# Baseball Data Mining

BaseBall Statistics via Big Data

Jorge Moreno
University of Colorado Boulder
CSCI 4502
Jomo2834@colorado.edu

Stefano DelPiccolo
University of Colorado Boulder
CSCI 4502
Stde2425@colorado.edu

Paris Dinh
University of Colorado Boulder
CSCI 4502
Padi1849@colorado.edu

# ABSTRACT/INTRODUCTION

Baseball is America's National pastime. A game that millions of people watch each year during the spring and summer. Stats for baseball have been recorded for over 100 years. Using large datasets, we would like to see if there is any possible way to evaluate players and teams based on historical and current data. We then would like to evaluate what are the key features to making a good but cheap team as well as making a good baseball batter.

Using this information, we would like to show trends of when teams/players have come to stardom/success, what makes team/players lose this edge, and the factors that relate to these success/failures.

Some possible outcomes from this data mining project would include an upper hand when placing bets on teams (We don't bet but if we did…), finding the next potential start/rising team, and better rankings for individuals and teams.

And finally, If time allows, we would like to evaluate the dataset to see if we can find more patterns using a variety of other tools including clustering data, using neural networks, and much more

# 1) Related Work

If we do a quick google search for "baseball data mining", there are tons of projects and write ups that people have done concerning this topic. "Data Mining and its Application to Baseball Stats" written by Devin Eudy is an example of this. They used a K-means Algorithm to analyze the data and spit out information regarding players. (1)

Another related work comes "Analyzing Baseball Statistics Using Data Mining" written by Brad Evermen used data to try to predict the performance of teams in the playoffs. Using some simple algorithms "Base on balls percentage, Batting Average, etc" and visual representations, they are able to do so. (2)

For a more sophisticated work, lets view "Predicting Major League Baseball Championship Winners through Data Mining" written by Brandon Tolbert and Theodore Trafalis. They use machine learning algorithms with custom weights to grab information and visualize them (3)

We would like to accommodate all these (except for Machine learning algorithms since we don't have a lot of time) to not only analyze teams, but also analyze individual players. We then would like to visualize these findings in order to show what makes a good and bad team/player.

# 2) Proposed Work

### *2.1) Baseball Reference*

Baseball Reference [5] is a website that contains stats of every player for every year. It also collects data by team, league, position. This database has stats for over a hundred years including stats on Babe Ruth. Although as you go more and more back the data becomes less reliable. An example of the available stats for a pitcher for a given year or for their entire career would be:

- Year
- Age
- Team
- League
- Wins
- Loses
- W/L
- ERA
- Games
- Games Started
- Games Finished
- Home Runs
- Hits

and many more. Through the baseball library you are able to pull all this information from baseball reference.

### *2.2) Statcast Data*

Starting in 2014 baseball park installed the statcast data [6] system in all stadiums. This makes the capture of tons of stats very easy and very reliable. This system tracks every pitch of every game of baseball for the last few years. So with each team playing 162 games a year, this leads to a very large dataset. Some of the things that are tracked by stats cast are the following.

- index,pitch_type
- game_date, release_speed
- release_pos_x
- release_pos_z, player_name
- batter, pitcher
- events, description
- spin_dir
- spin_rate_deprecated
- break_angle_deprecated
- break_length_deprecated

### *2.3) What are the subtasks?*

- Find Correlation between the data and see if this causes causation in a players/teams performance.
    - Correlation vs Causation
    - Chi-Square Testing
- Use pruning of Data to get the most reliable data
- Make our own Stat that will rank Pitchers
- Regression Testing on Data
- Use Entropy to find the best data sets to evaluate a player/team
- Try to apply a regression test and remove data that does not fall into an alpha level of .05 / confidence level of 95%

### *2.4) Is the work enough for your group size?*

We believe that both data sets have enough data to make it challenging to find answers to our questions. We will have to evaluate a lot of data per player and team, and since teams play 100+ games a year, there is bound to be a lot of misleading data.

# 3) Evaluation

- We will base our metrics on 3 groups. Individual players, Teams, and by positions. This way we can be as specific or as broad as we want. This will also give us clear leaders in each to see who is the best and also who need some work to be a competitive team or a competitive player. Our metric will be the comparisons between these groups. It will be used to choose one over another.

- In terms of accuracy and error it can be hard to evaluate success. Success could be being an above average player or team in a certain stats. Or it could be the best or even the worst player. Also since we will be pulling all data from the pybaseball library our accuracy is only as good as the library accuracy. Latency may be the only issue with the project with so much data coming in we may have issues storing and using the data that the library provides.

- With so many statistics already in existence in baseball it would be interesting to be able to compare what is already out there to what we are able to find. If our findings comes up with a completely unknown player it may be cause for speculation. If it comes up with one of the accepted best players we know it may be worthwhile finding.

# 4) Milestones

- Week of Oct 14th Library setup and data acquisition
- Week of Oct 21st Data Cleaning/ Reduction
- Week Nov 3rd Algorithm Creation
- Week Nov 11th Partial Data Testing
- Week Nov 18tth Full Data Testing
- Week Nov 25th Data Visualizations
- Week Dec 2nd Project Completed and Turned In

# 5) Dataset Interpretation

- AB
    - The number of times a player comes up to bat during the season
- WAR
    - Wins above replacement
- TBB
    - The total number of bases a player is responsible for over a given season.

