Predicting Points Scored Per Game for National Basketball Association Players Using Linear Regression

By:

Matthew Brigham

Cleveland State University

Statistics Department

STA 567

11-28-2020

Table of Contents

1. Background and Goals

Over the past several decades, there has been a dramatic increase in the amount of analytics and data being strewn around on platforms like ESPN and others. With society becoming more data-driven, more people have tried to understand sports through a different lens. Sports analysts (like those at ESPN) as well as professional sports teams, players, gamblers and gambling organizations, and the common viewer can all use these analytics to understand how games are played. Analysts have been interested in developing models to predict different performance metrics as well as certain outcomes. Topics of interest include predicting the probability of a win, how many wins a team will end the season with, how well players will perform in a particular game, and so forth.

There is an abundance of data and due to an increase in investments, AI technology, and resources available for analysis, there is an increase in interest in the sports analytics market. (Markets and Markets, 2020). The global sports analytics market was valued at $774.6 million in 2018. As the world becomes more data driven, this market is expected to increase by 31.2% year after year until 2025. (Grandview Research, 2019) Sports teams are known to use this data to make decisions about how to play against other teams and which players will be the most useful. (Austin, 2015)

There is a lot of interest in predicting performance and outcomes for NBA basketball. One benefit of analyzing basketball data is that the same metrics and statistics are applied to all of the players, unlike NFL football, where a quarterback's performance might be analyzed by passing yards while it would not make sense to judge a defensive safety with the same metric. (Wheeler, 2012) Sports analytics are also useful for people and organizations who want to

gamble on outcomes. There are different types of bets that can be placed on how many points a player will score in a game, which team will win a game, which player will win MVP, and many more. In the context of creating a model and predicting outcomes, it is always of interest to understand which variables are useful predictors and how the accuracy can be improved to help provide knowledge and a competitive edge.

2. Introduction and Statement of Problem

There are many reasons why it is useful to understand how well a player might perform. Having data on how well all of the players on a team perform would provide insight into which team might win a game. In the context of gambling, bets can be placed on the winner of a sport event as well as on how well a particular player will perform in a single game. In this paper, the relationship between a variety of performance metrics and the points scored per game will be explored. A linear regression model will be used to determine if a relationship can be accurately modeled to predict how many points a player will score per game.

3. Description of Data

There are numerous basketball metrics and online sources of data for players and teams across many different seasons. The source of data used in this paper is from basketball-reference.com. This website provided data for the NBA seasons from 2016 through 2019 including the following season performance totals for every player, as summarized in Table 1 below.

Performance Metric Abbreviation	Variable Definition
MP	Minutes Played
FGA	Field Goals Attempted (includes 2-Point and 3-Point Shot Attempts)
FTA	Free Throw Shots Attempted
ORB	Offensive Rebounds
DRB	Defensive Rebounds
TRB	Total Rebounds (sum of Offensive and Defensive Rebounds)
TOV	Turnovers
PF	Personal Fouls
PTS	Points Scored

Table 1: Definition of variables in dataset that are common basketball performance metrics. The values for each metric are season totals, e.g. PTS tells the total number of points scored over the course of a single season.

Several modifications were made to the data files retrieved from the website. Data from the seasons 2016-2017, 2017-2018, and 2018-2019 were all collected and then compiled into a single dataset. A new categorical variable was introduced to help identify the seasons. A value of (1) represents the 2016-2017 season, (2) represents the 2017-2018 season, and (3) represents the 2018-2019 season. The season totals for each player were converted to performance per game by dividing the values by the respective number of games played. There were 486 players in the 2016-2017 season. The players from the 2017-2018 and 2018-2019 seasons were filtered to ensure only new players were added to prevent duplicates. This was done to avoid highly correlated observations since having duplicate observations from different times might introduce a serial time effect. Additionally, the data set was reduced to only players who averaged at least 8 points per game to reduce skew. Of the 486 total players in the 2016-2017 season, only 146 met this criteria. The players who averaged less than 8 points per game are far more numerous and would bias a linear model to under-predict points scored. If the model is to be used for sports bets, most betting websites only offer over/under bets on the

top-performing players, therefore only players who are productive are considered. A criteria of 8 points per game was selected since it yielded the best results. The players from the 2016-2017 season were used to train a linear regression model to estimate the points scored per game for a player. The new players from the 2017-2018 and 2018-2019 seasons were used to test the predictive accuracy of the linear model.

4. Statistical Analysis

A linear regression model was fit using the following variables: field goals attempted, free throws attempted, effective field goal percentage, and free throw percentage. All of the numeric and continuous variables were examined in the data set to see which were the most significant predictors of points scored per game. Significance tests were used to determine whether to use predictors in the model. Various model checking metrics were used to assess the fit. The results shown in the following section reflect the removal of 3 outliers. This will be discussed further in section 4.4. The model was tested on new player data from the 2017-2018 and 2018-2019 seasons.

4.1 Log-Transformation of Data

Preliminary analysis of the data showed linear and non-linear relationships between each predictor and the response, points per game. This was discovered by examining scatterplots of the response versus each predictor. Since there appeared to be skew and non-normality in the density plot shown in Figure 5, a log transformation was applied to the data.

Some of the scatterplots are shown below in Figures 1-4 (only the scatterplots for variables that ended up being used in the model are shown). There is a linear relationship between PTS and FGA as well as FTA (Figures 1 and 2). Figure 3 shows that there isn't a strong relationship between PTS and EFG. In Figure 4, there is increasing standard deviation and a non-linear relationship between variables; this aforementioned pattern can be corrected with alternative forms of regression.

