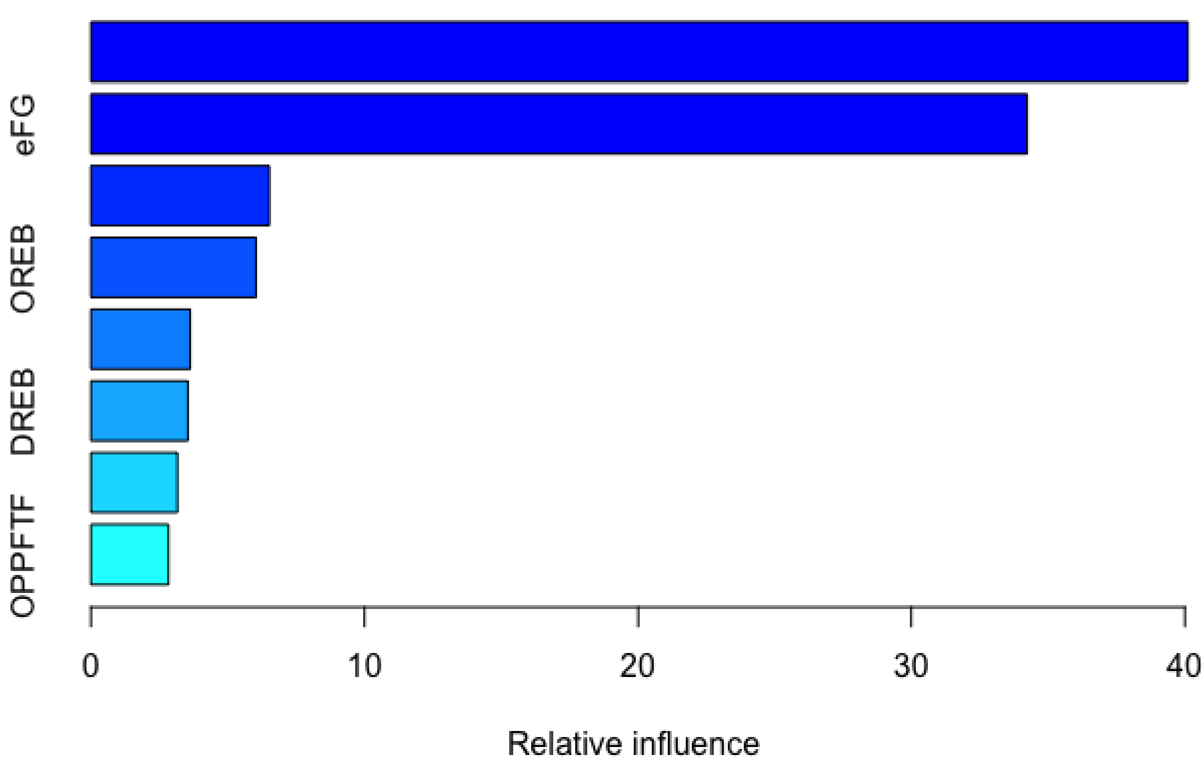

```
##           var   rel.inf
## OPPeFG OPPeFG 36.601679
## eFG       eFG 35.210560
## TOV       TOV  6.133314
## OPPTOV OPPTOV  5.201989
## OREB     OREB  5.157850
## DREB     DREB  4.084093
## FTF       FTF  3.963953
## OPPFTF OPPFTF  3.646562
```

```
predicted_gbWINS = predict(gbWINS, ff_test, n.trees = 5000)
predicted_gbWINS
```

```
##   [1] 54.44049 45.89374 54.60644 40.68630 41.00000 43.93951 28.02161 23.42374
##   [9] 55.74394 62.25554 52.41642 38.39444 44.04356 47.60842 45.25060 22.51185
##  [17] 47.30781 40.48799 43.94208 41.38360 33.91815 37.47008 25.40001 29.22159
##  [25] 26.60773 57.48890 47.36590 49.95835 39.05216 45.59009 51.32060 33.49707
##  [33] 32.30885 22.26007 54.97178 51.20055 48.93977 47.41121 50.82429 41.43935
##  [41] 42.49904 48.98574 36.50340 27.24876 28.30373 11.34907 49.43297 49.14642
##  [49] 42.16120 36.78434 30.01739 22.34322 53.87502 57.81247 42.34259 45.60753
##  [57] 46.49116 45.47086 40.02824 25.56844 27.17362 17.77033 25.17434 62.65383
##  [65] 45.06780 52.47862 46.44692 49.74115 44.18167 40.26226 26.53433 34.56317
##  [73] 33.89384 26.78798 30.22813 60.06000 36.13741 36.93802 35.28440 20.91014
##  [81] 62.66122 54.20535 45.18643 51.02636 38.68406 37.89987 35.22453 30.79658
##  [89] 31.06456 22.50787 25.16126 56.68220 48.24911 48.58269 46.49725 45.94790
##  [97] 40.09052 44.84519 38.48797 41.59021
```

3.7 Gradient Boosting Machine Model for `MOV`

```
set.seed (16)
gbMOV = gbm(formula = MOV ~ eFG + TOV + OREB +
  FTF + OPPeFG + OPPTOV + DREB +
  OPPFTF, data = ff_train, distribution =
"gaussian", n.trees =  5000, interaction.depth = 5)

summary(gbMOV)
```



```
##           var   rel.inf
## OPPeFG OPPeFG 40.107700
## eFG       eFG 34.240521
## TOV       TOV  6.511891
## OREB     OREB  6.021583
## OPPTOV OPPTOV  3.617506
## DREB     DREB  3.531286
## FTF       FTF  3.147438
## OPPFTF OPPFTF  2.822075
```

```
predicted_gbMOV = predict(gbMOV, ff_test, n.trees = 5000)
predicted_gbMOV
```

```
##   [1]   5.41401242   2.98954023   4.04905088  -1.66334388  -0.24000078
##   [6]   0.11639329  -6.03584089  -5.04345189   5.91855454   8.05355326
##  [11]   3.76276710   0.42580245   0.96263642   2.46004525   1.20142592
##  [16]  -6.85072444   2.68593527  -1.53143981   1.54774574  -0.03515514
##  [21]  -2.44130746  -0.78319931  -5.38327137  -4.27373614  -5.27720429
##  [26]   5.02091121   1.60751046   3.27731656  -0.81257916   0.71445214
##  [31]   0.96450630  -4.18510032  -2.38329799  -7.36643211   6.04378400
##  [36]   2.18513304   3.11570476   2.78447763   1.89781985   0.10434255
##  [41]  -0.48908900   0.81382123   1.53105953  -5.38207616  -3.96597397
##  [46] -11.41483236   2.35197568   3.47247108   3.05802638  -1.43422693
##  [51]  -2.52666049  -4.78003448   4.82989906   4.76372686   0.39456892
##  [56]   2.16179877   1.36581876   1.60101384  -0.54169606  -5.30367695
##  [61]  -4.02114242  -7.67902734  -5.69591678   6.07631960   3.84908966
##  [66]   5.33339226   1.69472122   2.80727646  -0.80714746  -1.03379846
##  [71]  -4.36500136  -1.92826525  -0.77525729  -5.26684295  -2.17438589
##  [76]   7.09703563  -1.08193043  -0.31680183  -1.56470605  -5.55053625
##  [81]   8.11648806   3.64261673   1.57617387   2.40660078  -0.91387736
##  [86]  -0.68600691  -2.50503055  -3.61996388  -3.00463098  -5.68999341
##  [91]  -7.78111208   4.42632578   1.42397457   3.16395713   1.71156017
##  [96]   2.59497571  -0.04093237  -0.15333345  -1.25801459  -0.44904712
```

