Jonathan Malone

Abstract

Based on research into currently employed NBA performance models, I have come up with my own model to determine the best player in the NBA. My model considers points, total rebounds, assists, steals, blocks, and turnovers, normalized over a 36-minute time period. Depending on the position played, these statistics are multiplied by custom weights based on a one-hundred-point scale, and then divided by one hundred to produce a more manageable total rating. The results of the model are as follows:

Giannis Antetokounmpo	20.6378
Joel Embiid	19.5513
Luka Dončić	18.8025
Shai Gilgeous-Alexander	17.6481
Ja Morant	17.2258

Four of the five are expected results, but Shai Gilgeous-Alexander is an interesting inclusion. Since the provided data didn't consider team record, Gilgeous-Alexander made the top five in this model because of his outstanding individual performance.

Resources

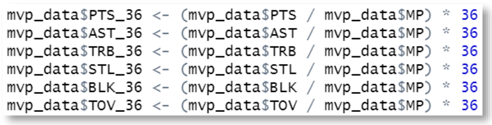
Regression Modeling--PART C, D, & E.pptx

https://www.statmuse.com/nba/ask/2023-league-average-by-position

https://www.geeksforgeeks.org/matrix-multiplication-in-r/#

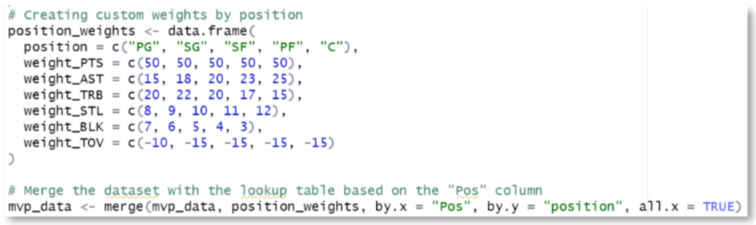
Benedic, Dustin – NBA statistical expert

The first parameter I wanted to consider was playing time. To be eligible for MVP, I believe a player should play a minimum of sixty games and average at least thirty minutes per game. Player health is the number one asset they bring to their team. I removed players from the data that did not meet these criteria. Next, I wanted to create several new predictor variables. To put the players on a level playing field, I calculated their points, assists, rebounds, steals, blocks, and turnovers per 36 minutes. Players on the most elite teams usually play less minutes per game, as they are benched during the end of blowouts to preserve their health. This prevents them from accumulating as many stats as players who play in more competitive games.

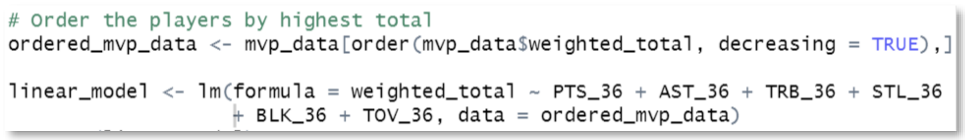


```
mvp_data$PTS_36 <- (mvp_data$PTS / mvp_data$MP) * 36
mvp_data$AST_36 <- (mvp_data$AST / mvp_data$MP) * 36
mvp_data$TRB_36 <- (mvp_data$TRB / mvp_data$MP) * 36
mvp_data$STL_36 <- (mvp_data$STL / mvp_data$MP) * 36
mvp_data$BLK_36 <- (mvp_data$BLK / mvp_data$MP) * 36
mvp_data$TOV_36 <- (mvp_data$TOV / mvp_data$MP) * 36
```

Next, I created custom weights by position. Assists should count less for point guards, as it is an expected stat, whereas a center or power forward who can also create assists should be rewarded at a higher level. The same logic goes for rebounds by guards compared to their bigger teammates who play closer to the basket, making rebounds easier to get. I heavily penalized turnovers, although to a lesser extent by point guards who handle the basketball more and will naturally accumulate more turnovers than other players. I then formulated the total by adding the newly created statistics multiplied by their weights and dividing by one hundred to deal with a smaller number. I ordered the results and fitted this to my first linear model.