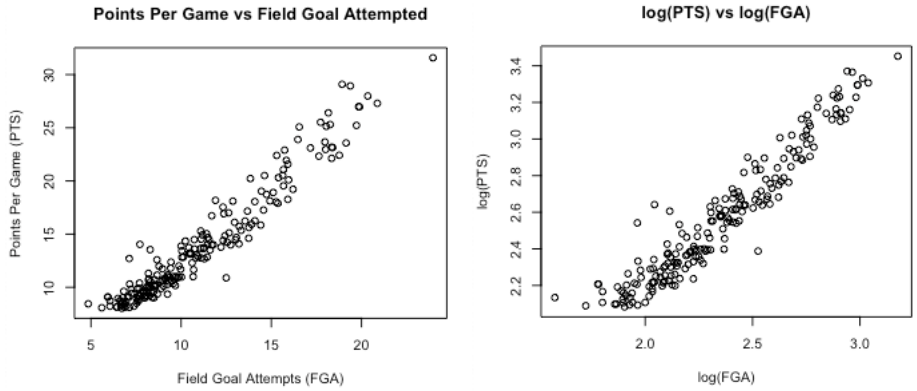

Figure 1: The linear relationship was preserved after log-transformation. It will be discussed in a later section that PTS will need to be log-transformed to create a linear relationship with other variables, for this reason, FGA should be log-transformed, too. Means are a straight line and standard deviations appear to be mostly constant.

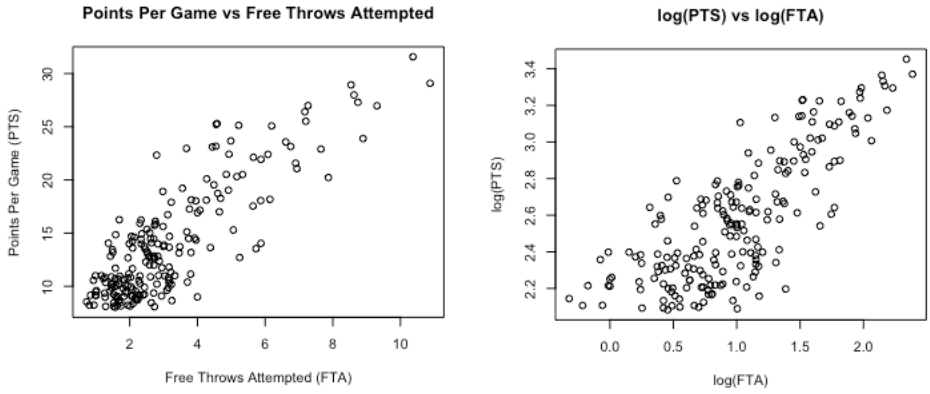

Figure 2: Comparing the two scatterplots, it is evident that a log transformation of both PTS and FTA helped to create a linear relationship. The means appear to be a straight line and the standard deviations are not perfectly constant.

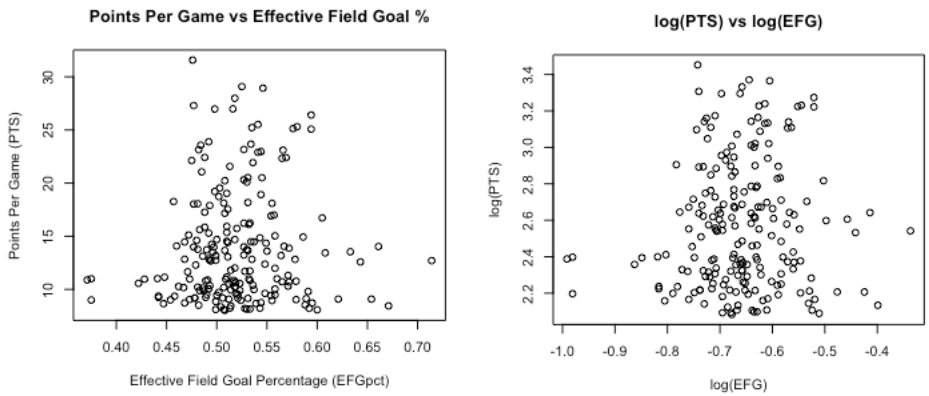

Figure 3: In this scatterplot, there appears to be no significant relationship between the two variables before or after transformation. In the model, EFGpct was transformed since all of the other predictors and response were transformed. Several models were considered without the inclusion of this variable, but the predictive accuracy improved with its inclusion.

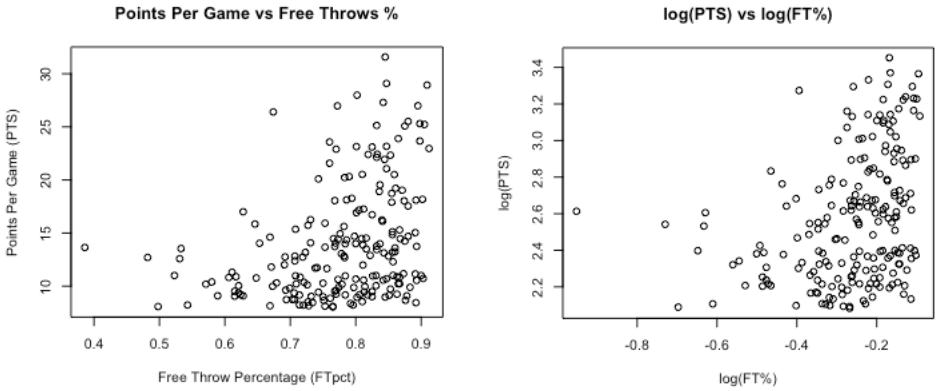

Figure 4: In the untransformed scatter plot, there is a non-linear trend and standard deviation seems to increase. Log-transformation does not appear to correct either the linearity or standard deviation concerns.

From the Shapiro Tests for normality, we see that most of the variables are significantly different from the normal distribution. Table 2 shows the variables (transformed and untransformed) that do follow a normal distribution. The variables that are not displayed do not follow a normal distribution. The density plot in Figure 5 below of PTS and logPTS graphically show that they do not follow a strict normal distribution due to skew.