3.8 Comparison of models

Both `WINS` and `MOV` were predicted using three different models:

1. Multiple Linear Regression
2. Random Forest
3. Gradient Boosting Machine

For the final model, I decided to use multiple linear regression. The **lmWINS** and **lmMOV** models peformed better than the **rfWINS**, **rfMOV**, **gbWINS**, and **gbMOV** models. Random forest and gradient boosting may have performed better, had I tweaked and tested different hyperparameter configurations.

Another reason I picked multiple linear regression as the final model type, is that it is a simpler model, which makes it easier to explain. In this case, the other models are black-box models, which can potentially provide more accuracy, but at the expense of explainability compared to a model like multiple linear regression.

Our **lmWINS** and **lmMOV** models can be stated mathematically as:

$$
\begin{aligned}
WINS = & -53.185 + 3.899 * eFG\% - 3.392 * TOV\% + 1.111 * OREB\% \\
& + 0.708 * FTF - 3.765 * OPPeFG\% + 2.902 * OPPTOV\% \\
& + 0.887 * DREB\% - 0.729 * OPPFTF + \epsilon
\end{aligned}
$$

$$
\begin{aligned}
MOV = & -38.851 + 1.436 * eFG\% - 1.323 * TOV\% + 0.438 * OREB\% \\
& + 0.296 * FTF - 1.420 * OPPeFG\% + 1.170 * OPPTOV\% \\
& + 0.386 * DREB\% - 0.288 * OPPFTF + \epsilon
\end{aligned}
$$

