STA5939 Final Project

The Impact of Hustle Plays in Winning NBA Games

By: Omar Chraibi

Background

In the past ten years, especially within the sports sector, sports analytics has grown dramatically in popularity. Players and companies are now looking to sports analytics for answers and solutions to improved performance, analyzing data, and effectively attracting fans and consumers as sport becomes more competitive and advanced. Consider the game of baseball as the subject of the 2011 film Moneyball, which explores the idea of letting statistics tell the tale rather than a scout with 25 years of expertise.

When it comes down to the general structure of the National Basketball Association. The NBA began in 1946 with 11 teams and now has 30 teams thanks to a series of club additions, subtractions, and relocations. 29 teams are based in the United States, and one is in Canada. 30 teams are split into two conferences of three divisions each with five teams in the current league structure. time is given a salary cap and salary floor. If teams exceed the salary cap, they must pay a luxury tax. There is also a draft every year, and the pick order is by ascending order, with the worst record picking first. There is a lottery system for the worst teams to randomize the results. With these regulations and the implementation of advanced technology. The NBA organizations are finding different ways to be competitive. Organizations approach the game differently, some prefer the traditional way of thinking, without analytics. Others have implemented analytics to be the driving force of their team.

Daryl Morey, the Houston Rockets' general manager since 2007, is one of the people who truly advocated for data-driven basketball. The Rockets have established themselves as strong rivals and competitors in recent years. The Rockets reached the Western Conference finals in

2017–2018 despite suffering injuries. They finished in the top seed in a Western Conference full of talent while setting the record for the most 3-pointers made in a single season. The increase in 3-point shot attempts is one of the most significant ways that data analytics has altered basketball.

The plays of games and the shots made were analyzed by analysts to identify which shots would be more likely to help the team win. Players were taking a lot of long-range, ineffective 2-point attempts, according to one analysis. Analysts and coaching staff across the league started urging their team's shooters to take more shots from the 3-point line because there was a low chance that they would result in a basket. According to statistics, there is a 35% probability that a shooter who is average too good will make the shot. The possession will be valuable if a shooter attempts three shots and makes one of them. The important lesson is that the long-range effort is worthwhile because of the 50% return from the extra point at the 3-point line.

This concept by Daryl Morey had a significant effect on how organizations and players view the game of basketball. NBA teams began to realize that analytics can lead to a competitive advantage. Because of the analytics revelation, it began to spark the "analytics vs. eye test" debate. This debate is a constant topic of conversation in the NBA.

NBA analytics are developing on the court of professional basketball, much like in other professional sports. The NBA has also made a significant investment in analytics. Nearly every team now has a front office NBA analytics department. Cameras that film every movement of the ball and all 10 players 25 times per second are used to gather data. Here are a few strategies teams are using data analytics to improve their competitiveness. Analyses are done on data from high school and college plays. Performance in games against particular kinds of players is measured. When conducting college scouting or foreign drafts, for instance, they are looking at a huge variety of analytics fields.

SportVU, an advanced analytic player tracking camera system, is revolutionizing the way the National Basketball Association (NBA) scouts and plays the game. It's one thing to recognize a player's talent, but SportVU can assess a player's efficiency and effectiveness throughout the game, not just in the first quarter. It can monitor a player's running distance and the actual amount of energy used throughout the game. Twenty years ago, it was impossible to quantify player efficiency, but with the advent of video tracking systems, it is now possible to do so. NBA sports analytics is undoubtedly developing.

Introduction

A hustle play in the NBA is when a team makes a successful play because of an unexpected deployment of energy. Examples include leaping out of bounds to save a lost ball, diving for a loose ball, or pursuing an offensive player on a fastbreak. Some athletes are reputed to be habitual hustlers. In the 2016–17 season, the NBA began recording, compiling, and disseminating a number of hustle statistics, such as blocked shots, deflections, recovered loose balls, and the like. During a basketball game or practice, there are numerous chances to showcase hustle and effort.

Tracking hustle points further stresses the value of taking charges or stomping the floor in search of a loose ball, something that many coaches attempt to do in some form. However, some coaches may believe that stomping the ground in search of a loose ball is expected, and they are correct. Nevertheless, scoring and grabbing rebounds are also expected, and it is safe to assume that all coaches keep an eye on them. Hustle plays should be tracked in the same way as other numbers. Every time a player moves down the floor, there is a chance for a hustle play, but in the excitement of the game, these opportunities are frequently missed.

The interesting thing about the hustle statistic is that it gives a number/data to something that was commonly tracked using the eye test. The eye test is a way to judge an athlete as they compete within their sport based upon your own observations. The issue with this is that people have cognitive biases and personal perspectives towards sports, which can hinder or overvalue what they are perceiving. These biases and perspectives can be influential in how people rate teams or athletes in the sport.

Typically, prior to the introduction of data analytics and the hustle statistic, in particular, the only resource available was the eye test. This addition of hustle data cuts through the several biases people has and quantifies the hustle effect numerically. Although the data introduces a more factual depiction of a player's hustle performance, statistics are limited as there are intangibles that cannot be properly tracked or contextualized. There are intangibles that statistics cannot measure, which requires a balance between these intangibles and statistics.

To ignore any advanced statistics in favor of eyeballing games while talking about basketball would also be a mistake. The whole purpose of advanced statistics is that they are at work during every minute of activity, even though viewing games is vitally crucial. They are thus more comprehensive and knowledgeable than a single person's limited observations. None of the 82 games played by all 30 teams can be watched live, but statistics can. When used properly, this capability makes them valuable. The main problem with advanced metrics is inaccurate application or interpretation. Even worse than just disregarding numbers, memorization of numbers without context or additional analyses is pointless.

Hustle plays are considered by many experts as the most important statistics that often get overlooked. The objective of this project is to determine which of the hustling plays may have an impact on wins and losses from the data collected by the NBA, before measuring or quantifying that impact.

In this project, we are going to be focusing on the impact of hustle plays in winning NBA games. We will not be analyzing per team basis or who benefits the most, just a general overview of how impactful the hustle statistics can be when it comes to winning NBA games. We will be developing different statistical models and graphs to explore and analyze the different potential effects of hustle plays on team success.

Data

Our data is pulled directly from the NBA website. Our data set is from 2016-2018 NBA seasons (This equates to 3 seasons worth of data). This was pre-COVID-19; the 2019-20 season was cut short, especially for those teams that failed to qualify for in-bubble play once the season resumed, because of the pandemic. The Data set includes these variables:

- Off_loose_balls and def_loose_balls:

  The variables ‘off_loose_balls’ and ‘def_loose_balls’, respectively, offensive, and defensive loose balls recovered–have minimums of 0, thereby suggesting that for at least one season the NBA tracked total loose balls recovered only. A loose ball recovered is just that–the offensive team has lost control, but not necessarily possession, of the ball which is then recovered and controlled by the offense or defense.