Normally Distributed Predictors
PF
logORB
logDRB
logTRB
logPF
logFTA
logTOV

Table 2: A list of variables that have normal distributions.

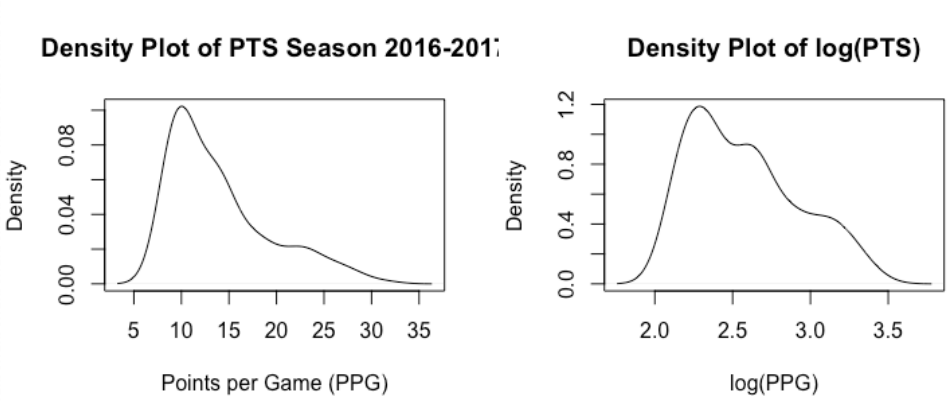

Figure 5: Observe that the distribution is skewed and has a long right tail. It is not normally distributed.

4.2 Variable Selection

Many different combinations of variables (with and without interaction) were considered as well as performing different types of transformations. The best model was determined based on backwards selection and tested with prediction accuracy. Using a t-test and evaluating the p-value/significance of each β_i coefficient, the least significant coefficient was dropped until a final reduced model was determined. This reduced model used the log-

transformed predictors: FGA, EFG, and the interaction term of FTA and FTpct. The R-output for this linear model is displayed below in Figure 6.

```
Call:
lm(formula = logPTS ~ logFGA + logFTA * logFTpct + logEFG, data = dptsfil_train)

Residuals:
      Min        1Q    Median        3Q       Max
-0.016173 -0.010053 -0.006058  0.003079  0.062990

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.018675   0.013298  76.604  < 2e-16 ***
logFGA          0.837902   0.004952 169.204  < 2e-16 ***
logFTA          0.176595   0.004999  35.325  < 2e-16 ***
logFTpct        0.108568   0.020093   5.403 1.89e-07 ***
logEFG          0.833920   0.011909  70.023  < 2e-16 ***
logFTA:logFTpct 0.056308   0.016829   3.346 0.000984 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01506 on 195 degrees of freedom
Multiple R-squared:  0.9982,    Adjusted R-squared:  0.9981
F-statistic: 2.142e+04 on 5 and 195 DF,  p-value: < 2.2e-16
```

Figure 6: A summary of the final reduced linear model is shown. All of the p-values are significant, therefore, backwards selection stopped at this point.

The ANOVA table in Figure 7 shows that the reduced model is adequate and can be used instead of the full model. It is preferable to use the reduced model because it requires fewer terms and is simpler to use.

Analysis of Variance Table

Model 1: logPTS ~ logFGA + logFTA * logFTpct + logEFG
Model 2: logPTS ~ logFGA + logFGpct + logFTA * logFTpct + logTRB + logMP + logEFG + logTOV + logPF

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	195	0.044204				
2	190	0.042221	5	0.0019829	1.7847	0.1178

Figure 7: The ANOVA Table compares the two models (full and reduced). The p-value of 0.1178 suggests that the reduced model does an adequate job modeling the data.

The final reduced model if given below as Equation 1. An Extra Sum of Squares F-Test was used to compare the reduced and the full models to test if the additional coefficients are significantly different from zero. This F-test was performed using ANOVA and yielded a test statistic of 1.7847 with a p-value of 0.1178. These results indicate that the reduced model is adequate and can be used in place of the full model.

$$logPTS = 0.834logFGA + 0.826logEFG + 0.181logFTA + 0.103logFTpct + 0.061logFTA * logFTpct \quad (1)$$

The value of R^2 was 0.9982, which means that 99.82% of the variation in points scored per game is explained by the predictors. Even though a high R^2 is desirable, it is concerning because it could mean that the model is overfit or biased. Ultimately, assessing the predictive capabilities will determine if the model is a poor or good fit.

4.3 Model Checking

There are several ways to evaluate the fit of a model, including VIF, mean square error, mean absolute error, QQ-plots, and residual plots. The Variance Inflation Factor (VIF) is a measure of multicollinearity. It is calculated by performing a regression of each predictor onto all of the others and getting the R^2. The VIF of each predictor is given in Table 2 below. High values for VIF indicate potential for multicollinearity, but there is no consensus for what point is considered concerning for VIF. For this analysis, VIF > 4 will be deemed concerning. Referring to Table 3, all of the VIF values are below the threshold of 4 except for the interaction term and

its constituent variables. This is not concerning because a strong correlation between logFTA and logFTA*logFTpct as well as logFTpct and logFTA*logFTpct is expected.

Predictor	VIF
logFGA	2.354
logFTA	6.753
logFTpct	6.424
logEFG	1.082
logFTA*logFTpct	10.122

Table 3: Displays the VIF values for each predictor in the model. VIF values greater than 4 are assumed to imply a risk of multicollinearity.

Mean square error and mean absolute error are metrics that measure how good of a fit the model is. It is important to note that outliers do influence mean square error due to the square term, while mean absolute error is robust. Table 4 shows the mean square error and mean absolute error for the model. The error for LogPTS was converted to PTS by taking the exponential. The interpretation is that the on average, the prediction is 1.011 points away from the actual points scored per game for all players used to build the model.