In the next section, we will evaluate our multiple linear regression model

---

4. Comparison of actual values with predicted values and metrics

4.1 Creating a dataframe comparing predicted `WINS` and predicted `MOV` using our **lmWINS** and **lmMOV** models fit the test data, with the observed values of `WINS` and observed values of `MOV`

```
predicted_wins = predict(lmWINS, ff_test)
pm = (predict(lmMOV, ff_test))

actual_vs_predicted <- data.frame(predicted_wins)
actual_vs_predicted$actual_wins <- ff_test[["WINS"]]
actual_vs_predicted$predicted_mov <- pm
actual_vs_predicted$actual_mov <- ff_test[["MOV"]]

colnames(actual_vs_predicted) <- c("Predicted Wins", "Actual Wins", "Predicted Margin of
Victory", "Acutal Margin of Victory")

actual_vs_predicted <- actual_vs_predicted[, c(2, 1, 4, 3)]
actual_vs_predicted
```

```
##      Actual Wins Predicted Wins Acutal Margin of Victory
## 1       60.00000       61.01426                     8.87
## 2       54.00000       48.44225                     3.95
## 3       49.00000       51.48235                     4.44
## 4       42.00000       41.65049                     0.71
## 5       39.00000       38.43315                    -0.23
## 6       32.00000       35.05902                    -2.90
## 7       19.00000       17.24568                    -9.34
## 8       19.00000       18.57312                    -9.61
## 9       65.00000       59.85960                     8.48
## 10      59.00000       58.05547                     7.78
## 11      58.00000       58.90024                     5.98
## 12      49.00000       44.46868                     2.60
## 13      48.00000       44.94865                     1.38
## 14      47.00000       47.55992                     2.89
## 15      44.00000       42.03005                    -0.30
## 16      27.00000       22.34545                    -7.04
## 17      53.00000       46.97189                     2.63
## 18      43.00000       39.29831                     0.49
## 19      41.00000       45.34191                     1.06
## 20      41.00000       40.81297                    -0.52
## 21      36.00000       40.88568                     0.20
## 22      31.00000       37.46738                    -1.11
## 23      28.00000       26.72349                    -5.70
## 24      24.00000       27.68881                    -5.63
## 25      20.00000       25.10779                    -6.73
## 26      57.00000       54.16213                     6.00
## 27      48.00000       47.17588                     2.72
## 28      48.00000       48.30282                     1.65
## 29      42.00000       40.20250                    -0.30
## 30      41.00000       42.65048                     0.20
## 31      40.00000       44.25392                     1.79
## 32      33.00000       33.44204                    -4.18
## 33      30.00000       30.47749                    -3.79
## 34      21.00000       23.73058                    -7.35
## 35      55.00000       55.19093                     6.20
## 36      55.00000       47.61074                     3.24
## 37      50.00000       49.85750                     2.90
## 38      49.00000       47.26998                     3.07
## 39      45.00000       44.86464                     2.18
## 40      41.00000       42.50803                     0.43
## 41      40.00000       40.09710                     0.16
## 42      38.00000       42.82806                     0.22
## 43      32.00000       38.64799                    -1.00
## 44      30.00000       31.29008                    -3.55
## 45      29.00000       32.11757                    -3.71
## 46      17.00000       16.43611                    -9.32
## 47      50.00000       45.01407                     1.57
## 48      48.00000       47.80338                     2.63
## 49      48.00000       44.02758                     1.85
## 50      37.00000       37.72357                    -0.79
## 51      29.00000       33.43477                    -3.66
## 52      25.00000       26.74820                    -4.46
```

```
## 53     57.00000       57.15153                                  5.09
## 54     56.00000       59.06653                                  6.45
## 55     49.00000       44.64585                                  1.78
## 56     45.00000       39.80145                                  0.32
## 57     44.00000       43.75584                                  0.40
## 58     41.00000       40.88056                                 -0.22
## 59     38.00000       37.77610                                 -1.50
## 60     28.00000       28.31915                                 -4.88
## 61     24.00000       25.38363                                 -4.68
## 62     21.00000       23.23027                                 -9.23
## 63     20.00000       23.14253                                 -6.99
## 64     62.12121       59.43994                                  7.17
## 65     52.18182       47.04342                                  3.30
## 66     50.93939       45.07695                                  1.42
## 67     49.69697       49.07492                                  2.56
## 68     44.72727       43.00650                                  0.95
## 69     41.00000       40.82477                                 -0.24
## 70     38.51515       40.42987                                  0.27
## 71     31.06061       25.54173                                 -4.79
## 72     28.57576       33.16422                                 -3.41
## 73     28.57576       31.75337                                 -3.30
## 74     27.33333       27.48641                                 -5.68
## 75     26.09091       31.56932                                 -3.76
## 76     52.00000       58.51056                                  5.46
## 77     48.00000       41.88999                                  1.52
## 78     42.00000       42.32496                                  0.78
## 79     37.00000       36.12199                                 -1.07
## 80     17.00000       23.72609                                 -6.63
## 81     61.00000       63.93179                                  6.52
## 82     53.00000       52.72042                                  4.66
## 83     53.00000       51.31638                                  4.09
## 84     50.00000       48.82607                                  3.30
## 85     42.00000       37.32024                                 -0.37
## 86     40.00000       36.95088                                 -1.51
## 87     37.00000       34.73534                                 -2.46
## 88     32.00000       31.99379                                 -3.01
## 89     27.00000       30.85953                                 -3.90
## 90     26.00000       28.09346                                 -3.60
## 91     25.00000       26.54171                                 -4.37
## 92     65.00000       59.99583                                  7.66
## 93     54.00000       51.07413                                  3.76
## 94     54.00000       52.38515                                  3.41
## 95     47.00000       47.16779                                  1.57
## 96     46.00000       50.29970                                  1.93
## 97     41.00000       39.88191                                 -0.28
## 98     36.00000       36.45967                                 -1.11
## 99     35.00000       37.67511                                 -1.27
## 100    34.00000       37.06367                                 -1.09
##     Predicted Margin of Victory
## 1                     7.5019316
## 2                     2.7918936
## 3                     3.8579222
## 4                     0.1291505
## 5                    -0.8615088
```

```
## 6                            -2.4180218
## 7                            -9.0182489
## 8                            -8.7940497
## 9                             7.2595134
## 10                            6.3801864
## 11                            6.3904133
## 12                            1.3311391
## 13                            1.4623198
## 14                            2.6330804
## 15                            0.3325235
## 16                           -6.9689690
## 17                            2.1073394
## 18                           -0.4522467
## 19                            1.5203129
## 20                           -0.2177054
## 21                            0.2178854
## 22                           -1.2279486
## 23                           -5.5494226
## 24                           -4.9422247
## 25                           -6.2136536
## 26                            4.9859142
## 27                            2.5351641
## 28                            2.7661311
## 29                           -0.2998492
## 30                            0.6139207
## 31                            1.3555547
## 32                           -2.9487069
## 33                           -3.8951258
## 34                           -6.5288869
## 35                            5.3778730
## 36                            2.7033167
## 37                            3.2481478
## 38                            2.3854624
## 39                            1.5309146
## 40                            0.5272282
## 41                           -0.3470279
## 42                            0.7780669
## 43                           -0.7966378
## 44                           -3.6334464
## 45                           -3.2572235
## 46                           -9.4193199
## 47                            1.6106493
## 48                            2.7168844
## 49                            1.3638212
## 50                           -1.3144823
## 51                           -2.7455177
## 52                           -5.4021469
## 53                            6.0919311
## 54                            6.7979839
## 55                            1.4475761
## 56                           -0.3844359
## 57                            0.8080249
## 58                           -0.2358748
## 59                           -1.2575482
```

```
## 60                    -4.9433063
## 61                    -5.8192197
## 62                    -6.5466151
## 63                    -6.9946522
## 64                     6.9045917
## 65                     2.4790596
## 66                     1.5072616
## 67                     3.0288613
## 68                     0.7386895
## 69                    -0.2732132
## 70                    -0.2543372
## 71                    -5.8070154
## 72                    -3.4383673
## 73                    -3.5368930
## 74                    -5.1236880
## 75                    -3.6180666
## 76                     6.6350172
## 77                     0.5130691
## 78                     0.4074916
## 79                    -1.8627472
## 80                    -6.5704605
## 81                     8.6455665
## 82                     4.4223505
## 83                     3.9850735
## 84                     2.9621934
## 85                    -1.3352777
## 86                    -1.4920864
## 87                    -2.4705996
## 88                    -3.5617876
## 89                    -4.0534494
## 90                    -4.9777262
## 91                    -5.6076159
## 92                     7.2597455
## 93                     3.7625920
## 94                     4.3145737
## 95                     2.2593586
## 96                     3.2245813
## 97                    -0.5246057
## 98                    -1.8442554
## 99                    -1.2508650
## 100                   -1.2999283
```

4.2 R^2

```
R2(predicted_wins, ff_test$WINS)
```

```
## [1] 0.9287359
```

```
R2(pm, ff_test$MOV)
```

```
## [1] 0.9707142
```

R^2 is the coefficient of determination. It is a measure of the goodness of fit of a model. Both the **lmWINS** and **lmMOV** models perform almost as well on the test data as they did on the training data they were originally fit on. This is a good indicator that our model generalize well, as it predicted well on data it was not trained on.

R^2 values in training and test data sets

Model	Training	Test
lmWINS	0.9437	0.9300
lmMOV	0.9785	0.9783

4.3 Mean Absolute Error

```
MAE(predicted_wins, ff_test$WINS)
```

```
## [1] 2.545743
```

```
MAE(pm, ff_test$MOV)
```

```
## [1] 0.5472033
```

The MAE values for the **lmWINS** model fit on the test data is 3.392 and for the **lmMOV** model fit on the test data is 0.673. MAE represents the average absolute difference between actual and predicted outcomes.

4.4 Root Mean Squared Error

```
RMSE(predicted_wins, ff_test$WINS)
```

```
## [1] 3.260744
```

```
RMSE(pm, ff_test$MOV)
```

```
## [1] 0.7257216
```

The RMSE values for the **lmWINS** model fit on the test data is 3.392 and for the **lmMOV** model fit on the test data is 0.673. RMSE represents the root of the average squared difference between actual and predicted outcomes.

5. Weights of the four factors

We can calculate the weights of the four factors in our **lmWINS** and **lmMOV** models by calculating the average of each of the coefficents for the four factors.