- Charges:

  The variable ‘charge’ occurs when an offensive player makes significant contact with a defensive player who has established their position with both feet on the ground and their torso square facing the opponent. An on-ball, block-charge situation occurs when contact is made between an offensive player (who is moving in a particular direction or trying to change directions) and defensive player. The defender is permitted to establish his legal guarding position in the path of the dribbler regardless of his speed and distance. To get into a legal position, the defender needs to establish himself in the path of the offensive player before contact is made, thus “beating him to the spot,” and before he starts his upward shooting motion.

- Wins:

  The variable ‘wins’ is the amount of wins the team finished with in that current season.

- Screen_assists_pts:

  The variable ‘screen_assists_pts’ equals the total points scored per game when one player makes a shot, or a field goal, immediately after a teammate sets a screen by placing his body between his teammate and a defensive player.

- Contested_2pt:

  The variable ‘contested_2pt’ which equals the average number of two-point shot attempts that were closely defended.

- Contested_shots:

  The variable ‘contested_shots’ which equals the average number of total shot attempts: two pointers plus three-pointers–that were closely defended. All shots, or field goal, attempts are worth two or three points, depending on distance from the basket.

- Contested_3pt:

  The variable ‘contested_3pt’ which equals the average number of three-point shot attempts that were closely defended.

- Screen_assists:

  The variable ‘screen_assists’ equals the average number of screens set per game, regardless of what then happens on the floor.

- Deflections:

  The variable ‘deflections’ equals the average number of opponent passes broken up, or deflected, per game.

- The variables team and season were concatenated to create an additional variable called ‘team_season’ where, for instance, the 2016-17 Miami Heat becomes MIA 17.

Methodology

Before we begin developing and implementing our statistical models, we will have to prepare and clean the data. The dependent variable and the chosen predictor variables shouldn’t have any outliers, or exceptionally high or low data points. In order to get the most out of linear regression models, even a small number of outliers can skew the results of any model. To successfully eliminate outliers, values that are inconsistent with other values from the same variable will be recognized and subsequently adjusted. When working with small data sets like our hustle data set, winsorization—the process of modifying extreme values—is an appropriate and legitimate substitute for deleting them.

The Winsorized process “is a method of averaging that initially replaces the smallest and largest values with the observations closest to them. This is done to limit the effect of outliers or abnormal extreme values, or outliers, on the calculation” (According to investopedia.com). To determine these outliers, we will create multiple visual plots to identify which ones we should keep and which ones we should remove. These plots will include a histogram, a scatterplot, and a boxplot. By using ‘winsorization’, we will replace these outliers and remove them by setting a maximum in our dataset. Once we run these functions and apply winsorization, we will move on. To recap, we first identified outliers in our data and then subsequently capped those same values so that they equal the Maximum or Minimum values.

Next, the Shapiro-Wilk test will be performed on each variable, regardless of how normal or not so normal their distributions may appear. Using the shapiro.test () function from base R.

Although there are several normality tests, the Shapiro-Wilk test is unquestionably the most used. A Shapiro-Wilk test's null hypothesis is that the data are regularly distributed. We reject the null hypothesis and conclude that the data are not normal if the p-value, which is the probability that an observed difference could have otherwise occurred by random chance, is less than or equal to 0.05.

Alternatively, when the p-value is greater than 0.05, we will instead come to the conclusion that the null hypothesis should not be rejected. The null hypothesis, abbreviated H0, should always be taken as the default position. This implies that there is nothing unusual or statistically significant in one variable or between two data series. To reject the null hypothesis and adopt the alternate hypothesis, known as H1, we would need acceptable statistical evidence.

Once we verify that each predictor is normally distributed, we will also visually plot these distributions using a density plot to visualize. A density curve is a curve with an area exactly 1 below it that is always on or above the horizontal axis. There are areas to the left and areas to the right when looking at a particular data point. A typical curve is one where the mean and median are equal, and it resembles a symmetric histogram. To recap, we tested our variables for normality to determine which of these to carry forward and which to discard from any further analysis and testing.

Moving on, we will compute correlation coefficients between our response and the remaining predictor variables. The correlation coefficients between the variable wins and the other variables will then be calculated and visualized using a correlation matrix. The correlation coefficient is represented by a number between -1 and 1. If a pair of variables' correlation coefficients is equal to or nearly equal to 1, we can infer that there is a positive association between them; if the coefficient is equal to or nearly equal to -1, however, we can infer that there is a negative association; and if they are close to 0, there is no significant association at all.

Here, our goal is to determine which variables in our linear regression model perform as predictors and evaluate which ones exhibit high significance. When working with large data sets, this exercise is especially pertinent because it makes much more sense to look deeper into the data and find high-potential predictors than it does to include every independent variable in a model, regardless of whether or not it adds any value. This is a mainly an exploratory step and help visualize the linear relationship between hustle and wins statistics. We will run correlation coefficients, plot them using ggplot2, and then create a correlation coefficient matrix across all variables. We will begin by simplifying the plotting wins (response variable) and deflections (one of the predictor variables). Once we do this, we will plot a correlation coefficient matrix with the remaining predictor variables. This will allow us to visualize the data and compute the correlation coefficient.

Through this preliminary work for linear regression, we were able to identify outliers. Once these values have been detected, we address them and correct them to reduce outliers' impact. Then, normality has been checked before deciding which subset of the variables to use; and correlations between prospective predictors and the variable, wins and the other predictors have been examined.

In order to create our linear models using training data and make predictions on test data, we first subset 75% of the hustle observations into a data set called train and the remaining 25% into a data set called test. Instead, if we developed and made predictions using 100% of the data, we ran the risk of overfitting our models, which means they would essentially memorize the data and might not be as responsive to fresh data.

The next section of dplyr code first extracts, or filters, every fourth observation from hustle and permanently casts the outcomes into a new object named test. As a result, the test data has 23 rows, or around 25% of the 90 observations in hustle, and the hustle row count is equal to 23.

Our first model, fit1, regresses wins against the variables screen_assists_pts, deflections, loose_balls, contested_2pt, and contested_shots. These variables were selected as predictors based on the correlation coefficients we just computed. We will use the tidy function to retrieve the appropriate statistics from the multilinear regression model. We will use the glance function to check for adjusted r^2, p-value, and AIC for fit1. Once again, multicollinearity is the condition when two or more independent variables in a regression model have a high degree of correlation. To check for multicollinearity, we invoke the vif () method from the car package. It's unlikely that fit1 contains multicollinearity, based on the correlation tests we performed before, but testing for it is still recommended to avoid overfitting.

We will now fit a second multiple regression model where the predictors: screen assist pts, deflections, and loose balls remain in play but the variables contested 2pt and contested shots are omitted because only a subset of the fit1 predictors has a statistically-significant impact on wins.

Thus, fit2, the name of our second regression, is just fit1 with less data. We will repeat similar steps as above. We will use the tidy function to retrieve the appropriate statistics from the multilinear regression model. We will use the glance function to check for adjusted r^2, p-value, and AIC for fit2. We will also check the VIF to avoid multicollinearity.