	LogPts	PTS
Mean Square Error	0.00023	1.000
Mean Absolute Error	0.01118	1.011

Table 4: Displays the mean square error and mean absolute error for the model. These values represent the distances between the actual and predicted points scored per game for players.

The final assessment of fit is through the use of residual plots. Residual plots help to check the assumptions for linear regression of linearity, normality, equal variances as well as assist in the detection of outliers. Plotting residuals versus fitted values can show linearity and equal variances. Figure 8 shows the plot of residuals versus fitted; it can be seen that there is a linear pattern, therefore the linear assumption of regression is satisfied. Also, from Figure 8, it can be seen that the equal variances assumption is satisfied because the spread is consistent. Looking at Figure 9, the QQ-Plot tells information about whether the predictors and response come from the normal distribution. It is clear that the existence of tails is responsible for the departure from normality. To test the extent of the impact on the model of the lack of normality, the accuracy of predictions will be analyzed. Since the regression technique is least squares estimation, the model is robust to non-normal data, however, the existence of tails and potential outliers may impact the predictive ability.

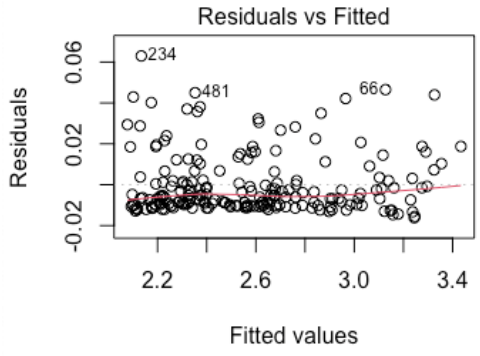

Figure 8: The plot of residuals versus fitted values shows that the linearity and equal variance assumptions for linear regression are satisfied.

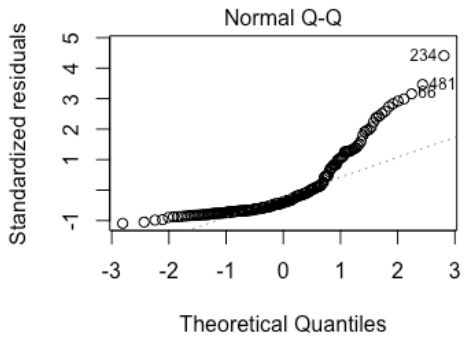

Figure 9: The QQ-Plot shows that the data is non-normal and there are tails. Some outliers may still remain in the data set.

4.4 Outliers

There are three metrics used to identify influential observations that should be considered as possible outliers: Cook's Distance, Studentized Residuals, and Leverage. From Figure 10 below, it is seen that one observation (Player 408) has a Cook's Distance greater than one. This observation was removed since it was also flagged influential by leverage and Studentized Residuals. Two other observations were also removed based on their leverage values and Studentized Residuals.

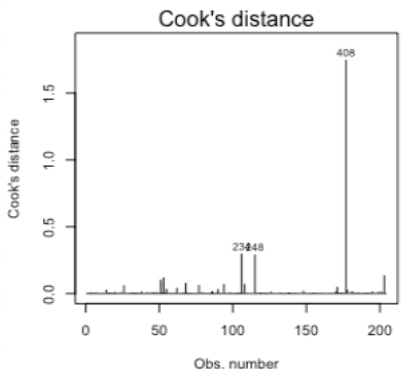

Figure 10: Cook's Distance values greater than one are deemed influential. Only one observation is flagged influential based on this metric.

After the removal of the three outliers, 13 observations were flagged as influential based on studentized residuals and 15 observations were flagged based on leverage values. Only one of the influential leverage values is also an influential studentized residual.

4.5 Prediction

The dataset used to test the accuracy of the predictions consisted of players from the 2017-2018 and 2018-2019 seasons who did not play or did not score at least 8 points per game in the 2016-2017 season. It is important to understand how the predictive accuracy changes from season to season since the model was trained on only a single season. It is assumed that players from different seasons would have performed similarly if they played in the 2016-2017 season with their skill level and experience. The performance of the model's predictive accuracy was quantified using the difference between the predicted and the actual points per game (mean absolute error). The mean absolute error is similar and slightly better for new seasons (refer to Table 5) than for the data it was trained on. One possible explanation for this is that the model is biased to underpredict the points scored per game because the data is more densely distributed at lower points per games values (refer to Figure 5). Since it is more likely that new players scored just above the threshold of 8 points per game, than say 30 points per game, the new data set is testing mostly players whose PTS distribution is centered near that of the training data set (fewer observations in the tails). See Figure 11 below to compare the densities of the points scored per game for seasons 2016-2017, 2017-2018, and 2018-2019:

Season	Mean Absolute Error
2017-2018	11.5%
2018-2019	13.2%

Table 5: This table displays the mean absolute error for the predicted values for different seasons.

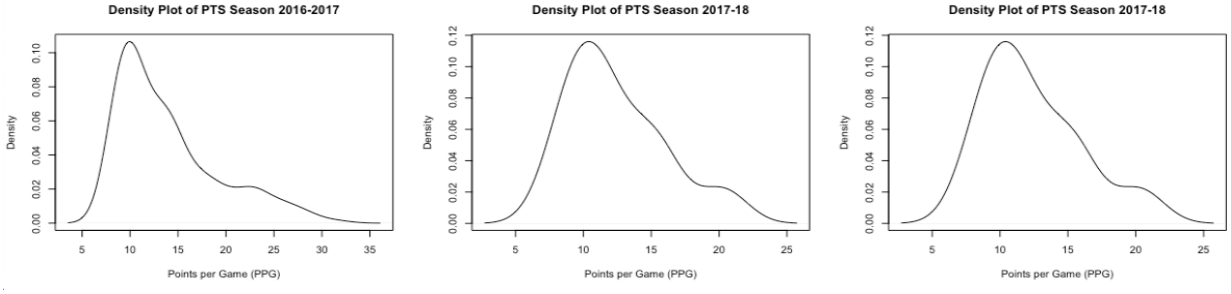

Figure 11: Notice that the peak densities are near the same values. This explains the bias of the model to underpredict points scored per game for players who are in the right tail. Since most of the players in the testing dataset score in the peak, their predictions are going to be more accurate.