5.1 Weights for the **lmWINS** model

```
sum_all_lmWINS = (3.81049 + 3.63928 + 1.11257 + 0.68903 + 3.86393 + 2.98585 + 0.87871 +
0.81222)
sum_all_lmWINS
```

```
## [1] 17.79208
```

```
sum_eFG_lmWINS = (3.81049 + 3.86393)
weight_eFG_lmWINS = 100 * (sum_eFG_lmWINS / sum_all_lmWINS)
weight_eFG_lmWINS
```

```
## [1] 43.13391
```

```
sum_TOV_lmWINS =(3.63928 + 2.98585)
weight_TOV_lmWINS = 100 * (sum_TOV_lmWINS / sum_all_lmWINS)
weight_TOV_lmWINS
```

```
## [1] 37.2364
```

```
sum_REB_lmWINS = (1.11257 + 0.87871)
weight_REB_lmWINS = 100 * (sum_REB_lmWINS / sum_all_lmWINS)
weight_REB_lmWINS
```

```
## [1] 11.19195
```

```
sum_FTF_lmWINS = (0.68903 + 0.81222)
weight_FTF_lmWINS = 100 * (sum_FTF_lmWINS / sum_all_lmWINS)
weight_FTF_lmWINS
```

```
## [1] 8.437743
```

Weights for the **lmWINS** model: 43.13% shooting, 37.24% turnovers, 11.19% rebounding, 8.44% foul rate

5.2 Weights for the **lmMOV** model

```
sum_all_lmMOV = (1.42327 + 1.35036 + 0.44414 + 0.28314 + 1.42177 + 1.19145 + 0.41600 +
0.27912)
sum_all_lmMOV
```

```
## [1] 6.80925
```

```
sum_eFG_lmMOV = (1.42327 + 1.42177)
weight_eFG_lmMOV = 100 * (sum_eFG_lmMOV / sum_all_lmMOV)
weight_eFG_lmMOV
```

```
## [1] 41.78199
```

```
sum_TOV_lmMOV =(1.35036 + 1.19145)
weight_TOV_lmMOV = 100 * (sum_TOV_lmMOV / sum_all_lmMOV)
weight_TOV_lmMOV
```

```
## [1] 37.32878
```

```
sum_REB_lmMOV = (0.44414 + 0.41600)
weight_REB_lmMOV = 100 * (sum_REB_lmMOV / sum_all_lmMOV)
weight_REB_lmMOV
```

```
## [1] 12.63193
```

```
sum_FTF_lmMOV = (0.28314 + 0.27912)
weight_FTF_lmMOV = 100 * (sum_FTF_lmMOV / sum_all_lmMOV)
weight_FTF_lmMOV
```

```
## [1] 8.257297
```

Weights for the **lmMOV** model: 41.78% shooting, 37.33% turnovers, 12.63% rebounding, 8.26% foul rate

5.3 Comparing Model Weights to Oliver's

Factor	**Oliver**	**lmWINS**	**lmMOV**
Shooting	40.00%	43.13%	41.78%
Turnovers	25.00%	37.24%	37.33%
Rebounding	20.00%	11.19%	12.63%

Factor	Oliver	lmWINS	lmMOV
Foul Rate	15.00%	8.44%	8.26%

The values we achieved are reasonably close to those Oliver proposed. Shooting is the most important factor for winning games in the NBA. Turnovers are very close in importance as the second most important factor. This makes sense intuitively from a very high level as if you don't have possession of the ball, you can't score the ball and therefore win a game. In the **lmWINS** and **lmMOV** models, we also place rebounding and foul rate as the 3rd and 4th most important factors respectively. However, the weights that we assign those factors in our model is about ~ 55% - 63% of the weight that Oliver assigned.

6. Four Factors historical changes

6.1 Preparing data sets for visualizations

```
four_factors_historical <- four_factors_scaled

four_factors_hist <- read_excel("per_100_possessions.xlsx")
four_factors_hist_new <- four_factors_hist %>% dplyr::rename(eFG = `eFG%`,
                                TO = `TOV%`,
                                OREB = `ORB%`,
                                FTF = `FT/FGA`)
```

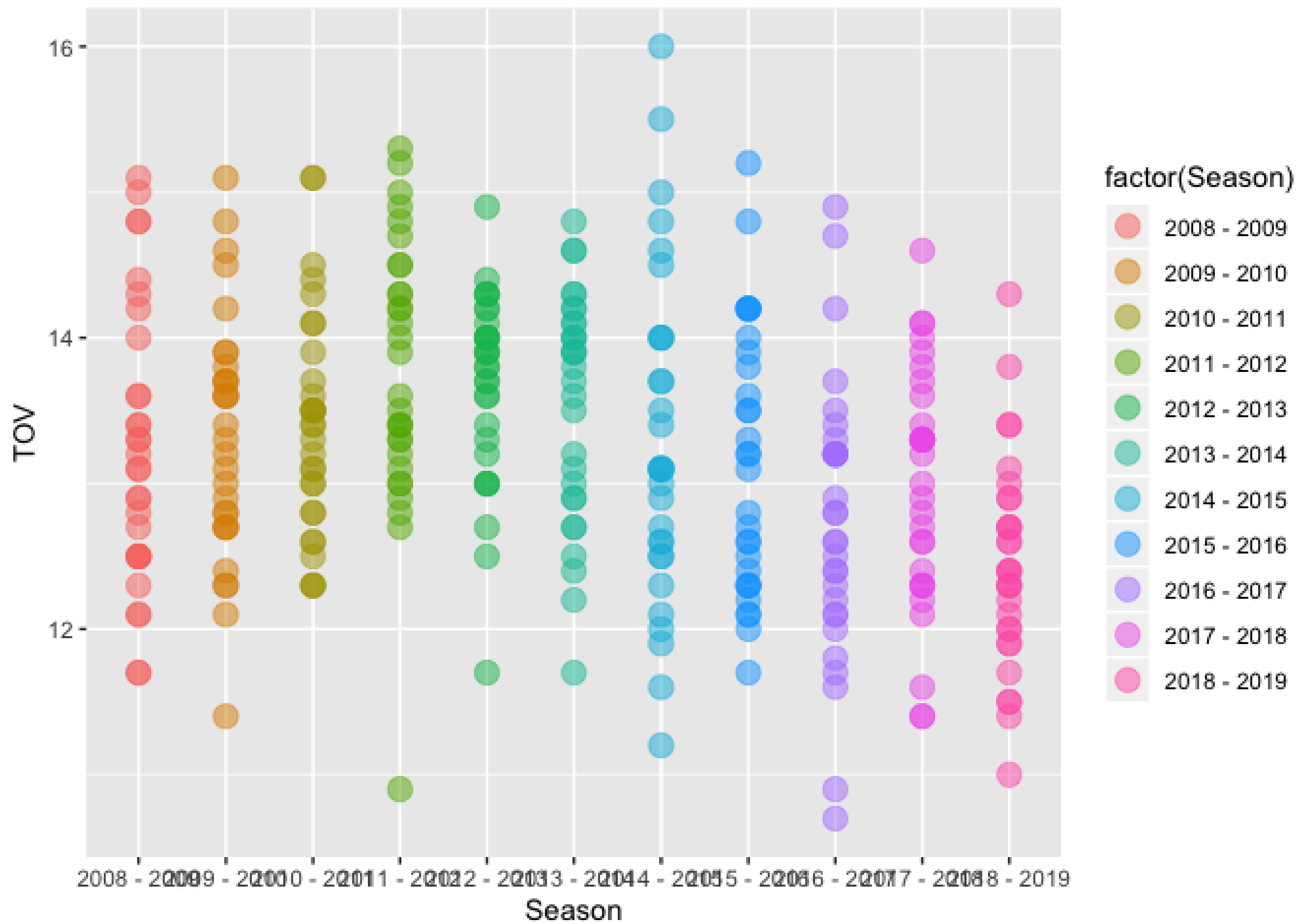
6.2 Scatter plots of the four factors for all NBA teams from the 2008 to 2018 seasons