Next, we will move onto predictive modeling. We will use a predictive model to predict the number of wins. We will use a function with a 95% confidence range that predicts regular season victories. A range of values less than and greater than the predicted value for the response variable that we can be 95% confident contains the actual value for the response variable is known as the confidence interval, or CI.

Once we obtain these results, we will create an actual wins vs. predicted wins plot using ggplot2. This plot will consist of actual wins, predicted wins, and the difference between the two. Once we finish graphing, we will begin moving towards the final steps of the project. We will create a regression tree. Regression trees, also known as decision trees, are simple to create, evaluate, and explain. At a very basic level, regression trees stratify or segment the data into multiple regions of the predictor. Our model will contain the five predictors from our original multiple linear regression; additionally, we'll leverage the previous 75% split from the hustle data set called train as our data source.

Results

Our methodology consists of a detailed, step-by-step explanation of what is covered in the project. Now, we will go over our findings. We begin by looking at our results when we clean the data. To clean the data, we created three different plots to accurately visualize our outliers using the predictor variable deflections. Referring to plot1, we can see there seem to be about 2 outliers. To remove them, we will apply winsorization. This includes modifying any values in the variable deflections greater than 17.8 to instead equal 17.8. We will continue this process by creating boxplots with the remaining predictors and making these changes. The notable changes made were in contested2_pts, changing the values greater than 48.5 to 48.5, changing the contested_shots values that were greater than 69.3 to 69.3; and changing the contested_shots values that were less than 57.4 to 57.4.

Once we finish cleaning the data, we will move onto checking the normality assumption of the predictors. In doing so, we will use the Shapiro-Wilk function. The predictors will have a normal distribution, and based on the Shapiro-Wilk test results, where the p-value is significantly above the 0.05 threshold for significance, is normally distributed. When running all the predictors, we found that all the predictors except charges were normally distributed. The variable charges were not normally distributed based on the Shapiro-Wilk tests and drawing a line in the sand where the p-value is above or below the 0.05 threshold for significance. We also confirm this by visualizing the normality using density plots. Moving forward, we will remove charges from our linear models. (I know that in class, Dr. Niu told me that this assumption is not enough to remove charges from the dataset, but for the sake of consistency between the report and PowerPoint, I will continue without the charges predictor).

Next, we will begin to utilize the correlation coefficients. First, we will calculate and plot a correlation coefficient between the deflection's predictor and wins. Our results find that there is a 0.24 correlation coefficient, which is relatively weak. To continue looking at the rest of the predictors and their correlation coefficient with wins, we will create a correlation matrix. When looking at the correlation coefficient matrix, we find that none of the predictors has a strong correlation, one way or another, with wins. In descending order in comparison to wins, deflections have the highest value at 0.24, contested_2pts at 0.185, screen_assists_pts at 0.170, loose_balls and contested_shots at 0.130, screen_assists at 0.122, and contested_3pts at -0.070.

Now, we are moving onto linear regression. Once we split our data into train and test data, we run a linear regression model containing wins as the response, we have screen_assists_pts, deflections, loose_balls, contested_2pts, and contested_shots as our predictors. When analyzing our results, we use the functions, summary, tidy, glance, and vif. When looking at our predictors, we see that contested2_pts and contested_shots are not significant, as their p-values are not below the alpha = 0.05 significance threshold. We also have an adjusted r^2 value of 0.137, a p-value of 0.0150, and an AIC of 518. When we check the VIFs to make sure there is no multicollinearity, none of our predictors have a value above 5, which shows multicollinearity is not an issue in our model.

Since we could not establish a high significance for the hustle statistics, we will create a reduced model called fit2, removing the hustle statistics that have a lesser effect on winning. To specify, these were our predictors: contested2_pts and contested_shots which were not statistically significant. Now, we run a linear regression model called fit2 that contains wins as the response. We have screen_assists_pts, deflections, and loose_balls as our predictors. When analyzing our results, we use the functions, summary, tidy, glance, and vif. When analyzing our results, we see that all our predictors are statistically significant, as their p-values are below the alpha = 0.05 significance threshold. We also have an adjusted r^2 value of 0.156, a p-value of 0.00334, and an AIC of 514. When we check the VIFs to make sure there is no multicollinearity, none of our predictors have a value above 5, which shows multicollinearity is not an issue in our model.

Next, we analyze our predictive modeling. Three arguments are passed to the predict () function: the model and the data source are required, while the confidence interval defaults to 95%. The results are cast into an object called fit2_pred, where fit equals the predicted number of regular season wins, lwr represents the low end of our confidence interval, and upr represents the high end of our confidence interval. We will use fit2_pred and test vertically and then call the mutate () function from dplyr to create a new variable called wins_dif. The result equals 9.94, thereby suggesting that our second regression performs worse on the testing data than it did against the training data. We will then plot a frequency distribution graph to visualize the differences.

When looking at the results, we got better or more accurate results when actual regular season victories equal 41 or more, as opposed to when clubs won fewer regular season games within an 82-game schedule. Then, we run a second ggplot2 object, a line chart, comparing actual wins against predicted wins, with the shaded areas immediately above and below the predicted values representing the upper and lower confidence intervals.

Our objective was to find which hustling statistics would have a statistically meaningful impact on wins and to measure that impact. This was a far more modest project, as we had a limited number of predictors and only three seasons' worth of data. We've found that explaining 16% of the variance in regular season wins with points off screens, deflections, and recovered loose balls is by no means negligible. And to be completely honest, we've established an idea of when it makes the most sense for players to give maximum hustle/effort, as well as when and where there isn't a fair reward.

Finally, we will move onto our regression tree. We create a regression tree using five predictors. The predictors from our initial multiple linear regression will be included in our model, and we'll also use the earlier 75% split from the hustle data set known as train as our data source. Now we will analyze the regression tree.

- Teams that average more than 26.05 points off screens per game can be expected to win 50 or 51 regular season games.
- Alternatively, teams that average fewer than 26.05 points off screens per game can be expected to win anywhere between 27 and 51 regular season games, depending on other variables and other splits.

- Teams that average fewer than 26.05 points off screens and fewer than 12.85 deflections per game can be expected to win somewhere between 27 and 37 regular season games.
- Teams that average fewer than 26.05 points off screens but more than 12.85 deflections per game can be expected to win somewhere between 31 and 51 regular season games.

Discussion

Because the hustle data set only contains 90 rows, we chose winsorization as our method of data cleansing rather than outlier removal. To achieve the best results, linear regressions also requires that the independent and dependent variables have a normal distribution, (which is why non-normal variables are frequently transformed to make them normal). The absence of outliers in the hustle data set does not, however, imply that our variables will now have normal, or Gaussian, distributions.

Based on the Shapiro-Wilk tests and drawing a line in the sand where the p-value is above or below the 0.05 threshold for significance, it turns out that only the variable charges have a non-normal distribution. The variables contested 2pt and screen assists pts both have p-values barely above 0.05, indicating that their respective distributions are almost non-normal. We are using a 0.05 p-value as a strict cutoff, so we did not include the variable charges in our linear modeling.