Lahman baseball library functions

- BWAR
    - Returns a list of Batting WAR for ever player.
- Salaries()
    - Returns a list of salaries for every player.

- Batting()
    - Returns a list of batting data for all players.
- Statcast()
    - Get data for every pitch from set a set of date ranges.

We decided to use one year as out baseline due to the amount of data. With data being on every pitch over a 162 game season to start to add up very quickly. We decided to use the year 2016 even though there was more recent years is because of the salary data available. 2016 is the last year available for salaries, and since our data heavily relied on using salary as a mark of a good player we wanted the most current data available.

# 6) The Making of Our Own Stat

Wins Above Replacement (WAR) is an attempt by the sabermetric baseball community to summarize a player's total contributions to their team in one statistic. You should always use more than one metric at a time when evaluating players, but WAR is all-inclusive and provides a useful reference point for comparing players. WAR offers an estimate to answer the question, "If this player got injured and their team had to replace them with a freely available minor leaguer or a AAAA player from their bench, how much value would the team be losing?" This value is expressed in a wins format, so we could say that Player X is worth +6.3 wins to their team while Player Y is only worth +3.5 wins, which means it is highly likely that Player X has been more valuable than Player Y. WAR currently is the baseline for evaluating players. It puts all the most important stats together into the baseline stat.

We as a team disagreed with what makes a good player. WAR only mesures solo stats. We believe that baseball is a team game so the baseline stat should be a team player. The stat we came up with the ultimate team player stat TBB.

TBB is the proposed new statistic that is responsible for counting the total number of bases a player is responsible for across the whole season.From TBB you could easily calculate my other proposed statistic Bases Per At Bat (BPAB) Which as suggested would be TBB/AB. This statistic would be a true team player stat as getting and moving people on bases is the most important part of baseball. Currently if a batter is consistently moving people to let's say third bases but not batting them in there is no way to evaluate that. Only the player who bats the person in would get credit, but the player that batted them from first to third bases deserves just as much credit if not more than the person that batted them from just third to home. There is no true statistic that takes in only bases moved. So I created one. Theses statistics would be most closely related to Runs Batted In (RBI), in which essinactly count the movement of other players on the bases. The main difference being that I count every base and not just the bases that scores the run. A correlation test could be dividing TBB by 4 to see how closely these stats compare. Since I am also counting bases that the batter himself gets on it could also be highly related to On Base Percentage (OBP). If a batter has a high OBP then his BPAB would also be higher because they are directly correlated.

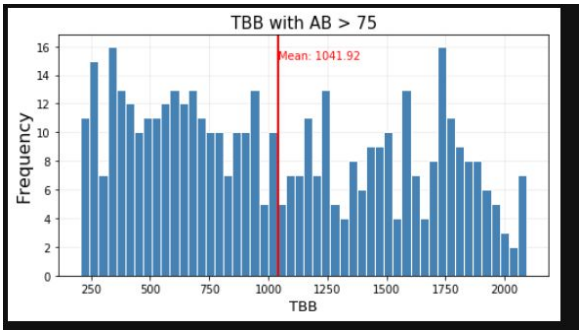
BPAP: Since TBB is highly dependant on the number of at the player receive during the season. The more AB they have, the more opportunities they have to get bases. Due to this TBB is highly un-normal while BPAB is high normalized.

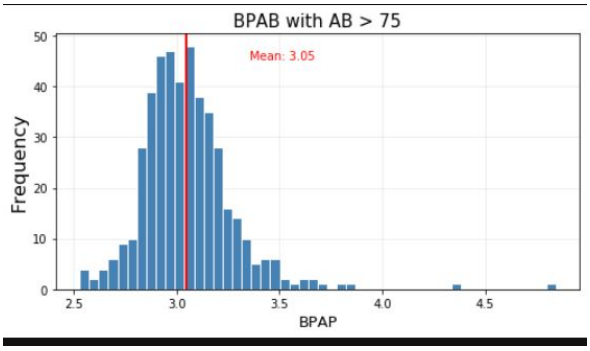
WAR = (Batting Runs + Base Running Runs + Fielding Runs + Positional Adjustment + League Adjustment +Replacement Runs) / (Runs Per Win)

TBB :The way this statistic was calculated is by using the statcast 2016 season data. The first thing I did was drop all pitches in which no event happens there should be no bases moved so all of these were dropped. I then look at every at bat and create 4 new columns. The first 3 columns are the base state after the at bat. This is done by look at the base state of the next at bat and this will be the change that the previous at bat caused. The last column would be the change in the score. You have to look at the change of score to account for the runners who scored. These runners would no longer be on base and need to still be accounted for in the total bases moved. Each at bat will then go through various if else statements to determine the situation that happened. For example, the first if statement is a check to see if anyone is on base. This is the case where it is easiest to count the bases from this at bat. Its either 1, 2, 3 or 4 for a home run. There is only more and more cases from here on out. The rest of the cases are dependant on how many runs were scored during the at bat. This is also where the statistic cannot be 100% accurate. Looking at a case where two runs or score. There is a variety of ways this can happen. The list is as follows. Third base and second base can score, Third base and first base can score,3 base and the batter can score, Second base and first base can score, second base and batter can score, and finally first base and the batter can score. I had to make certain assumptions because using the statcast data there is no for sure way to see who actually scored the run, you can only see that runs were scored. For example if bases are loaded and 2 people score, I have to assume that third and second scored. There might be very few cases where in this situation somehow third and first could score and second could get out, but due to to the limitations of statcast I would have no idea this happened. I made the best out of what I had. I store all the data in a dictionary with the key being the batter ID this way after each bat .The way I calculate bases could mean that even if 2 runs are scored in two different situations the amount of bases could be vastly different. For example if two people score the bases would be very different if third and second scored vs batter and first bases score. In the first situation is responsible 3 bases (also depends on were first base and the batter got to) vs the other situation is responsible for 7 bases. In opinion this