6.2.1 `eFG`

```
eFG_all_hist <- ggplot(data = four_factors_historical, aes(x = Season, y = eFG))

eFG_all_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```

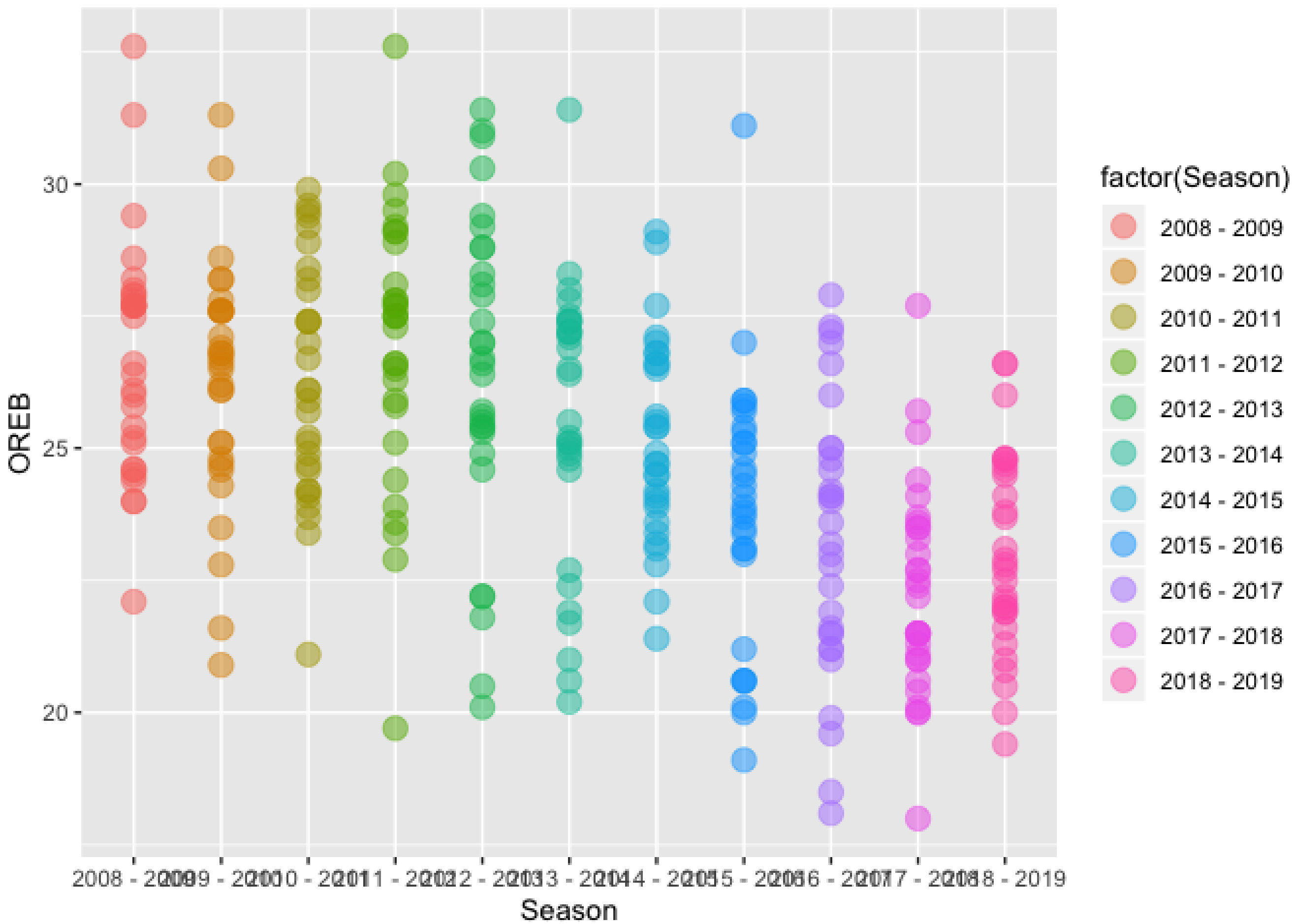
6.2.2 TOV

```
TOV_all_hist <- ggplot(data = four_factors_historical, aes(x = Season, y = TOV))