In fact, none of the remaining hustle variables have correlation coefficients with wins that are comparable to or as significant as the correlation coefficient between deflections and wins. None of the remaining hustle variables have a strong correlation, either way, with wins.

When looking at linear regression, we can see that our second model, fit2, is a better fit than fit1. To compare the two, all our fit2 predictors were statistically significant. The adjusted r^2 for fit2 is 0.16 compared to fit1 which has an adjusted r^2 of 0.14. (When a response variable is regressed against several predictors, as in fit1, R2 always increases as more predictors are added to the model, regardless of how significant or not so significant they are; for this reason, the Adjusted R2 statistic is a more suitable measure). Fit 2 has a lower AIC, which means the model is stronger. Also, the average difference between actual and fitted wins is slightly greater in fit2 (8.43) vs fit1 (8.27).

As a result, our regression tree generates findings that are closely related to our multiple regressions and offers extra information through a set of if-else rules that linear models are unable to deliver. According to both model types evaluated here, the predictors: screen assists pts, deflections, and loose balls are more important than contested 2-point and contested shots. Our main objective was to determine which of these hustle measures has a greater impact on victories than other comparable data, even if neither model can predict wins with a great deal of precision.

We've decided that some so-called hustle plays merit a 100% effort, while others do not, such as setting screens on offense, which creates limitless shot opportunities, deflecting passes on defense, which thwarts the opponent's offense, and grabbing loose balls while on offense or defense.

Therefore, the same three variables—the same three hustle statistics—were identified by both our linear regression and our regression tree as having the greatest impact on victories.

Overall, they offered unique perspectives. Based on a data set with three years of data, our reduced linear model estimates that points off screens, pass deflections, and loose balls recovered account for roughly 16% of the variance in regular season wins. On the other hand, our regression tree, which was based on several if-else rules, produced several predicted wins.

To look for more accurate results, a random forest model was considered and applied, but the result did not significantly change the predictive modeling accuracy. We attempted to implement the predictors from fit1 and the predictors from the entire dataset. When applying the random forest, we were getting some insignificant results. We were able to conclude that although random forests are generally considered to be a stronger model than regression, there are some limitations that come with random forests. A limitation of random forest is the lack of data provided. Since only three seasons of data were applied, this could have affected results. Generally, random forests do not produce good results when the data is sparse. Another possible limitation was the number of predictors, since we had tried the five predictors from fit1 and the eight total predictors in the data, this could have impacted the results as well. Generally, linear regression needs less data to achieve better results.

Conclusion

The NBA started providing this public tracking data with the implication that this could lead to a competitive advantage. We were able to successfully identify three hustle statistics—points off screens, pass deflections, and loose balls recovered—that together account for 16% of the variance in regular season win totals over the three seasons tested, even though our linear regressions did not effectively explain or predict wins. As a result, we learned where athletes should give their all and where they may, if necessary, rest. Our regression tree determined that the same three variables were more important than the other hustle statistics in our data set, and through a series of if-else rules, it predicted the number of regular-season wins.

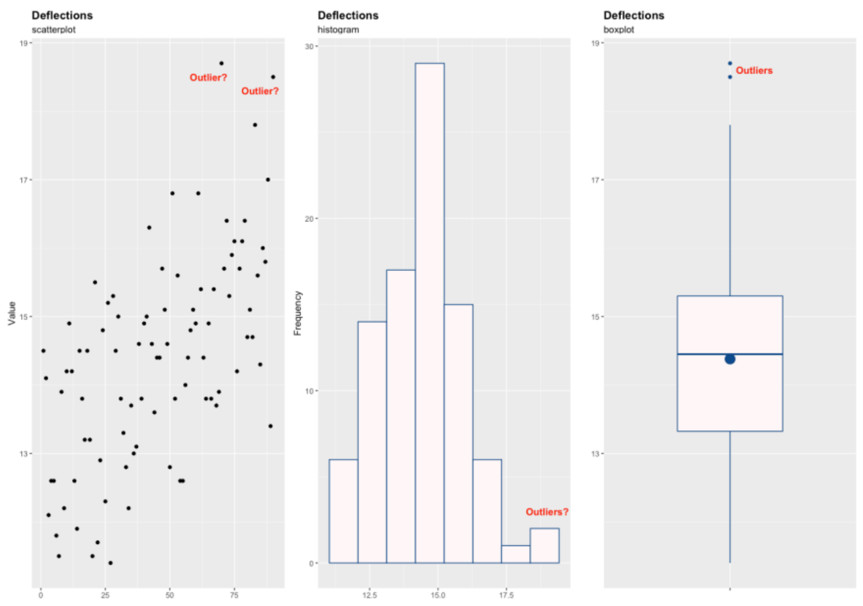
Linear regression analysis might not accurately explain or predict wins, but it did provide useful insight when analyzing the data. The exploratory analysis enabled us to identify which hustle statistics are the most significant and which should be discarded. It seems that one of the biggest limitations faced in this project was the lack of a large, lomgitudinal dataset of hustle statistics. Hustle statistics first started being tracked in the 2016–2017 season. Due to the pandemic, we recorded three seasons of data, which have limited our analysis.

This project was an exploratory analysis regarding the direct relationship between wins and hustle data. The recommendation for the next step is to use a wider data set that includes additional variables like shots made, shots attempted, 2 points or 3 points shots made, turnover margins, etc. The addition of variables that are potentially equipped for predicting wins should improve accuracy and predictability. The addition of more data can lead to more advanced models being used, better accuracy with the variance of wins coming from our regression analysis, and more insight regarding the importance of the hustle statistics.

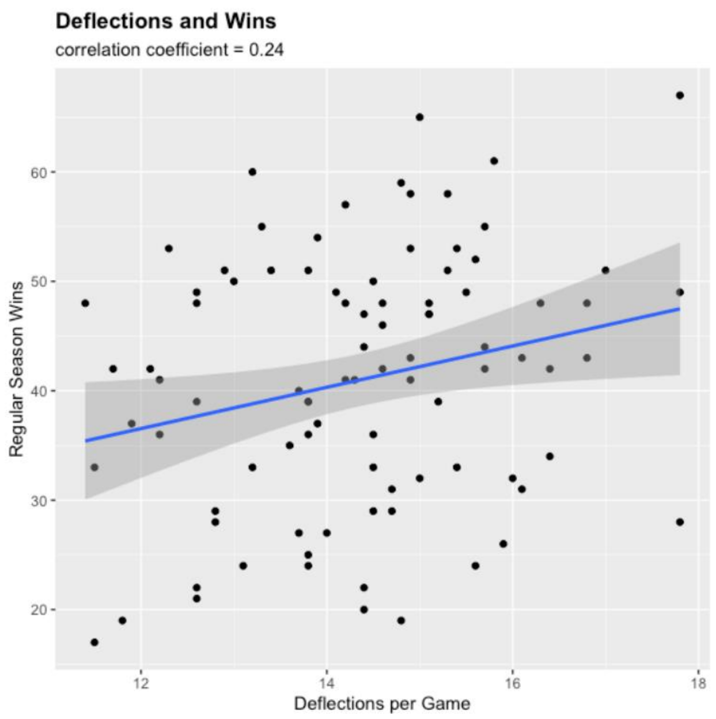
This analysis could provide athletes with the ability to know when to maximize effort regarding certain plays and go beyond the eye test. In the heat of the moment, there is no changing human reaction/decision-making regarding some of the 50-50 hustle plays. We believe that allocating effort to certain types of hustle plays can help these teams create a competitive advantage by applying this effort consistently, thus showing the impact of hustle plays in winning NBA games.