5. Statistical Conclusions

The model suggests that out of all the predictors in the original dataset, the most useful ones for predicting the points per game are FGA, FTA, FTpct, and EFG. A useful model was developed for predicting the points per game using these variables. An extra sums of squares F-test (p-value = 0.1178, F-statistic = 1.7847) revealed that the reduced model that uses only four predictors is adequate compared to the full model. For a test set of 21 observations from the 2017-2018 season, all 21 observations had a residual error of less than 3%. For a test set of 43 observations from the 2018-2019 season, 40 observations had less than 3% error. These results suggest that the performance of the players from a single season can be used to predict the points scored per game by players in other seasons.

One potential issue is that the data is skewed. Most of the players are not high-scoring, and even after filtering out the players who averaged less than 8 points per game, there is still

some skew. This property results in the under-prediction of points per game for players who are in the right tail.

8. Conclusions and Future Work

The model's weakness comes from the distribution of the data as well as the difficulty in measuring certain factors that impact performance. Not all of the variables had strong linear associations with the points scored per game and normality were other areas of concern—even after transformation. Least squares estimation is robust to non-normality but the influence of outliers can have an impact on the predictive ability. The variables were kept, however, because they still did an adequate job predicting the points scored per game. The model may have benefitted from the inclusion of other variables that reflected psychological factors. If a player was injured for part of the season, and came back that season, then their performance may be inconsistent. Other variables that affect performance that are difficult to quantify are confidence, strength of competition, how well the player matches up to the specific player(s) that are guarding them, and other personal or private aspects. Aside from these, the model proved to be accurate in predicting the points scored per game for players in the same season as well as others with small error.

Future work may try to address the linearity/normality concerns of the data. Alternative data transformation techniques may solve the issue of linearity and normality, such as the Box-Cox transformation. (Goh et al, 2019) Another improvement would be to use more data from other seasons to build the model and ignore more low-scoring players to help reduce skew.

Caution should be used here as using data from the same players over different seasons would violate independence assumptions and would be a longitudinal study with repeated measures. Attempting to implement different types of regression or machine learning, such as weighted regression or support vector machines, may yield stronger results. (Ramsey & Schafer, 2013) Different modeling methods may help to address the concerns with the data that transformation could not fix. Determining a model that predicts points scored per game is useful as it is, however, the results can be used to answer other research questions, such as wins and losses. It is possible to use a linear model in conjunction with a win/loss classifier to predict whether a team will win or lose. (Wheeler, 2012) It may be of interest to model players' performance over time and considering opponent skill. It has been shown that player PER and Win Shares can successfully predict wins and losses. (Yang, 2015) These metrics are descriptive of the skill level of the player. This would allow for points scored predictions for individual games as opposed to a season average. The results of this paper suggest that a reasonable prediction of specific game performance is possible.

9. References

Grandview Research. Sports Analytics Market Size: Industry Growth Report, 2019-2025. (2019, November). Retrieved November 28, 2020, from https://www.grandviewresearch.com/industry-analysis/sports-analytics-market

Austin. (2015, November 22). San Antonio Spurs' Data-Driven Approach to Win Games. Retrieved November 29, 2020, from https://digital.hbs.edu/platform-digit/submission/san-antonio-spurs-data-driven-approach-to-win-games/

Wheeler, K. (2012). *Predicting NBA Player Performance* (Tech.). Retrieved November, 2020, from http://cs229.stanford.edu/proj2012/Wheeler-PredictingNBAPlayerPerformance.pdf

Ramsey, F. L., & Schafer, D. W. (2013). *The Statistical sleuth: A course in methods of data analysis*. Australia: Brooks/Cole, Cengage Learning.

Yang, S. (2015). Predicting Regular Season Results of NBA Teams Based on Regression Analysis of Common Basketball Statistics. Retrieved November, 2020, from https://www.stat.berkeley.edu/~aldous/Research/Ugrad/Stanley_Yang%20_Thesis.pdf

Markets and Markets. Sports Analytics Market by Sports Type (Individual and Team), Component, Application (Performance Analysis, Player Fitness and Safety, Player and Team Valuation, and Fan Engagement), Deployment Model, and Region - Global Forecast to 2024. (2020). Marketsandmarkets.Com. https://www.marketsandmarkets.com/Market-Reports/sports-analytics-market-35276513.html#:~:text=In%20North%20America%2C%20the%20high,companies%20in%20real%2Dtime%20analytics.

Goh et al. (2019). *Predicting the performance of the players in NBA Players by divided regression analysis*. Malaysian Journal of Fundamental and Applied Sciences. https://pdfs.semanticscholar.org/a690/c83e59ead11bfa6a3e4d11a6058ff8fa9ed9.pdf

Allison, P. (2019, November 27). *When Can You Safely Ignore Multicollinearity?* Statistical Horizons. https://statisticalhorizons.com/multicollinearity