TOV_all_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



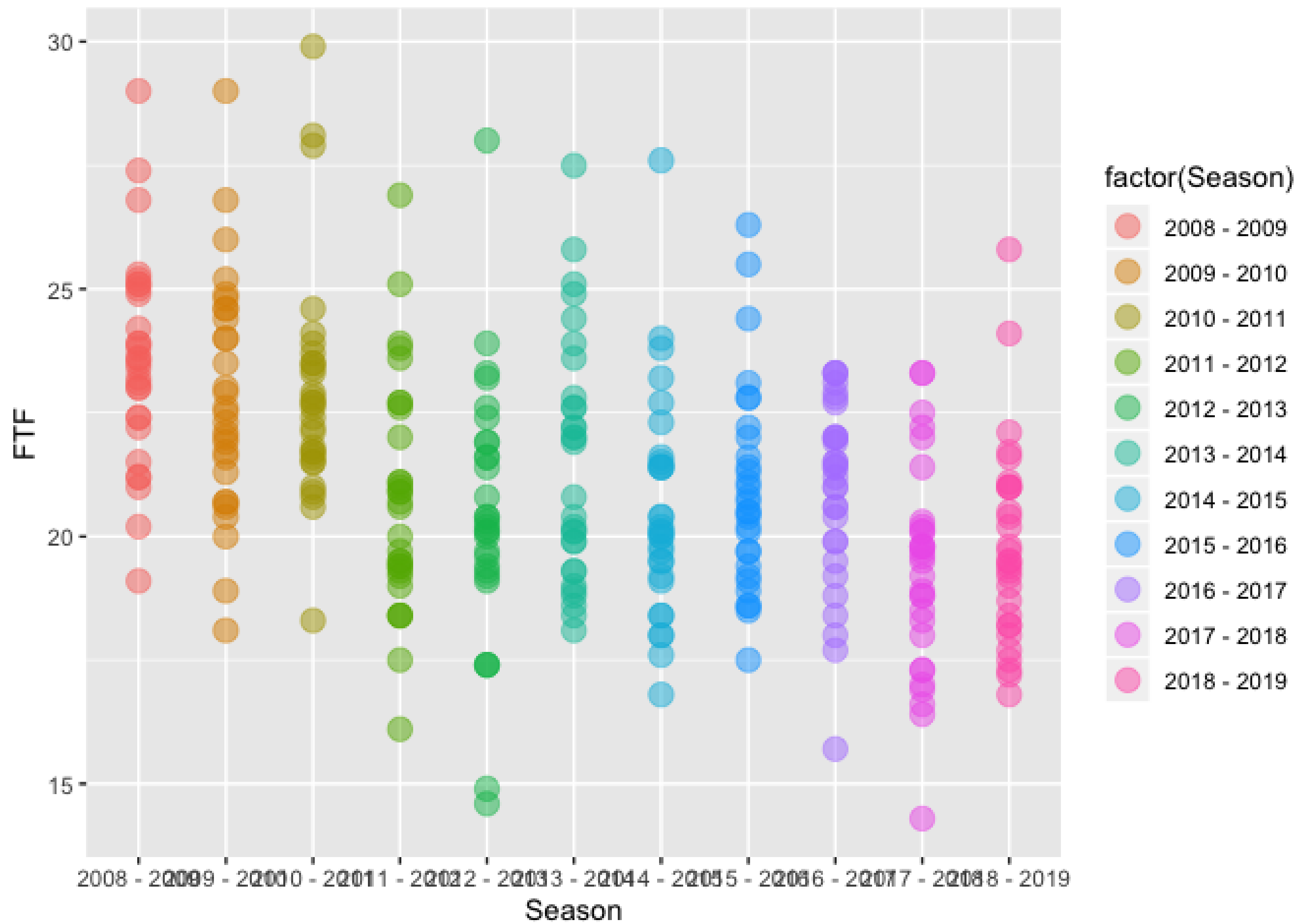
6.2.3 OREB

```
OREB_all_hist <- ggplot(data = four_factors_historical, aes(x = Season, y = OREB))

OREB_all_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



6.2.4 FTF

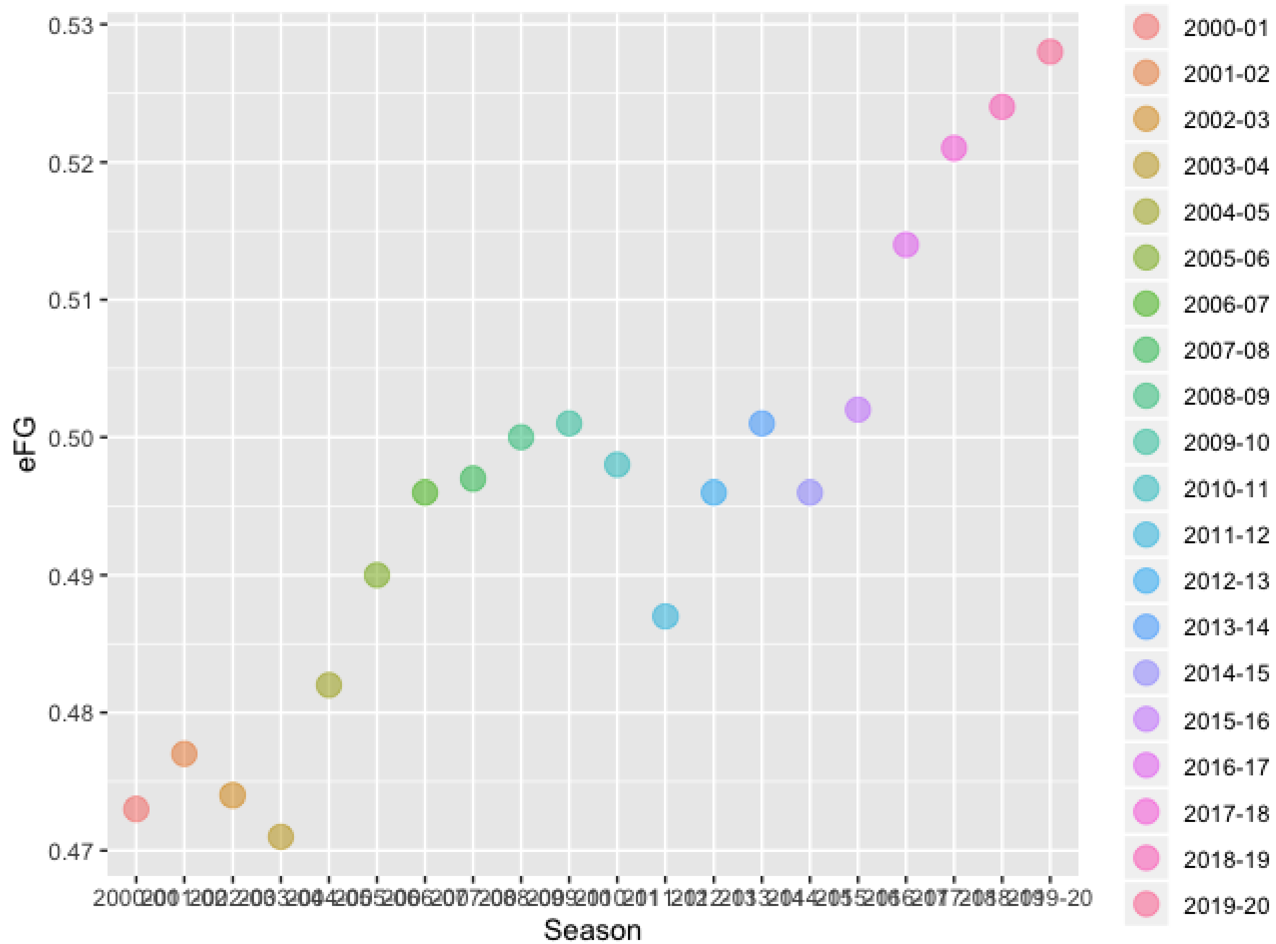
```
FTF_all_hist <- ggplot(data = four_factors_historical, aes(x = Season, y = FTF))