Appendix: Containing important plots

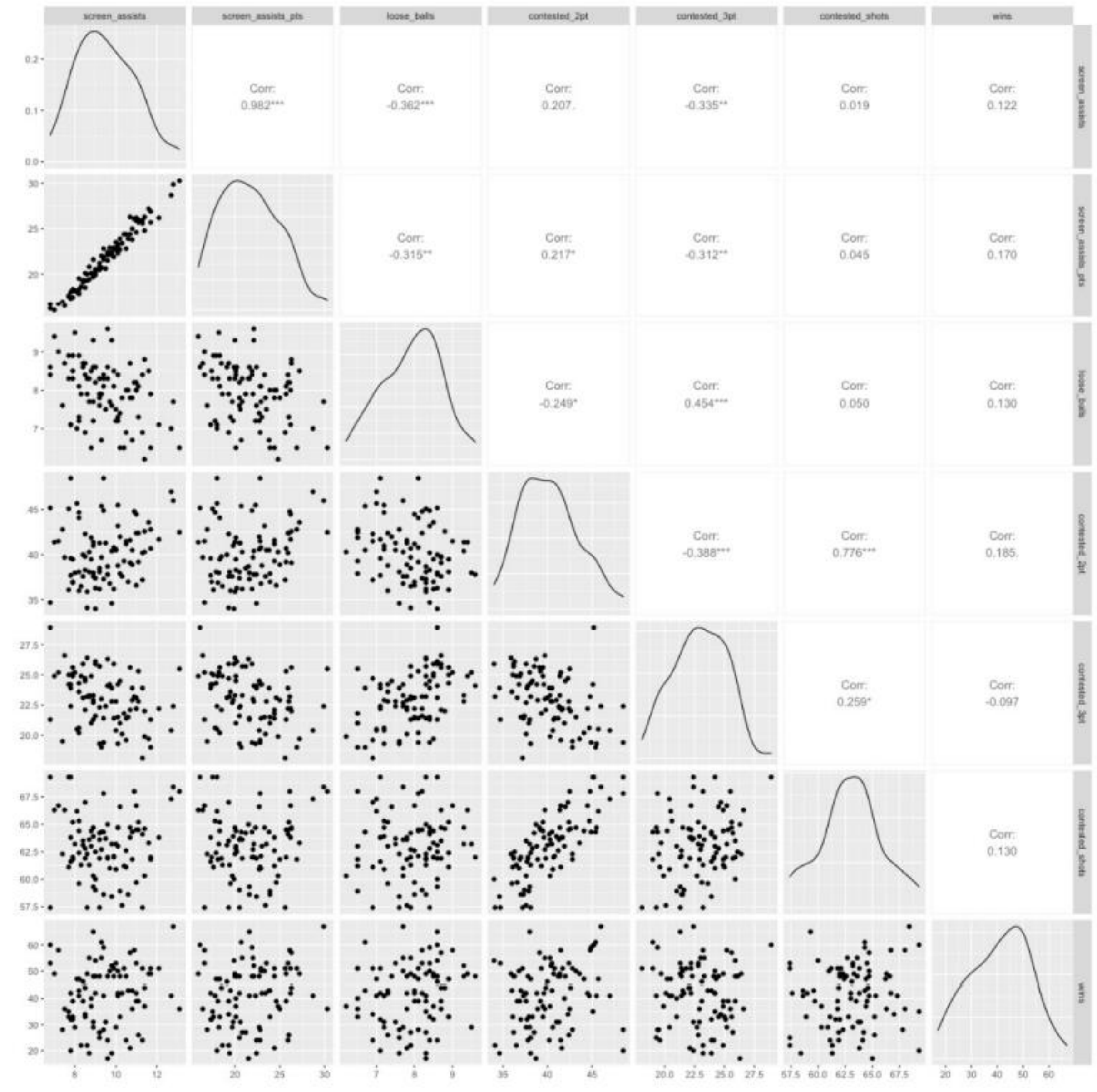
Plot 1: Deflections outlier



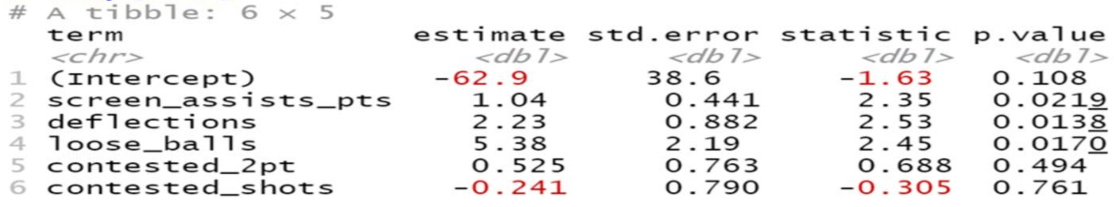
Deflections and wins correlation coefficient



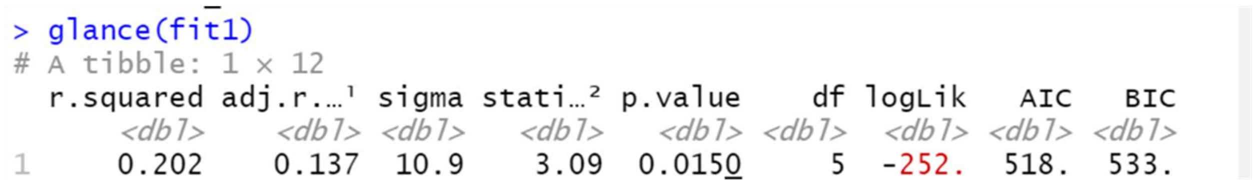
Correlation coefficient matrix, wins as response



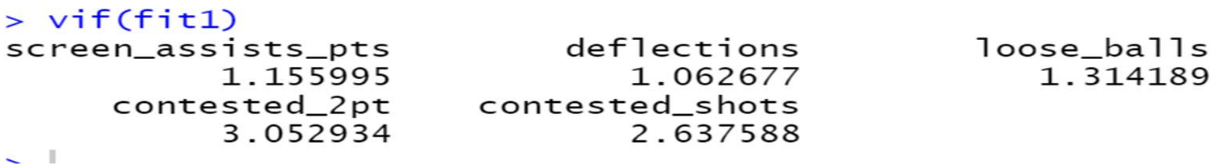
Fit1 results tidy:



```
> tidy(fit1)
# A tibble: 6 x 5
  term                estimate std.error statistic p.value
  <chr>                  <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)          -62.9      38.6      -1.63   0.108
2 screen_assists_pts     1.04      0.441     2.35   0.0219
3 deflections            2.23      0.882     2.53   0.0138
4 loose_balls            5.38      2.19      2.45   0.0170
5 contested_2pt          0.525     0.763     0.688  0.494
6 contested_shots       -0.241     0.790    -0.305  0.761
```