R Code

```
# Matthew Brigham
# STA 567
# Final Project Code
# November 2020

# Import Data

    d = read.csv("STA 567 Final Data Set.csv", header = T)

    d= d[which(d$Season == 1), ] # select season 2016-2017
    s1617 = d

# Clean Data

    # transform season totals to per game averages
    d$MP = d$MP/d$G     # minutes played
    d$FGA = d$FGA/d$G   # field goals attempted
    d$FTA = d$FTA/d$G   # free throws attempted
    d$ORB = d$ORB/d$G   # offensive rebounds
    d$DRB = d$DRB/d$G   # defensive rebounds
    d$TRB = d$TRB/d$G   # total rebounds
    d$TOV = d$TOV/d$G   # turnovers
    d$PF = d$PF/d$G     # personal fouls
    d$PTS = d$PTS/d$G   # points

    # log transform variables
    d$logORB = log(d$ORB)
    d$logDRB = log(d$DRB)
    d$logTRB = log(d$TRB)
    d$logMP = log(d$MP)
    d$logPF = log(d$PF)
    d$logPTS = log(d$PTS)
    d$logFGA = log(d$FGA)
    d$logFTA = log(d$FTA)
    d$logTOV = log(d$TOV)
    d$logEFG = log(d$eFG.)
    d$logFGpct = log(d$FG.)
    d$logFTpct = log(d$FT.)


    # Remove NA, NaN, and Inf values
    d = d[!is.infinite(d$logORB), ]
    d = d[!is.infinite(d$logDRB), ]
    d = d[!is.infinite(d$logTRB), ]
    d = d[!is.infinite(d$logPF), ]
    d = d[!is.infinite(d$logPTS), ]
    d = d[!is.infinite(d$logTOV), ]
    d = d[!is.infinite(d$logFGA), ]
    d = d[!is.infinite(d$logFTA), ]
    d = d[!is.infinite(d$logEFG), ]
```

```
d = d[!is.infinite(d$logFTpct), ]
d = d[!is.na(d$logEFG), ]

# Exploratory data analysis

    # Pairwise Scatterplots of Transformed and Untransformed Predictors

    x1 = d[c("PTS", "FT.", "FG.", "MP", "FGA", "FTA", "eFG."  , "ORB"  ,
           "DRB" , "TRB" , "TOV", "PF")]
    pairs(x1)

    # x2 = d[c("PTS", "logMP", "FGA", "logFTA", "eFG.", "logORB",  "logDRB",
    #         "logTRB", "TOV", "logPF")]
    # pairs(x2)
    #
    # x3 = d[c("logPTS", "logMP", "FGA", "logFTA", "eFG.", "logORB", "logDRB",
    #         "logTRB", "TOV", "logPF")]
    # pairs(x3)

    x4 =  d[c("logPTS", "logFTpct", "logFGpct", "logMP", "logFGA", "logFTA",
            "logEFG", "logORB",  "logDRB", "logTRB", "logTOV", "logPF")]
    pairs(x4)


    # Remove outliers
    d = d[-(which(d$Rk == 408)), ]
    d = d[-(which(d$Rk == 248)), ]
    d = d[-(which(d$Rk == 122)), ]

    # Remove Players who scored less than 8 points per game
    train = d
    dptsfil_train = train[-which(train$PTS < 8), ]

    # Individual scatter plots of response~predictor (Untransformed)
    par(mfrow=c(1,4))
    plot(dptsfil_train$PTS ~ dptsfil_train$FGA, main =
          "Points Per Game vs Field Goal Attempted", xlab = "Field Goal Attempts (FGA)",
        ylab = "Points Per Game (PTS)")
    plot(dptsfil_train$PTS ~ dptsfil_train$FTA, main =
          "Points Per Game vs Free Throws Attempted", xlab = "Free Throws Attempted (FTA)",
        ylab = "Points Per Game (PTS)")
    plot(dptsfil_train$PTS ~ dptsfil_train$eFG., main =
          "Points Per Game vs Effective Field Goal %", xlab = "Effective Field Goal Percentage (EFGpct)",
        ylab = "Points Per Game (PTS)")
    plot(dptsfil_train$PTS ~ dptsfil_train$FT., main =
          "Points Per Game vs Free Throws %", xlab = "Free Throw Percentage (FTpct)",
        ylab = "Points Per Game (PTS)")

    # Individual scatter plots of response~predictor (Transformed)
    plot(dptsfil_train$logPTS ~ dptsfil_train$logFGA, main = "log(PTS) vs log(FGA)",
        xlab = "log(FGA)", ylab = "log(PTS)")
    plot(dptsfil_train$logPTS ~ dptsfil_train$logFTA, main = "log(PTS) vs log(FTA)",
        xlab = "log(FTA)", ylab = "log(PTS)")
```

```
    plot(dptsfil_train$logPTS ~ dptsfil_train$logEFG, main = "log(PTS) vs log(EFG)",
       xlab = "log(EFG)", ylab = "log(PTS)")
    plot(dptsfil_train$logPTS ~ (dptsfil_train$logFTpct), main = "log(PTS) vs log(FT%)",
       xlab = "log(FT%)", ylab = "log(PTS)")

    # Density plots (look for normality, transform to correct normality)
    par(mfrow=c(1,2))
    plot(density(dptsfil_train$PTS), xlab = "Points per Game (PPG)",
       ylab = "Density", main = "Density Plot of PTS Season 2016-2017")
    plot(density(dptsfil_train$logPTS), xlab = "log(PPG)", ylab = "Density",
       main = "Density Plot of log(PTS)")


################################################################################
################################################################################
#Create a Model of Log-Transformed Variables With Interaction on 2016-2017 Season
################################################################################
################################################################################

# Variable Selection

    log_full_lm = lm(logPTS ~ logFGA+ logFGpct + logFTA*logFTpct + logTRB + logMP + logEFG + logTOV + logPF,
data = dptsfil_train)
    summary(log_full_lm)

    #removed logMP
    lm_step1 = lm(logPTS ~ logFGA + logFGpct + logFTA*logFTpct + logTRB + logEFG + logTOV + logPF, data =
dptsfil_train)
    summary(lm_step1)

    #removed logTRB
    lm_step2 = lm(logPTS ~ logFGA + logFGpct + logFTA*logFTpct + logEFG + logTOV + logPF, data = dptsfil_train)
    summary(lm_step2)