is a fair way to evaluate bases. It is a much harder task to bat in people from hitting and 1st base you would almost certainly have to hit a homerun could cause this event. In the other situation all it would take is a long pop fly to achieve this result. That's why I think it is fair to be worth double the bases. It is more worth it to hit in runners farther from home. If done in one at bat there is less risk of getting out then having to move them over multiple at bats therefore the extra bases is justified.

BPAB = TBB / AB get the normalized value.




# 7) Regression Model and its Application

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. The general model/Formula.

A linear regression line has an equation of the form:

***Y = a + bX***,

Where ***X*** is the explanatory variable and ***Y*** is the dependent variable. The slope of the line is ***b***, and ***a*** is the intercept (the value of ***y*** when ***x*** = 0).

For the first test figure(1) we used WAR Vs. Salary Since WAR is considered the end all be all stat, the regression should show that as WAR increases so does salary.

For the next test figure(2) we compared OBP and Salary. Since OBP and TBB should be highly correlated if TBB is to be reliable stat salary should increase with OBP.

For the next test figure(3) we compared WAR and TBB. As WAR is the baseline stat if better players have a higher WAR then a higher WAR should mean a higher TBB if our stat is to be effective.

For the next test figure(4) we compare BPAB and salary. If better players get paid more. We would like to see as BPAB increase that salary also increases.

For the next test figure(5) we compare TBB and Salary.Same reason as figure(4)

For the last test figure(6) we compare BPAB and WAR for the same reason as figure(3).

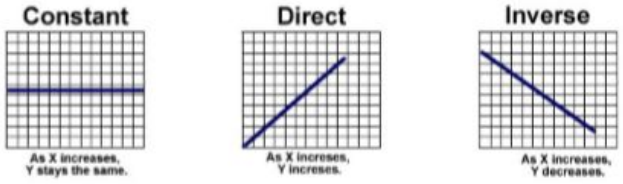
We used the OLS to get these results. We ran all our variables against both WAR and Salary. Code and the results are in figures 11 and 12.

# 8) Chi-Square Testing

## *8.1 Correlation Vs Causation*

One of the first data mining techniques we wanted to try on our datasets was to test correlation on different attributes of batter data. Using Pearson Correlation Coefficient, the following hold true:

$-1 \leq r \leq 1$ where r stands for the correlation. The following Image also demonstrates what a 0, 1 and -1 correlation look like.




A $r > 0$ means that the x and y have a direct increase relation with one another. They have a more positive linear relation as r approaches 1.

A $r < 0$ means that the x and y have an inverse opposite relation to one another. They have a negative linear relationship as they approach -1.

Finally, an r = 0 means that they are constant and both datasets don't affect one another. There is no linear relationship between one another.

Later in this section, we will discuss our correlation findings for the data we compared.

With all that said, Correlation does NOT imply causation. It does however help us identify possible key attributes that might be helpful for later processing. An example of how correlation does imply causation is below. [9]


## 8.2 Chi-Square

Another data technique we wanted to use was the chi-square test. The Chi-Square statistic is commonly used to test relationships between categorical variables.The following image shows how to calculate the chi-square test statistic ( $\chi^2$ ) and the expected value associated to the observed value.

Note: the expected value is :

$\frac{row(i) * Column(j)}{N}$ where N = total amount

(Image from CU Boulder CSCI 4502/5502 Lecture 04)

✦ $X^2$ (chi-square) test (categorical data)

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{N}$$

- r = number of rows
- c = number of columns
- i= row index
- j = column index
- $O_{ij}$ = observed value at a certain row/column index
- $e_{ij}$ = expected value at a certain row/column index

A chi-square test usually has a null hypothesis that states all observed data is independent of one another. This is the null hypothesis we used throughout our test steps.

In order to proceed with our testing, we also had to calculate the Degree of Freedom for each test Case. This was done by the following formula:

$$DF = (Number\ of\ Rows - 1) * (Number\ of\ Columns - 1)$$

After having our chi-square test statistic value and our DF for each step, we used [8] to calculate the p-value associated to the data.

Discussion on our p-value and the Confidence level we choice will later on in this section.

## 8.3 Breaking Up the Data

For our data, we decided to use the following attributes to test our regression models, talked about above, and correlation and chi square tests. (Note the corresponding figures in our Appendix are in parenthesis)

- WAR vs Salary (Figure 1)
- OBP vs Salary (Figure 2)
- TBB vs WAR (Figure 3)
- BPAB vs Salary (Figure 4)
- TBB vs Salary (Figure 5)
- BPAB vs WAR (Figure 6)

For a chi-square test, we need to get data into a categorical form. An example is shown below. (Picture from CU Boulder csci 4502/5502 Lecture 04 )

| | play chess | not play chess | total |
|---|---|---|---|
| like fiction | 250 (90) | 200 (360) | 450 |
| not like fiction | 50 (210) | 1000 (840) | 1050 |
| total | 300 | 1200 | 1500 |

In order to so, we had to come up with a way to split up the x and y values in categories with a set increment. Because we did regression models using a pandas dataframe and used numpy/stats this problem was solved for us. If you look in the figures mentioned above, you will see that our x and y variables where all split into bins that the models produced for us.