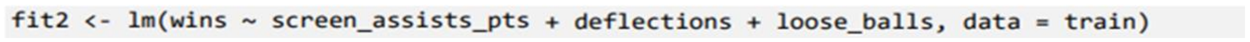


```
> glance(fit1)
# A tibble: 1 × 12
  r.squared adj.r.…¹ sigma stati…² p.value    df logLik   AIC   BIC
      <dbl>    <dbl> <dbl>   <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.202    0.137  10.9    3.09  0.0150     5  -252.  518.  533.
```

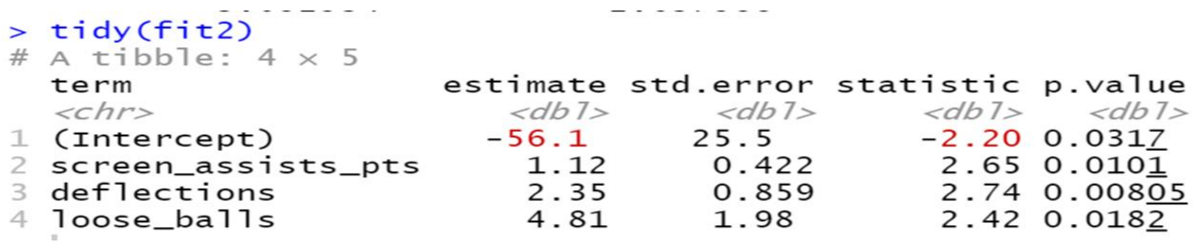


```
> vif(fit1)
screen_assists_pts         deflections         loose_balls
          1.155995            1.062677            1.314189
     contested_2pt     contested_shots
          3.052934            2.637588
```

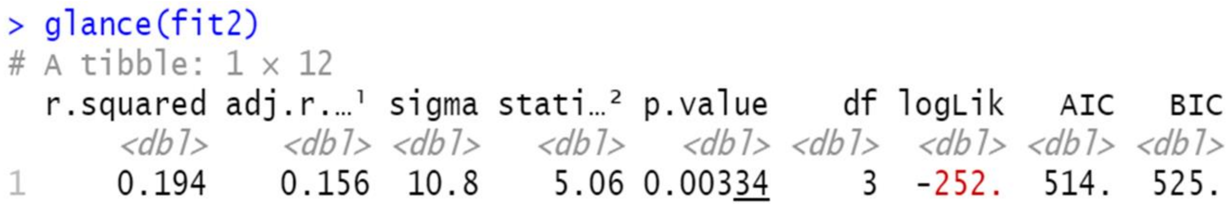
Fit2:



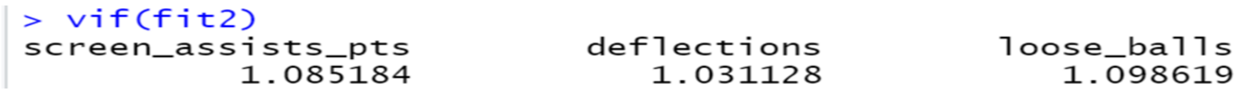
```
fit2 <- lm(wins ~ screen_assists_pts + deflections + loose_balls, data = train)
```



```
> tidy(fit2)
# A tibble: 4 × 5
  term               estimate std.error statistic p.value
  <chr>                 <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)          -56.1     25.5       -2.20 0.0317
2 screen_assists_pts     1.12     0.422      2.65 0.0101
3 deflections            2.35     0.859      2.74 0.00805
4 loose_balls            4.81     1.98       2.42 0.0182
```

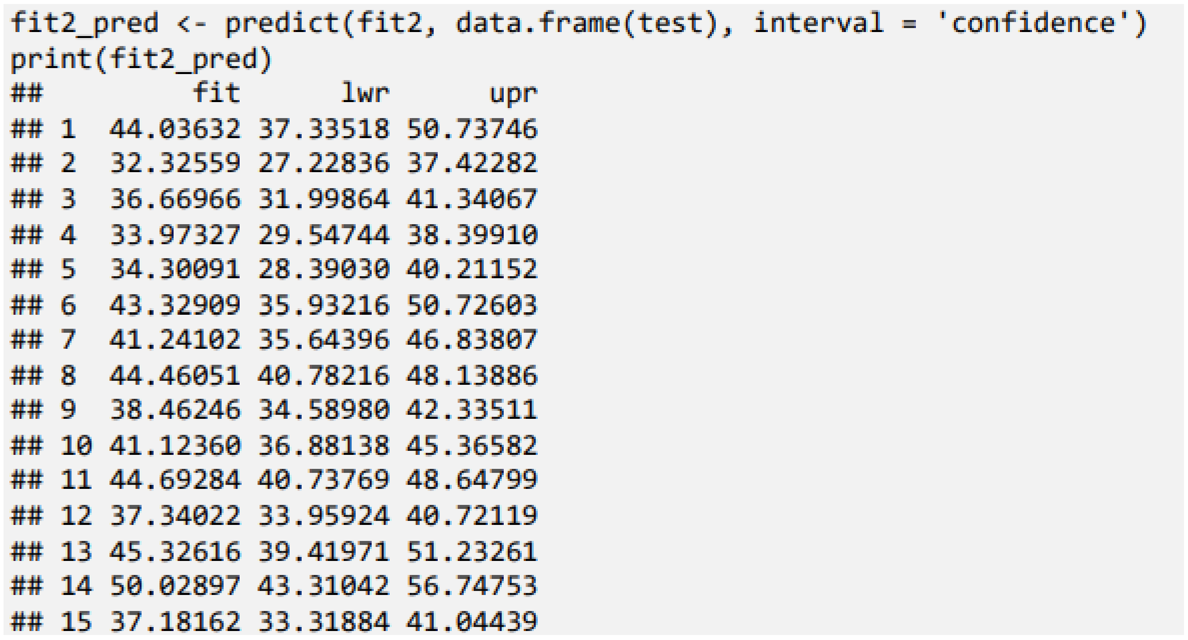


```
> glance(fit2)
# A tibble: 1 × 12
  r.squared adj.r.…¹ sigma stati…² p.value    df logLik   AIC   BIC
      <dbl>    <dbl> <dbl>   <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.194    0.156  10.8    5.06 0.00334     3  -252.  514.  525.
```