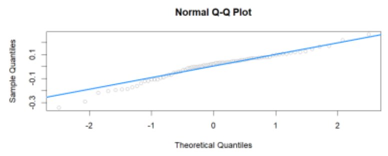
```
# Creating custom weights by position
position_weights <- data.frame(
  position = c("PG", "SG", "SF", "PF", "C"),
  weight_PTS = c(50, 50, 50, 50, 50),
  weight_AST = c(15, 18, 20, 23, 25),
  weight_TRB = c(20, 22, 20, 17, 15),
  weight_STL = c(8, 9, 10, 11, 12),
  weight_BLK = c(7, 6, 5, 4, 3),
  weight_TOV = c(-10, -15, -15, -15, -15)
)

# Merge the dataset with the lookup table based on the "Pos" column
mvp_data <- merge(mvp_data, position_weights, by.x = "Pos", by.y = "position", all.x = TRUE)
```

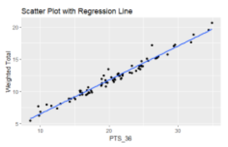


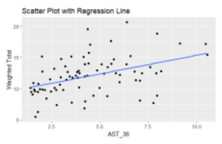
```
# Order the players by highest total
ordered_mvp_data <- mvp_data[order(mvp_data$weighted_total, decreasing = TRUE),]

linear_model <- lm(formula = weighted_total ~ PTS_36 + AST_36 + TRB_36 + STL_36
                   + BLK_36 + TOV_36, data = ordered_mvp_data)
```

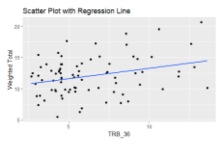
The first results of my model looked quite promising. I used the step function to make sure all of my variables were necessary to the model. Then I checked the model summary. Most of my variables were statistically significant with p-values much less than 0.05, but my intercept and BLK_36 predictor p-values were too high. I adjusted the weights for blocks and reran the results until I had all variables with a p-value less than 0.05, making them all statistically significant. However, the intercept was still high. Of particular interest was the R-squared value, showing an extraordinarily high value of 0.9986. This indicated the model potentially fit quite well.

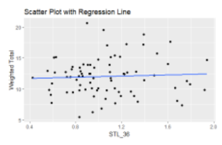
To continue my analysis, I ran a Q_Q plot of the residuals to see if they were normally distributed. I also ran scatter plots for each predictor variable against the response variable. The PTS_36 variable had an almost perfect regression line, but the other variables were off by varying degrees.

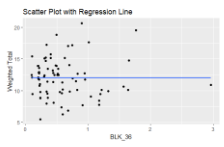




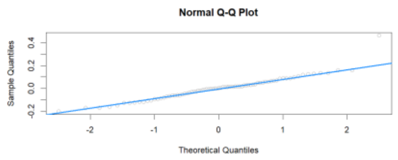




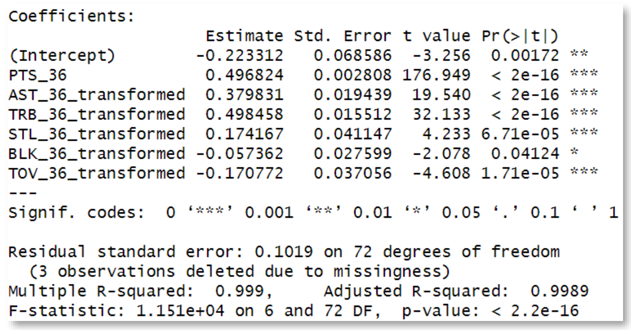




To decide how to transform the data, I used a Box Cox Normality Plot for each of the variables. PTS_36 had a value of almost one, showing that it already had a normal distribution. The other variables had values between 0.2 and 0.6, so I used power transformations for each statistic instead of log transformations. The normalization for each statistic was much more visible in the new scatter plots, and the residuals followed the theoretical quantities almost perfectly.