An example is figure 1, War vs Salary. For our y axis, salary, we see that it increments in sets of 0.5e7 or 5 mill. As for the x, WAR was split into increments of 2 starting at -2. By counting the number of increments that where present in both x and y axis, we were able to come up with an observed table matrix and an expected matrix table. For a visual representation look at Figure 7 and Figure 8,

Observed and Expected for War vs Salary. The rows where the salary increase each one going up by .5e7 while the columns where the WAR increase each going up by 2. The observed is the amount of times data was found in this section and the expected was gotten from the formula previously talked about.

For more info on clarification on how the process was gotten, please refer to figures 9 as well as our github

### *8.4 Results*

In the end, we got some interesting results testing correlation and chi square values.

To start of, let's see all our data for each process.

- WAR vs Salary (Figure 1)
  - Correlation = 0.18966693
  - Low positive linear relationship
  - P Value = 0.2576
- OBP vs Salary (Figure 2)
  - 0.21880156
  - Low positive linear relationship
  - P Value = 0.2395
- TBB vs WAR (Figure 3)
  - 0.70517746
  - High positive linear relationship
  - P Value = 0
- BPAB vs Salary (Figure 4)
  - 0.06144511
  - Very low positive linear relationship.
  - Almost no relationship
  - P Value = 0.7966
- TBB vs Salary (Figure 5)
  - 0.3837375
  - Medium positive linear relationship
  - P Value 6.492e-7
- BPAB vs WAR (Figure 6)
  - 0.30905767
  - Medium positive linear relationship
  - P Value = 0.0005096

As expected from looking at our graphs, the highest positive linear relationship we see between the attributes we selected is TBB and WAR. This tells us that our best player predictor stat and the one that is being formulated in the dataset are very relatable meaning our data can be trusted.

With that said, a WAR and BPAB where not as close as we wish they would be. As stated above, WAR was the baseline we are using to identify what a good player is. BPAB we assumed would be a true team player stat as getting and moving people on bases is the most important part of the game.

For out Chi-Square tests, we decided to have a confidence interval level of 95% with a significance level of 0.05. In order to reject the null hypothesis with our chi square, which stated the variables were independent, P < significance level.

P < 0.05.

We reject the null hypothesis when it comes down to the following data: TBB vs WAR, TBB vs Salary, and BPAB vs WAR. What this is telling us is that we reject the hypothesis that any of these tests and their variables that they have are independent from each other.

We can conclude from this that TBB and war have a relationship we can trust at a 95 % CI. As well with TBB vs Salary and BPAB vs WAR

# 9) Frequent Itemsets with Apriori

### *9.1 The Apriori Algorithm*

Another data mining technique we applied to player batting data was the Apriori algorithm. In using this technique, we hoped to discover patterns which could give us a way to discern better players from the average joes. The Apriori algorithm is used to find frequently appearing items amongst a set of baskets where each basket represents a different player and each item represents an offensive statistic such as Home Runs (HR), Stolen Bases (SB), etc. In finding frequent itemsets of offensive batting stats, we can determine what the average joe may look like in data and find players who are a step above the others.

### *9.2 The Datasets*

Two datasets were used from pybaseball; batting wins above replacement, WAR and Batting Statcast, BSC data. Using BSC, we can mine frequent patterns amongst players' offensive performance and find where most people are distributed. From this, we can discern current players who are better than others.

WAR data will act as our baseline for comparison as it is commonly used alone as a metric for determining the worth of a player. We wish to use BSC data to see if we are able to differentiate players.

### *9.3 Preprocessing*

The WAR data originally comes in the shape (107 049, 17). This includes multiple years of data. For the sake of testing on a small scale, we limit data to be during 2017. In the interest of testing on complete data, we drop missing values and restrict WAR to the year of 2017 leaving us with a shape of (794, 10). We also dropped some unnecessary attributes that we were not concerned with.

Performing the same preprocessing that WAR got, BSC's clean shape was (957,77). It also got additional set up to

be used effectively with Apriori. Of possible solutions, we decided to map each player's attributes onto a scale that was based on that particular attribute's distribution. The scale is ranked from very low to very high and are designated with acronyms VL, L, H, VH. Those that are very low on the distribution are those who are below the first quartile, inclusive. Those who are ranked low and high are those who fall between the first quartile and the median and the median to the third quartile, respectively. Those above the third quartile are ranked very high. Once mapped, we apply the Apriori algorithm.

### *9.4 Methodology*

An initial guess at a good minimum support and confidence that would yield interesting results from our data was needed. Minimum support is the threshold of an item's appearances over the entire dataset where we'd consider it frequent. Minimum confidence is the ratio threshold of an item's appearances over the support for that item in the data. For minimum support, we think that .28 would suffice under the assumption that players are equally distributed along our scale. As for minimum confidence, we'd like strong associations between attributes and our itemsets where .95 confidence will work for now. Implementation of Apriori yields results that will be discussed later. The important takeaway from these results is what it didn't give us; the people we are interested in are those that do not meet the criteria for run of the mill players. Filtering players based on a super set of frequent attributes that average players have gives us a group of good and average players.