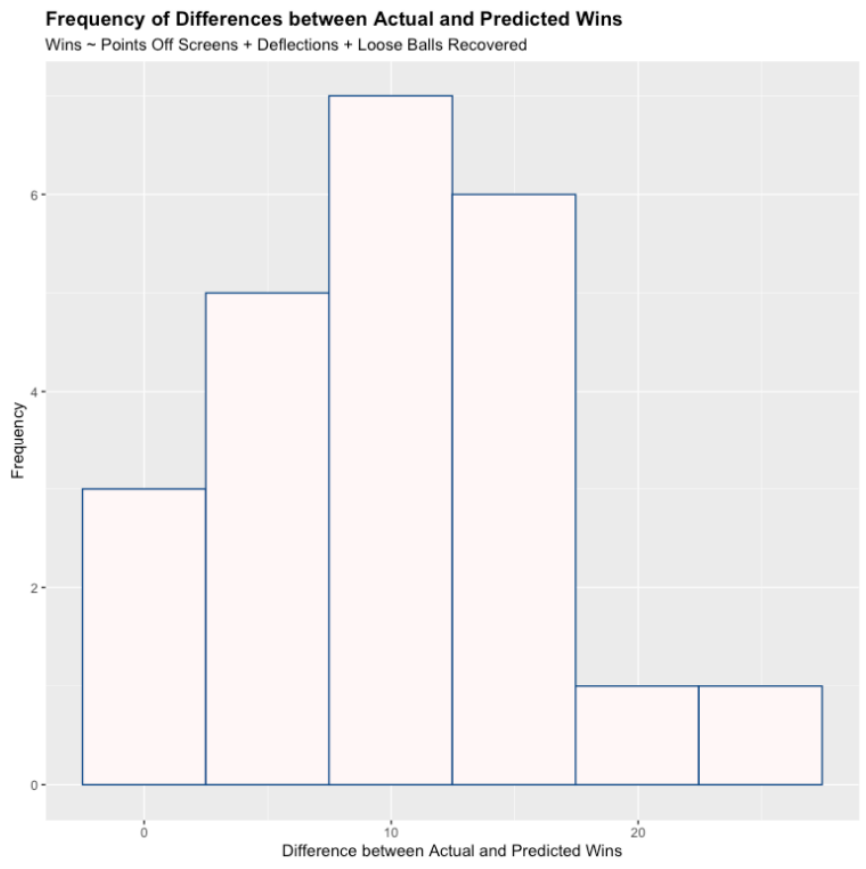
```
> vif(fit2)
screen_assists_pts         deflections         loose_balls
          1.085184            1.031128            1.098619
```

Fit2 confidence interval 95%

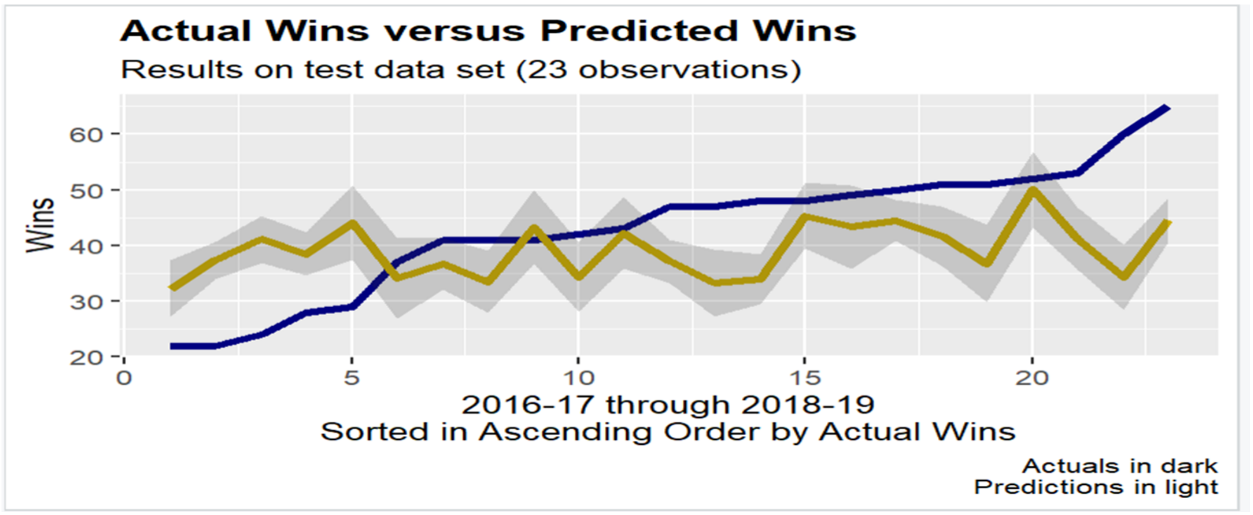


```
fit2_pred <- predict(fit2, data.frame(test), interval = 'confidence')
print(fit2_pred)
##          fit      lwr      upr
## 1  44.03632 37.33518 50.73746
## 2  32.32559 27.22836 37.42282
## 3  36.66966 31.99864 41.34067
## 4  33.97327 29.54744 38.39910
## 5  34.30091 28.39030 40.21152
## 6  43.32909 35.93216 50.72603
## 7  41.24102 35.64396 46.83807
## 8  44.46051 40.78216 48.13886
## 9  38.46246 34.58980 42.33511
## 10 41.12360 36.88138 45.36582
## 11 44.69284 40.73769 48.64799
## 12 37.34022 33.95924 40.72119
## 13 45.32616 39.41971 51.23261
## 14 50.02897 43.31042 56.74753
## 15 37.18162 33.31884 41.04439
```