    #removed logFGpct
    lm_step3 = lm(logPTS ~ logFGA + logFTA*logFTpct + logEFG + logPF + logTOV, data = dptsfil_train)
    summary(lm_step3)

    #removed logTOV
    lm_step4 = lm(logPTS ~ logFGA + logFTA*logFTpct + logEFG + logPF, data = dptsfil_train)
    summary(lm_step4)

    #removed logPF
    lm_step5 = lm(logPTS ~ logFGA + logFTA*logFTpct + logEFG, data = dptsfil_train)
    summary(lm_step5)

# Assess Model

    # Adjusted R-Squared
    summary(lm_step5)$r.squared
    summary(lm_step5)$adj.r.squared

    # Examine residual plots of reduced model
```

```
par(mfrow=c(1,4))
plot(lm_step5, which = 1)
plot(lm_step5, which = 2)
plot(lm_step5, which = 3)
plot(lm_step5, which = 4)

# ANOVA - F test of overall significance of regression
reduced = lm_step5
full = log_full_lm
anova(reduced, full)  # p value is 0.1178

# Test for Mulitcollinearity
vif(lm_step5)

# Mean Square Error
SSE_red = anova(reduced)["Residuals", "Sum Sq"]
MSE = SSE_red/(nrow(dptsfil_train) - 8 - 1)
MSE

# Mean Absolute Error
mean(abs(resid(lm_step5))) # 0.01117734

# min/max error/residual
min(resid(lm_step5)) # -0.01675371
max(resid(lm_step5)) # 0.03751517

# Calculate Cooks Value, Studentized Resid, and Leverage
dptsfil_train$cooks = with(dptsfil_train, cooks.distance(lm_step5))
dptsfil_train$hat = with(dptsfil_train, hatvalues(lm_step5))
dptsfil_train$studres = with(dptsfil_train, studres(lm_step5))

# Find Influential Observations
nrow(dptsfil_train[which(dptsfil_train$cooks>=0.9), ]) #0 potential influential data point
nrow(dptsfil_train[which(dptsfil_train$studres >= 2), ]) #9 potential influential data points
nrow(dptsfil_train[which(dptsfil_train$hat > 2*length(lm_step5$coefficients)/nrow(dptsfil_train)), ]) #9
potential influential data points
#dptsfil_train[which(dptsfil_train$cooks>=0.9), ] #observations with Cook's values near or greater than 1
#dptsfil_train[which(dptsfil_train$studres >= 2), ] #observations with studentized residuals greater than 2
#dptsfil_train[which(dptsfil_train$hat > 2*length(lm_step5$coefficients)/nrow(dptsfil_train)), ] #observations
with influential leverages


# Assessment of Fit on Training Data

# log_pred = predict(lm_step5, newdata = dptsfil_test, se.fit = TRUE)
# pred_Pts = exp(log_pred$fit)
#
# # Calculate Prediction Error
# error = 100*(pred_Pts - dptsfil_test$PTS)/dptsfil_test$PTS
# par(mfrow=c(2,1))
# plot(error ~ dptsfil_test$Rk, xlab = "observation #", ylab = "error %",
#      main = "Percentage Prediction Error by Player") # shows skewed to under predict PPG
# plot(abs(error) ~ dptsfil_test$Rk, xlab = "observation #", ylab = "error %",
```

```
    #     main = "Percentage Prediction Error by Player") # absolute value of error
    # abline(mean(abs(error)), 0) # average of 0.9059528% error with several extreme values

############################################################
############################################################
### Test model on 2017-2018 Season
############################################################
############################################################

# Import Data Set

    s1718 = d[which(d$Season == 2), ] # select season 2017-2018

    test2 = s1718

# Eliminate Duplicate Players

    i = 1
    listindex = 1
    list = c(NA)
    for (j in 1:length(test2$Player)) {
      for (i in 1:length(s1617$Player)) {
        if (test2$Player[j] == s1617$Player[i]) {
          list[listindex] = j
          listindex = listindex + 1
        }
      }
    }
    test2 = s1718[-list,]

# Clean Data

    # Convert Season Totals to per game statistics
    test2$MP = test2$MP/test2$G     # minutes playetest2
    test2$FGA = test2$FGA/test2$G   # fieltest2 goals attemptetest2
    test2$FTA = test2$FTA/test2$G   # free throws attemptetest2
    test2$ORB = test2$ORB/test2$G   # offensive rebountest2s
    test2$test2RB = test2$DRB/test2$G   # test2efensive rebountest2s
    test2$TRB = test2$TRB/test2$G   # total rebounds
    test2$TOV = test2$TOV/test2$G   # turnovers
    test2$PF = test2$PF/test2$G     # personal fouls
    test2$PTS = test2$PTS/test2$G   # points

    # log transform variables
    test2$logORB = log(test2$ORB)
    test2$logDRB = log(test2$DRB)
    test2$logTRB = log(test2$TRB)
    test2$logMP = log(test2$MP)
    test2$logPF = log(test2$PF)
    test2$logPTS = log(test2$PTS)
    test2$logFGA = log(test2$FGA)
    test2$logFTA = log(test2$FTA)
    test2$logTOV = log(test2$TOV)
```

```
    test2$logEFG = log(test2$eFG.)
    test2$logFGpct = log(test2$FG.)
    test2$logFTpct = log(test2$FT.)

    # Remove possible Inf, NA, and NaN values
    test2 = test2[!is.infinite(test2$logORB), ]
    test2 = test2[!is.infinite(test2$logDRB), ]
    test2 = test2[!is.infinite(test2$logTRB), ]
    test2 = test2[!is.infinite(test2$logPF), ]
    test2 = test2[!is.infinite(test2$logPTS), ]
    test2 = test2[!is.infinite(test2$logTOV), ]
    test2 = test2[!is.infinite(test2$logFGA), ]
    test2 = test2[!is.infinite(test2$logFTA), ]
    test2 = test2[!is.infinite(test2$logEFG), ]
    test2 = test2[!is.infinite(test2$logFTpct), ]