We then compare our curated group of good players against those that are not using our baseline WAR data. Performing a difference of means statistical analysis utilizing a two tailed Z test between our populations with a significance level of 5% confirms a significant difference in the populations.

### *9.5 Apriori Results*

min_supp = 0.28
min_conf = 0.95

The max sized frequent itemset we were able to mine reached a length of 8 attributes. We extracted 19,017 itemsets of 8 from 77 attributes. One such result is as follows:

('L BsR', 'L wSB', 'VL 2B', 'VL 3B', 'VL CS', 'VL HBP', 'VL HR', 'VL HR/FB')

The interpretation elementwise of this itemset is a common scale of a particular stat. For example, the first element is a low Bases Running Runs above average. Other acronyms and their meanings can be referenced at [11]. Attributes found common in large sets suggests some commonality amongst players which are the average players.

Collection of all these common stats into a set of criteria that may indicate an average player is used as a filter among the players yielding two groups. The count of above average players comes to 41 of the 611 players considered on this small scale experiment. For a visual representation refer to figure 10.

We perform a difference of mean WAR hypothesis test on this compiled group against the rest to see if these players are indeed better than all others. This test follows standard calculation of a test statistic under the assumption that the data is normally distributed with mean 0 and standard deviation of 1. A p value was calculated from this and used to decide if the difference in skill between our curated players and others were significant with an alpha of 0.05.

At a glance, the data shows us that

| Good Players | Average players |
|---|---|
| Mean WAR | |
| 2.7059 | 0.7733 |
| Standard Deviation | |
| 2.2921 | 1.6352 |

We calculated a test statistic of

Z = 5.30273

From this, we get a p value of a two tailed Z test

P = 1.1408511699663287e-07 = 0

With a significance level of 0.05, we reject the null hypothesis that there is no significant difference between these two populations in favor of there being a significant difference in WAR of our selected players versus all other players we have data on.

### *9.6 Calculation Methods*

We test these hypotheses:
$H_0 : \mu_{gp} - \mu_{bp} = 0$
$H_1 : \mu_{gp} - \mu_{bp} \neq 0$

$$\text{test statistic} = z = \frac{(\mu_1 - \mu_2) - c}{\sqrt{\frac{\sigma_1^2}{m} + \frac{\sigma_2^2}{n}}} \sim N(0,1)$$

$$\text{p-value} = 2 \cdot \Phi(-|z|)$$

$$\Phi(z) = P(Z \leq z) = \int_{-\infty}^{z} f(x)\, dx$$

# 10) Other Findings

Data mining through all the baseball stats was a harder task than we thought it would be. We assumed since so much data was already present, we could easily grab them and find patterns. In reality, this was a harder obstacle than we thought and took a while. With all that said, there were still a couple of questions that we wanted to see if we could answer using the information and data techniques we learned from above

### *10.1) Does Salary = Good Player?*

Our first question we proposed, was to see if a bigger salary had any correlation to what makes a Good Player. In order to test this, we referred back to our Chi-Squared findings and used the following three variables that ended up passing our chi-squared test and having good correlations : TBB, WAR: BPAB.

We used WAR as our baseline gain since this is the variable that was provided to us pybaseball.

The approach to finding an answer to this question was simple, we would use a simple linear regression model (using the OLS model from statsmodel.api) and having our independent variable equal to Salary and our dependant variable equal to our three variable we were testing.

Figure 13 shows the linear regression model when we had BPAB as our dependent variable. We see that the correlation coefficient was 0.06144 with an r-Squared of 0.004. This means .4% of the variance comes from BPAB itself. All in all, it does not seem that this variable had any relation with salary.

Figure 14 shows the linear regression model when TBB was our independent variable. For this model, we see that the correlation coefficient was 0.384 and had an R-squared value of 0.147. From this, we can say that this these two variables had a positive linear relationship with one another and 14.7% of the variance in salary was caused by TBB.

Finally, figure 15 shows the model with WAR being our dependant variable. For this model, the correlation coefficient was 0.1897 and an R-squared of 0.036. Although it does seem like a positive correlation was forming, the variance doesn't seem to be too good to tell us that correlation in this case caused causation.

In the end, the TBB value that we produced seemed to have the best correlation when it came to salary of a player.

### *10.2)* Does Age = Performance?

Another question we wanted to answer was to see if Age had and correlation to performance. This was tricky because pybaseball does not have a column that has the age for players. It does however, have birth year, month, and day. We were able to create a column of players current age, using current year not accounting month and day. We used a similar approach as the previous example except our independent variable was Age.

For this problem, we decided to evaluate all of the tests into one paragraph. Figures 16,17 and 18 show the results from each test. Right away, you can see that they were not as good as the previous questions. The correlation coefficients we got from each one where in the negative (usually meaning a negative relation) and close to 0 (meaning no relation between the variables) the R-squared values also support this by having low values in each with a max of 2% for one and 0% for the lowest.

With all that said, we believe that the dependant variables that we used for this test where  not the best to use when trying to find the answer to this question. Neither our two stats we created or the WAR stat was able to show any relation between them.