FTF_all_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



6.3 Scatter plots of league averages of the four factors from the 2001 to 2019 seasons

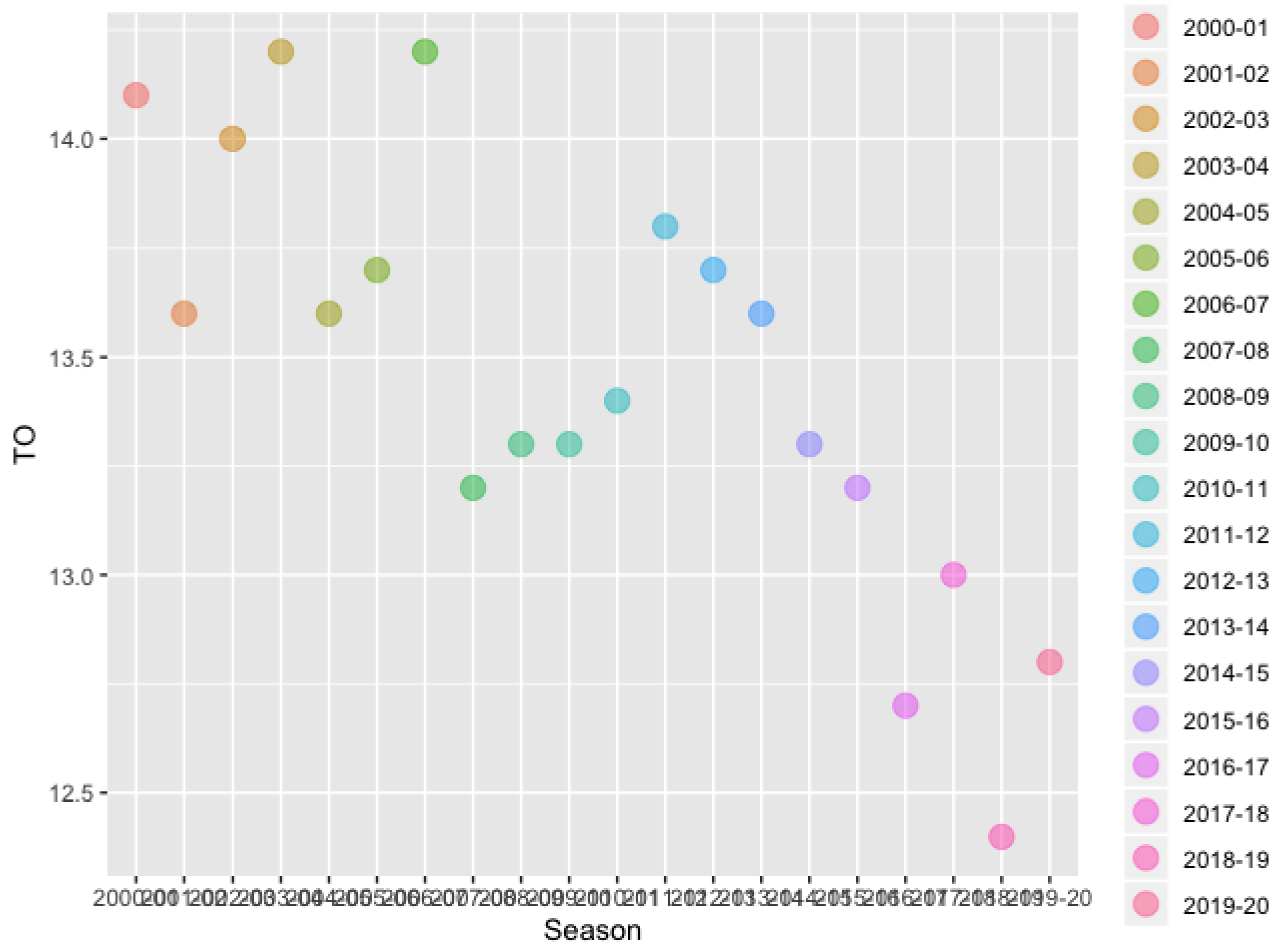
6.3.1 eFG

```
eFG_ave_hist <- ggplot(data = four_factors_hist_new, aes(x = Season, y = eFG))

eFG_ave_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



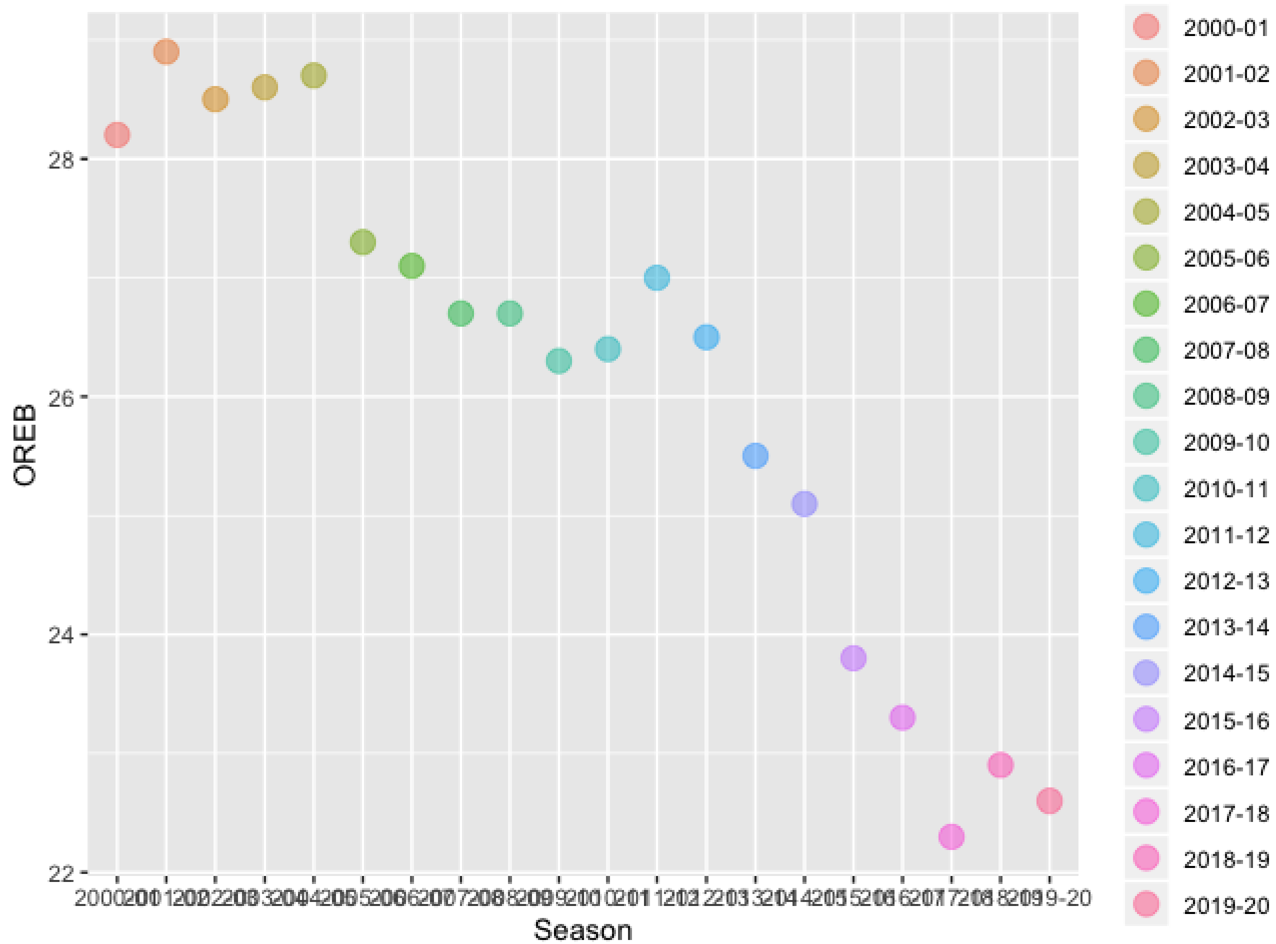
6.3.2 TOV