To finish my model, I fitted an updated model with the transformed data and printed the summary to see if it was an upgrade over the original model. The intercept was now in line with the predictor variables, having a statistically significant p-value of 0.0017. All variables showed correlation with the weighted total, with PTS_36 having the highest correlation coefficient of 0.4985. The R-squared value actually increased to a staggering 0.999.



```
Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)         -0.223312   0.068586  -3.256  0.00172 ** 
PTS_36               0.496824   0.002808 176.949  < 2e-16 ***
AST_36_transformed   0.379831   0.019439  19.540  < 2e-16 ***
TRB_36_transformed   0.498458   0.015512  32.133  < 2e-16 ***
STL_36_transformed   0.174167   0.041147   4.233 6.71e-05 ***
BLK_36_transformed  -0.057362   0.027599  -2.078  0.04124 *  
TOV_36_transformed  -0.170772   0.037056  -4.608 1.71e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1019 on 72 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.999,	Adjusted R-squared:  0.9989 
F-statistic: 1.151e+04 on 6 and 72 DF,  p-value: < 2.2e-16
```

After printing the results of the final model, the results were in. The sports media had incorrectly voted Joel Embiid as 2022-2023 NBA MVP. According to my data, the third-place finisher, Giannis Antetokounmpo, should have won his second overall MVP award.

Giannis Antetokounmpo	20.6378
Joel Embiid	19.5513
Luka Dončić	18.8025
Shai Gilgeous-Alexander	17.6481
Ja Morant	17.2258