# 11) Results

In the end, we hit most of our goals that we declared at the beginning of our project. We first began by creating our own variable, TBB and BPAB talked about in Section 6.

From here, we went on and created  OLS regression models using TBB, OBP, WAR, and BPAB as our independent variables. Reasons and results for this where talked on Section 7.

Chi-Squared testing was then done in Section 8 to answer our questions concerning correlation and causation, and using the chi-square test on a 95% confidence level to get the top variables from the four we decided to choose.

At the same time, we used the Apriori algorithm to determine frequent itemsets with a pre-defined minimum support and minimum confidence. Results where talked about in Section 9 for this approach.

And finally, we went on and tried to answer two other questions that were brought up during our research on

question 10. This used a combination of our Chi-Square approach and the OLS models to try to answer them.

The only point we were unable to hit on was entropy testing because of time constraints.

Each one of these sections goes in depth on the results and findings.

# REFERENCES


[1] Devin Eudy, Data Mining and Its Application to Baseball Stats, https://www.cs.csustan.edu/~mmartin/teaching/CS4960S15/Eudy_CS4960_paper.pdf

[2] Brad Everman, Analyzing Baseball Statistics Using Data Mining, https://truculent.org/papers/DB%20Paper.pdf

[3] Brandon Tolbert and Theodore Trafalis, Predicting Major League Baseball Championship Winners through Data Mining, https://www.athensjournals.gr/sports/2016-34-1-Tolbert.pdf

[4] MLB Advanced Media, LPI, https://www.mlb.com/

[5] Sports Reference, BaseBall ,, https://www.baseball-reference.com/

[6] Savant,Statcast CSV Documentation, https://baseballsavant.mlb.com/csv-docs

[7] JLDBC, pybaseball GIT, https://github.com/jldbc/pybaseball

[8] MathIsFun, Chi-Square Calculator, https://www.mathsisfun.com/data/chi-square-calculator.html

[9] Bjorn Ostman, Correlation does too imply causation http://pleiotropy.fieldofscience.com/2012/04/correlation-does-too-imply-causation.html

[10] Statistic Solutions Advancement Through Clarity, Using Chi-Square Statistics in Research, https://www.statisticssolutions.com/using-chi-square-statistic-in-research

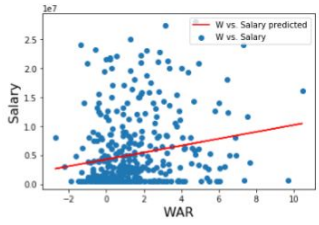
[11] Neil Weinberg, Complete List (Offense), https://library.fangraphs.com/offense/offensive-statistics-list/

[12] Our Git Hub, Baseball Data Mining, https://github.com/Morenju1363/DataMinningProject

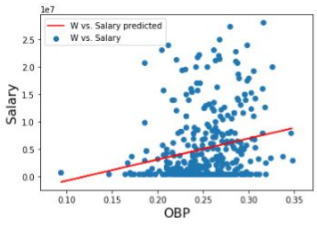

# Honor Code Pledge

“On my honor, as a University of Colorado Boulder student, I have neither given nor received unauthorized assistance.”

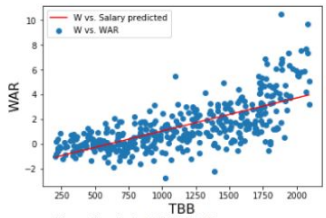
# Appendix



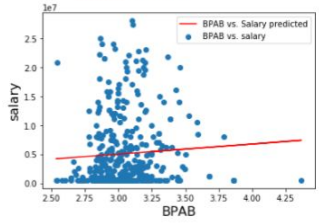

CorrCoef: 0.18966693
Chi[-Square: 53.0625
DF: 35
PValue: 0.2576



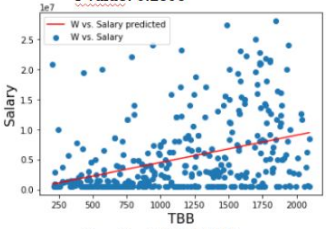

CorrCoef: 0.21880156
Chi-Square: 47.16994
DF: 30
PValue: 0.2395



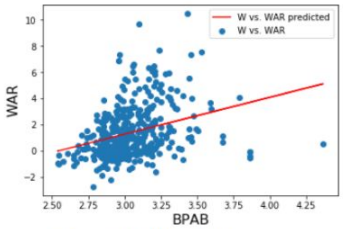

CorrCoef: 0.70517746
Chi-Square: 343.869
DF: 56
PValue: 0




CorrCoef: 0.06144511
Chi-Square: 27.8715
Df: 35
PValue: 0.7986




CorrCoef: 0.3837375
Chi-Square: 99.022
DF = 40
PValue: 6.492e-7




CorrCoef: 0.30905767
Chi-Square: 88.1525
DF: 49
PValue: 0.0005096

*Figures 1,2,3 (top) and 4,5,6 (bottom)*

Figure 7

```
Observed Matrix:
[1, 78, 125, 42, 13, 7, 1, 0]
[1, 9, 31, 11, 9, 1, 0, 0]
[0, 7, 13, 10, 3, 2, 0, 0]
[0, 2, 8, 7, 1, 1, 0, 1]
[0, 4, 6, 4, 1, 1, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 0]
```