Frequency of differences between actual and predicted wins



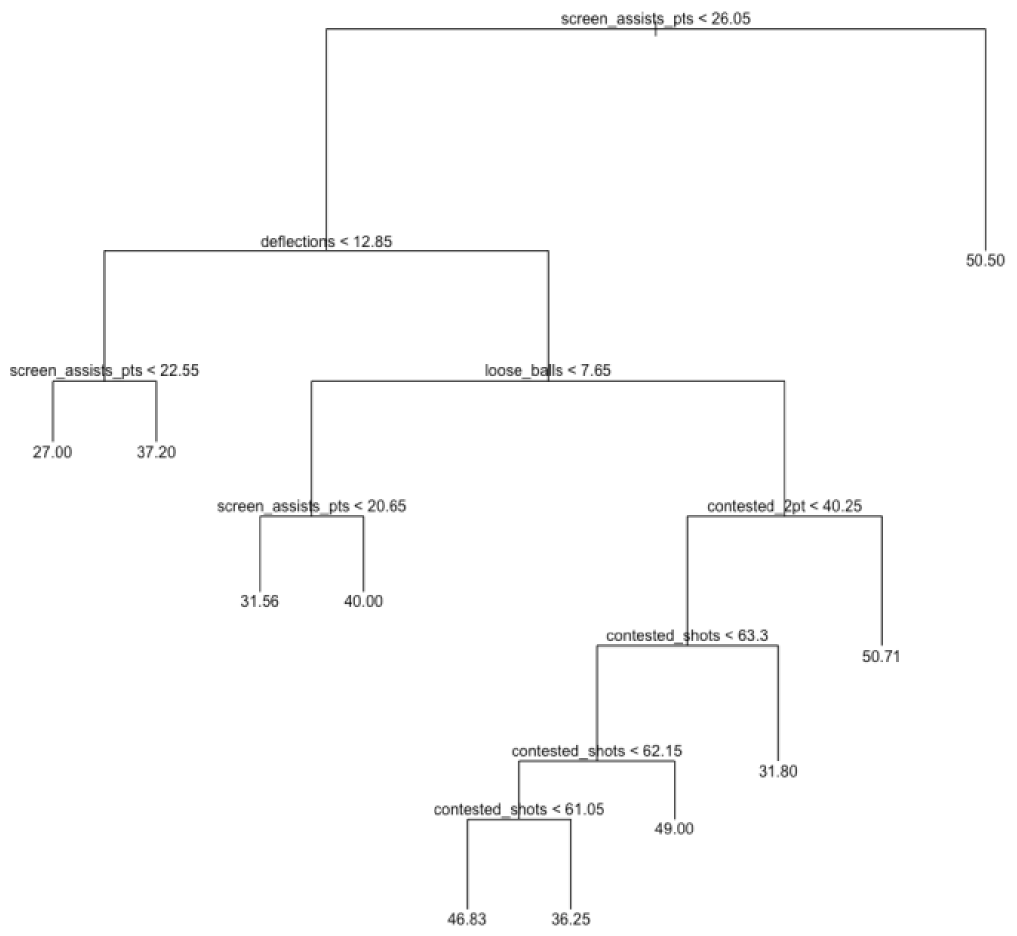
Actual vs predicted wins



Regression tree formula

```
fit3 <- tree(formula = wins ~ screen_assists_pts + deflections + loose_balls +
       contested_2pt + contested_shots, data = train)
```

Regression tree



Some code not all code included for length purposes

```
#plot outliers

#outliers

sp1 <- qplot(seq_along(hustle$deflections), hustle$deflections) +

  labs(title = "Deflections", subtitle = "scatterplot", x = "", y = "Value") +

  theme(plot.title = element_text(face = "bold")) +

  annotate("text", x = 65, y = 18.5, label = "Outlier?", color = "red",

           size = 4, fontface = "bold") +

  annotate("text", x = 85, y = 18.3, label = "Outlier?", color = "red",

           size = 4, fontface = "bold")

hist1 <- ggplot(hustle, aes(x = deflections)) +

  geom_histogram(fill = "snow1", color = "dodgerblue4", bins = 8) +

  labs(title ="Deflections", subtitle = "histogram", x = "", y = "Frequency") +

  theme(plot.title = element_text(face = "bold")) +

  annotate("text", x = 18.75, y = 3, label = " Outliers?", color = "red",

           size = 4, fontface = "bold")

bp1 <- ggplot(hustle, aes(x = "", y = deflections)) +

  labs(title = "Deflections", subtitle = "boxplot", x = "", y = "") +

  geom_boxplot(color = "dodgerblue4", fill = "snow1", width = 0.5) +

  stat_summary(fun = mean, geom = "point", shape = 20, size = 8,

               color = "dodgerblue4", fill = "dodgerblue4") +

  annotate("text", x = "", y = 18.6, label = " Outliers",

           color = "red", size = 4, fontface = "bold") +
  theme(plot.title = element_text(face = "bold")) #Checking normality

shapiro.test(hustle$deflections)

# deflections correlations

cor(hustle$deflections, hustle$wins)

cor1 <- ggplot(hustle, aes(x = deflections, y = wins)) + geom_point() +

  labs(title = "Deflections and Wins", subtitle = "correlation coefficient = 0.24",

       x = "Deflections per Game", y = "Regular Season Wins") +

  geom_smooth(method = lm) +

  theme(plot.title = element_text(face = "bold"))

print(cor1)

#correlation matrix

hustle2 <- select(hustle,-c(1:3, 6, 8))

ggpairs(hustle2)

cor(hustle2)

hustle %>%

  filter(row_number() %% 4 == 1) -> test

train <- anti_join(hustle, test)

dim(train)

dim(test)

fit1 <- lm(wins ~ screen_assists_pts + deflections + loose_balls + contested_2pt +

             contested_shots, data = train)

tidy(fit1)

hustle %>%
  filter(team_season == "MIA 17") %>%
  select(wins, screen_assists_pts, deflections, loose_balls, contested_2pt,
         contested_shots)
print(round(wins))
augment(fit1) -> fit1_tbl
head(fit1_tbl)
fit1_tbl %>%
  mutate(wins_dif = abs(wins - .fitted)) -> fit1_tbl
mean(fit1_tbl$wins_dif)
glance(fit1)
vif(fit1)
par(mfrow = c(2, 2))
plot(fit1)
fit2 <- lm(wins ~ screen_assists_pts + deflections + loose_balls, data = train)
tidy(fit2)
augment(fit2) -> fit_tbl2
print(fit_tbl2)
fit_tbl2 %>%
  mutate(wins_dif = abs(wins - .fitted)) -> fit_tbl2
mean(fit_tbl2$wins_dif)
glance(fit2)
vif(fit2)
par(mfrow = c(2,2))
```

```
plot(fit2)

fit2_pred <- predict(fit2, data.frame(test), interval = 'confidence')

print(fit2_pred)

select(test, wins) -> test

cbind(fit2_pred, test) %>%

  mutate(wins_dif = abs(wins - fit)) -> fit_tbl_pred

mean(fit_tbl_pred$wins_dif)

#actual vs predicted plot

p1 <- ggplot(fit_tbl_pred, aes(x = wins_dif)) +

  geom_histogram(fill = "snow1", color = "dodgerblue4", bins = 6) +

  labs(title = "Frequency of Differences between Actual and Predicted Wins",

       subtitle = "Wins ~ Points Off Screens + Deflections + Loose Balls Recovered",

       x = "Difference between Actual and Predicted Wins", y = "Frequency") +

  theme(plot.title = element_text(face = "bold"))

print(p1)

p2 <- ggplot(fit_tbl_pred, aes(x = row_num, y = wins, group = 1)) +

  geom_line(aes(y = wins), color = "navy", size = 1.5) +

  geom_line(aes(y = fit), color = "gold3", size = 1.5) +

  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2) +

  labs(title = "Actual Wins versus Predicted Wins",

       subtitle = "Results on test data set (23 observations)",

       x = "2016-17 through 2018-19\nSorted in Ascending Order by Actual Wins", y =

"Wins",
       caption = "Actuals in dark\nPredictions in light") +
  theme(plot.title = element_text(face = "bold"))
print(p2)
#regression tree
fit3 <- tree(formula = wins ~ screen_assists_pts + deflections + loose_balls +
contested_2pt + contested_shots, data = train)
summary(fit3)
plot(fit3)
text(fit3)
```