```
TOV_ave_hist <- ggplot(data = four_factors_hist_new, aes(x = Season, y = TO))

TOV_ave_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



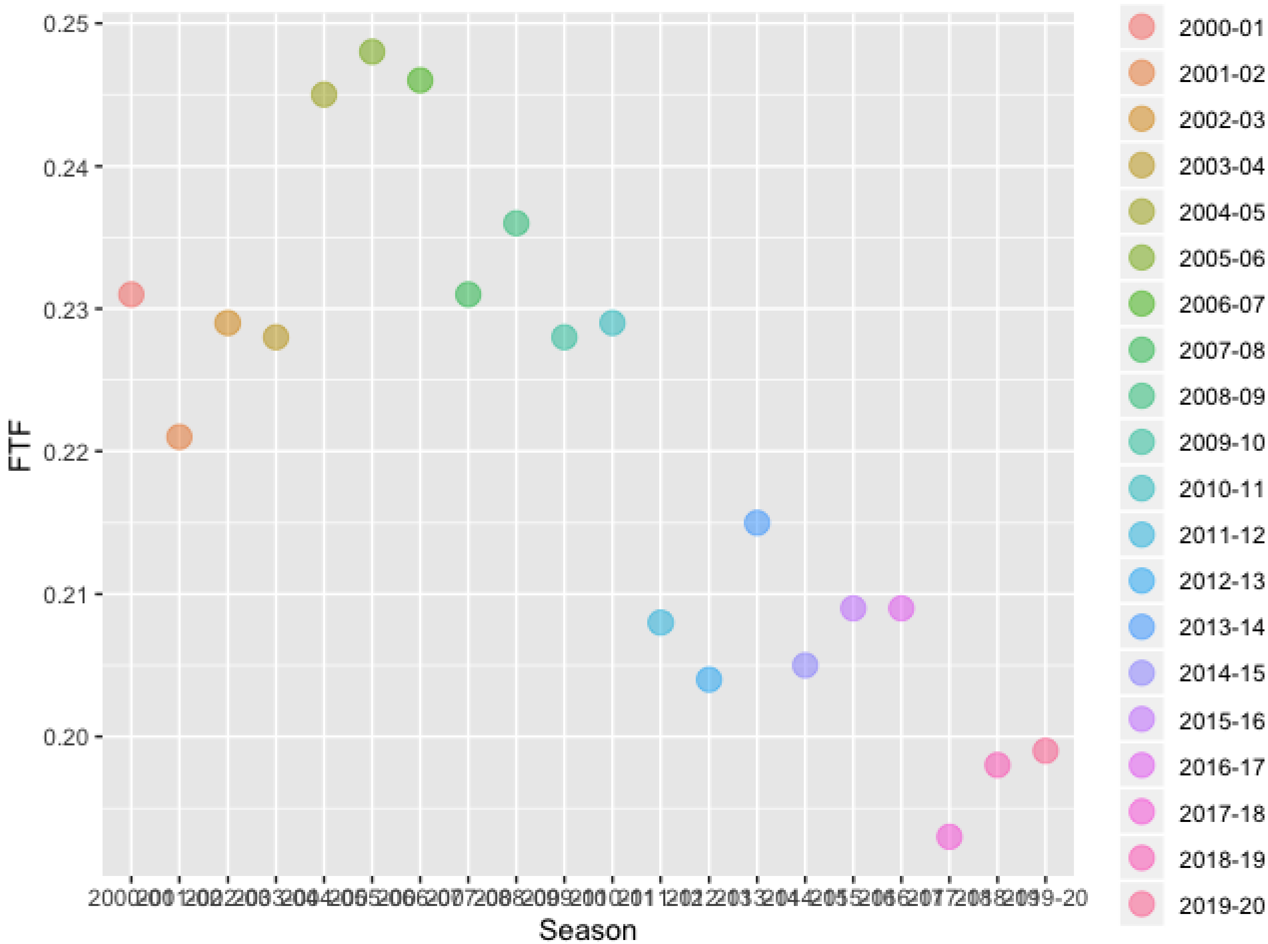
6.3.3 OREB

```
OREB_ave_hist <- ggplot(data = four_factors_hist_new, aes(x = Season, y = OREB))

OREB_ave_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



6.3.4 FTF

```
FTF_ave_hist <- ggplot(data = four_factors_hist_new, aes(x = Season, y = FTF))

FTF_ave_hist + geom_point(alpha=1/2, size=4,
 aes(color=factor(Season)))
```



0.25
0.24
0.23
0.22
0.21
0.20
FTF
Season
2019-20
2000-01
2001-02
2002-03
2003-04
2004-05
2005-06
2006-07
2007-08
2008-09
2009-10
2010-11
2011-12
2012-13
2013-14
2014-15
2015-16
2016-17
2017-18
2018-19
2019-20