*Figure 8*

```
Expected Matrix:
[1.328358208955224, 66.41791044776119, 121.54477611940298, 49.8134328358209, 18.59701492537
3134, 7.970149253731344, 0.664179104477612, 0.664179104477612]
[0.30845771144278605, 15.422885572139304, 28.223880597014926, 11.567164179104477, 4.3184079
60199005, 1.8507462686567164, 0.15422885572139303, 0.15422885572139303]
[0.17412935323383086, 8.706467661691542, 15.932835820895523, 6.529850746268656, 2.437810945
2736316, 1.044776119402985, 0.08706467661691543, 0.08706467661691543]
[0.09950248756218906, 4.975124378109452, 9.104477611940299, 3.7313432835820897, 1.393034825
8706469, 0.5970149253731343, 0.04975124378109453, 0.04975124378109453]
[0.07960199004975124, 3.9800995024875623, 7.2835820895522385, 2.985074626865672, 1.11442786
06965174, 0.47761194029850745, 0.03980099502487562, 0.03980099502487562]
[0.009950248756218905, 0.4975124378109453, 0.9104477611940298, 0.373134328358209, 0.1393034
8258706468, 0.05970149253731343, 0.004975124378109453, 0.004975124378109453]
```

*Figure 9*

Chi Squared = SUM of (Ovserved - Expected)^2 / expected

https://www.mathsisfun.com/data/chi-square-calculator.html To get The Pvalue

```
from math import pow
def chiSquared(observed,expected, N):
    sumProduct = 0
    for row in range(len(observed)):
        for col in range(len(observed[0])):
            if(expected[row][col] == 0):
                sumProduct += 0
            else:
                sumProduct += pow(observed[row][col] - expected[row][col],2) / expected[row][col]
    return sumProduct
```

```
def estimate2(row, col, total):
    #Row * Column / Total
    return((sum(row) * sum(col)) / total)
```

```
def tableManipulation(numberOfRows, numberOfColumns, xIncrement, yIncrement, xStart, yStart, DF, x , y):
    saveXStart = xStart
    #Making new Dataframe with DF x and y columnns
    data = {x : DF[x], y : DF[y]}
    combine = pd.DataFrame(data)
    N = 0
    #Matrix with numberOfRows  x  numberOfColumns
    table = [[0 for x in range(numberOfColumns)] for y in range (numberOfRows)]
    for row in range(numberOfRows):
        for col in range(numberOfColumns):
            table[row][col] =  len(combine.loc[(combine[y] > yStart) &  # Lower Y constraint
                                               (combine[y] <= yStart+yIncrement) & # Upper Y Constraint
                                               (combine[x] > xStart) &  #  Lower X Constraint
                                               (combine[x] <= xStart+ xIncrement)]) #  Upper X Contraint
           # print(table[row][col])
            N +=  table[row][col] #Get total of input data
            xStart += xIncrement # increment column which is Y constraints. Filling table
        xStart = saveXStart #reset X constraint to start at bottom left
        yStart += yIncrement #increment y Constraint to go up one row
    for i in table:
        print (i)

    numpyArray = np.array(table)
    expected = [[0 for x in range(numberOfColumns)] for y in range (numberOfRows)]
    for row in range(numberOfRows):
        for col in range(numberOfColumns):
            expected[row][col] = estimate2(numpyArray[row] , numpyArray[:,col],N) # Estimate Function



    print("Chi Square Value is: ", chiSquared(table,expected,N))
    print("Degrees of Freedom is (rows - 1 * Col-1) : ", (len(table)  -1 )* (len(table[0]) -1))
    print ("Correlation Coefficient: ",np.corrcoef(DF[x],DF[y]))
```