    test2 = test2[!is.na(test2$logEFG), ]

    # Remove players who scored less than 8 points per game
    dptsfil_test2 = test2[-which(test2$PTS < 8), ]

#Get Size and Density Plot of Prediction Set

    nrow(dptsfil_test2)
    plot(density(dptsfil_test2$PTS), xlab = "Points per Game (PPG)",
       ylab = "Density", main = "Density Plot of PTS Season 2017-18")

# Prediction

    log_pred = predict(lm_step5, newdata = dptsfil_test2, se.fit = TRUE)
    pred_Pts = exp(log_pred$fit)

    # Calculate prediction error
    error2 = 100*(pred_Pts - dptsfil_test2$PTS)/dptsfil_test2$PTS
    error2 = error2[!is.na(error2)]
    par(mfrow=c(1,2))
    plot(abs(error2) ~ dptsfil_test2$Rk, xlab = "observation #", ylab = "error %",
       main = "Percentage Prediction Error by Player")
    abline(mean(abs(error2)), 0) # average of 1.387193% error with several extreme values
    mean(abs(error2))
    mean(abs(pred_Pts - dptsfil_test2$PTS)) #Mean absolute error of prediction

    plot(error2 ~ dptsfil_test2$Rk, xlab = "observation #", ylab = "error %",
       main = "Percentage Prediction Error by Player") #shows that predictions are skewed to underpredict ppg

# Get Number of predictions that are less than 3% error

    numobsless3pct = length(which(abs(error2)<3)) #number of predictions
    numobsless3pct
    numobsmore3pct = length(which(abs(error2)>=3))
    numobsmore3pct

############################################################
############################################################
```

```
### Test model on 2018-2019 Season
############################################################
############################################################

# Import Data Set
      s1819 = read.csv("2018-2019.csv", header = T)

      test2 = s1819

# Eliminate Duplicate players

      i = 1
      listindex = 1
      list = c(NA)

      for (j in 1:length(test2$Player)) {
        for (i in 1:length(s1617$Player)) {
          if (test2$Player[j] == s1617$Player[i]) {
            list[listindex] = j
             listindex = listindex + 1
          }
        }
      }

      test2 = s1819[-list,]

# Clean Data

      # Convert Season Totals to per game statistics
      test2$MP = test2$MP/test2$G     # minutes playetest2
      test2$FGA = test2$FGA/test2$G   # fieltest2 goals attemptetest2
      test2$FTA = test2$FTA/test2$G   # free throws attemptetest2
      test2$ORB = test2$ORB/test2$G   # offensive rebountest2s
      test2$test2RB = test2$DRB/test2$G   # test2efensive rebountest2s
      test2$TRB = test2$TRB/test2$G   # total rebounds
      test2$TOV = test2$TOV/test2$G   # turnovers
      test2$PF = test2$PF/test2$G     # personal fouls
      test2$PTS = test2$PTS/test2$G   # points

      # log transform variables
      test2$logORB = log(test2$ORB)
      test2$logDRB = log(test2$DRB)
      test2$logTRB = log(test2$TRB)
      test2$logMP = log(test2$MP)
      test2$logPF = log(test2$PF)
      test2$logPTS = log(test2$PTS)
      test2$logFGA = log(test2$FGA)
      test2$logFTA = log(test2$FTA)
      test2$logTOV = log(test2$TOV)
      test2$logEFG = log(test2$eFG.)
      test2$logFGpct = log(test2$FG.)
      test2$logFTpct = log(test2$FT.)
```

```
    # Remove possible Inf, NA, and NaN values
    test2 = test2[!is.infinite(test2$logORB), ]
    test2 = test2[!is.infinite(test2$logDRB), ]
    test2 = test2[!is.infinite(test2$logTRB), ]
    test2 = test2[!is.infinite(test2$logPF), ]
    test2 = test2[!is.infinite(test2$logPTS), ]
    test2 = test2[!is.infinite(test2$logTOV), ]
    test2 = test2[!is.infinite(test2$logFGA), ]
    test2 = test2[!is.infinite(test2$logFTA), ]
    test2 = test2[!is.infinite(test2$logEFG), ]
    test2 = test2[!is.infinite(test2$logFTpct), ]
    test2 = test2[!is.na(test2$logEFG), ]

    # Remove players who scored less than 8 points per game
    dptsfil_test2 = test2[-which(test2$PTS < 8), ]

# Prediction

    log_pred = predict(lm_step5, newdata = dptsfil_test2, se.fit = TRUE)
    pred_Pts = exp(log_pred$fit)

    # Calculate prediction error
    error3 = 100*(pred_Pts - dptsfil_test2$PTS)/dptsfil_test2$PTS
    error3 = error3[!is.na(error3)]
    par(mfrow=c(1,2))
    plot(abs(error3) ~ dptsfil_test2$Rk, xlab = "observation #", ylab = "error %",
       main = "Percentage Prediction Error by Player")
    abline(mean(abs(error3)), 0) # average of 1.387193% error with several extreme values
    mean(abs(error3))
    mean(abs(pred_Pts - dptsfil_test2$PTS)) #Mean absolute error of prediction

    plot(error3 ~ dptsfil_test2$Rk, xlab = "observation #", ylab = "error %",
       main = "Percentage Prediction Error by Player") #shows that predictions are skewed to underpredict ppg

# Get Number of predictions that are less than 3% error

    numobsless3pct = length(which(abs(error3)<3))
    numobsless3pct
    numobsmore3pct = length(which(abs(error3)>=3))
    numobsmore3pct

    nrow(dptsfil_test2)

    test2[which(error3< -8),]

    plot(density(dptsfil_test2$PTS), xlab = "Points per Game (PPG)",
       ylab = "Density", main = "Density Plot of PTS Season 2017-18")
    nrow(dptsfil_test2)
```