Code

```
library(ggplot2)
library(dplyr)
library(MASS)

#Loading and summarizing original data
original_data <- read.csv('nba_2022_2023.csv')
summary(original_data)

#Removing players who played less than 60 games or less than 30mpg
mvp_data <- subset(original_data, G >= 60 & MP >= 30)
summary(mvp_data)

#Convert stats to per 36 minute values
mvp_data$PTS_36 <- (mvp_data$PTS / mvp_data$MP) * 36
mvp_data$AST_36 <- (mvp_data$AST / mvp_data$MP) * 36
mvp_data$TRB_36 <- (mvp_data$TRB / mvp_data$MP) * 36
mvp_data$STL_36 <- (mvp_data$STL / mvp_data$MP) * 36
mvp_data$BLK_36 <- (mvp_data$BLK / mvp_data$MP) * 36
mvp_data$TOV_36 <- (mvp_data$TOV / mvp_data$MP) * 36

#Remove unnecessary columns
mvp_data <- subset(mvp_data, select = -c(PTS, AST, TRB, ORB, DRB, STL, BLK, TOV, ORB, PF, Age, GS, Tm))
head(mvp_data)

#Creating custom weights by position
position_weights <- data.frame(
  position = c("PG", "SG", "SF", "PF", "C"),
  weight_PTS = c(50, 50, 50, 50, 50),
  weight_AST = c(15, 18, 20, 23, 25),
  weight_TRB = c(20, 22, 20, 17, 15),
  weight_STL = c(8, 9, 10, 11, 12),
  weight_BLK = c(7, 6, 5, 4, 3),
  weight_TOV = c(-10, -15, -15, -15, -15)
)

#Merge the dataset with the lookup table based on the "Pos" column
mvp_data <- merge(mvp_data, position_weights, by.x = "Pos", by.y = "position", all.x = TRUE)

#Create a new column for the weighted total
mvp_data$weighted_total <- ((mvp_data$PTS_36 * mvp_data$weight_PTS) +
  (mvp_data$AST_36 * mvp_data$weight_AST) +
  (mvp_data$TRB_36 * mvp_data$weight_TRB) +
  (mvp_data$STL_36 * mvp_data$weight_STL) +
  (mvp_data$BLK_36 * mvp_data$weight_BLK) +
  (mvp_data$TOV_36 * mvp_data$weight_TOV)) / 100

#Order the players by highest total
ordered_mvp_data <- mvp_data[order(mvp_data$weighted_total, decreasing = TRUE),]

#First linear model
linear_model <- lm(formula = weighted_total ~ PTS_36 + AST_36 + TRB_36 + STL_36
                  + BLK_36 + TOV_36, data = ordered_mvp_data)
summary(linear_model)

#Checking to see if all variables were necessary
First <- lm(weighted_total~1, data = ordered_mvp_data)
All <- lm(weighted_total ~ PTS_36 + AST_36 + TRB_36 + STL_36 + BLK_36 + TOV_36, data =
ordered_mvp_data)
step(First, direction = "forward", scope = formula(All))
step(All, direction = "backward", scope = formula(First))
step(First, direction = "both", scope = formula(All))

AIC(linear_model)
BIC(linear_model)
```

```
#Extract residuals from the linear model
residuals <- residuals(linear_model)

#Create a Q-Q plot of the first model residuals
qqnorm(residuals)
qqline(residuals, col = "red")

#Plotting first model residuals
par(mfrow = c(1, 2))
plot(fitted(linear_model), resid(linear_model), col = "grey", pch = 20,
    xlab = "Fitted", ylab = "Residuals", main = "Fitted Vs Residuals")
abline(h = 0, col = "darkorange", lwd = 2)
qqnorm(resid(linear_model), main = "Normal Q-Q Plot", col = "darkgrey")
qqline(resid(linear_model), col = "dodgerblue", lwd = 2)

#Scatter plot of each stat in the original model. I ran it for each by
#replacing the variables in the code below
ggplot(ordered_mvp_data, aes(x = TOV_36, y = weighted_total)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter Plot with Regression Line",
      x = "TOV_36",
      y = "Weighted Total")

#Box Cox plot on entire model
linear_model = lm(weighted_total ~ PTS_36 + AST_36 + TRB_36 + STL_36 + BLK_36 +
                TOV_36, data = ordered_mvp_data)
boxcox(Fit1.linear_model, plotit = TRUE)

#Box Cox plot on each predictor variable. I replaced the variable each time
#and reran the lines
linear_model = lm(weighted_total ~ AST_36, data = ordered_mvp_data)
boxcox(Fit1.linear_model, plotit = TRUE)
```

```
#Transformed values based on Box Cox results

ordered_mvp_data$PTS_36_transformed <- ((ordered_mvp_data$PTS_36^0.2) - 1) / 0.2

ordered_mvp_data$AST_36_transformed <- ((ordered_mvp_data$AST_36^0.5) - 1) / 0.5

ordered_mvp_data$TRB_36_transformed <- ((ordered_mvp_data$TRB_36^0.5) - 1) / 0.5

ordered_mvp_data$STL_36_transformed <- ((ordered_mvp_data$STL_36^0.4) - 1) / 0.4

ordered_mvp_data$BLK_36_transformed <- ((ordered_mvp_data$BLK_36^0.3) - 1) / 0.3

ordered_mvp_data$TOV_36_transformed <- ((ordered_mvp_data$TOV_36^0.6) - 1) / 0.6


#Second linear model

Fit2.linear_model = lm(weighted_total ~ PTS_36 + AST_36_transformed + TRB_36_transformed +
STL_36_transformed + BLK_36_transformed + TOV_36_transformed, data = ordered_mvp_data)

boxcox(Fit2.linear_model, plotit = TRUE)


#Extract residuals from the second linear model

residuals2 <- residuals(Fit2.linear_model)


#Plotting second model residuals

plot(fitted(Fit2.linear_model), resid(Fit2.linear_model), col = "grey", pch = 20,

    xlab = "Fitted", ylab = "Residuals", main = "Fitted Vs Residuals")

abline(h = 0, col = "darkorange", lwd = 2)

qqnorm(resid(Fit2.linear_model), main = "Normal Q-Q Plot", col = "darkgrey")

qqline(resid(Fit2.linear_model), col = "dodgerblue", lwd = 2)


#Scatter plot of each stat in the second model. I ran it for each by

#replacing the variables in the code below

ggplot(ordered_mvp_data, aes(x = BLK_36_transformed, y = weighted_total)) +

  geom_point() +

  geom_smooth(method = "lm", se = FALSE) +

  labs(title = "Scatter Plot with Regression Line",

       x = "BLK_36_transformed",

       y = "Weighted Total")


#Making sure each variable was still necessary after transformations
```

```
First <- lm(weighted_total~1, data = ordered_mvp_data)

All <- lm(weighted_total ~ PTS_36 + AST_36_transformed + TRB_36_transformed + STL_36_transformed +
BLK_36_transformed + TOV_36_transformed, data = ordered_mvp_data)

step(First, direction = "forward", scope = formula(All))

step(All, direction = "backward", scope = formula(First))

step(First, direction = "both", scope = formula(All))


#Printing summary of second model

summary(Fit2.linear_model)


#Printing model results for MVP

head(ordered_mvp_data[c("Player", "weighted_total")])
```