*Figure 10*

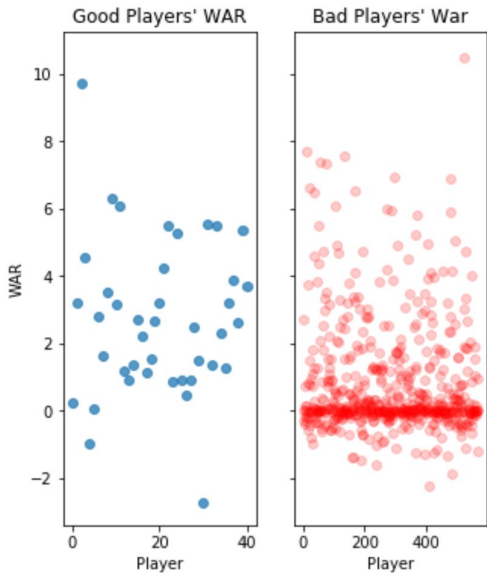
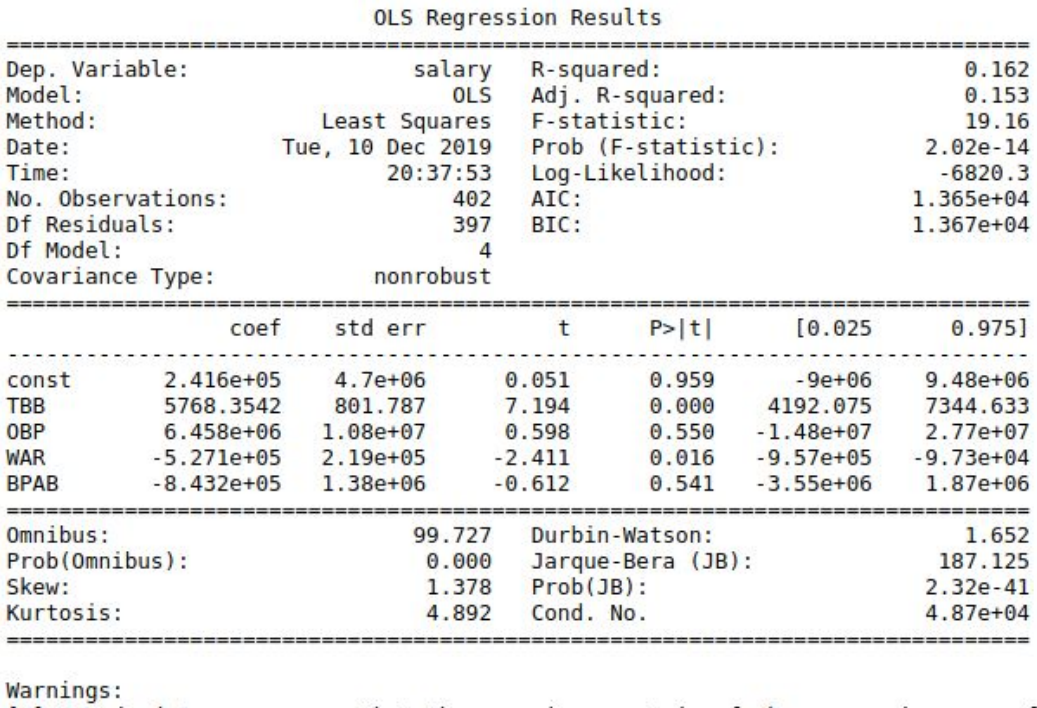


*Figure 11 Salary MLR*



OLS Regression Results

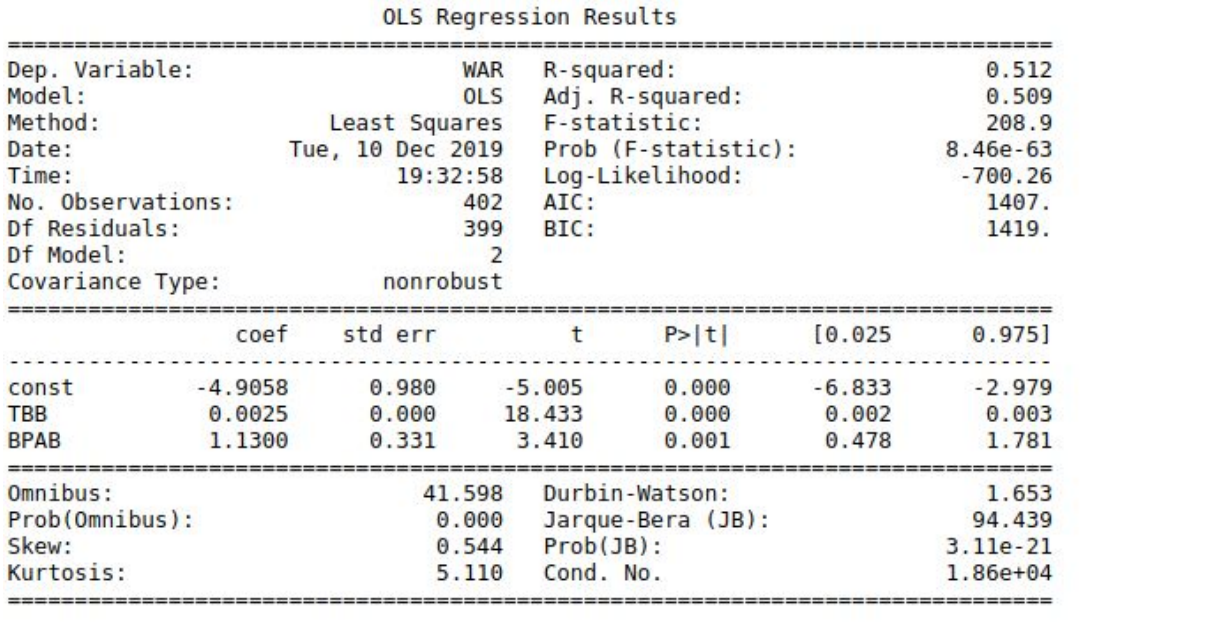
| | | | |
|---|---|---|---|
| Dep. Variable: | salary | R-squared: | 0.162 |
| Model: | OLS | Adj. R-squared: | 0.153 |
| Method: | Least Squares | F-statistic: | 19.16 |
| Date: | Tue, 10 Dec 2019 | Prob (F-statistic): | 2.02e-14 |
| Time: | 20:37:53 | Log-Likelihood: | -6820.3 |
| No. Observations: | 402 | AIC: | 1.365e+04 |
| Df Residuals: | 397 | BIC: | 1.367e+04 |
| Df Model: | 4 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 2.416e+05 | 4.7e+06 | 0.051 | 0.959 | -9e+06 | 9.48e+06 |
| TBB | 5768.3542 | 801.787 | 7.194 | 0.000 | 4192.075 | 7344.633 |
| OBP | 6.458e+06 | 1.08e+07 | 0.598 | 0.550 | -1.48e+07 | 2.77e+07 |
| WAR | -5.271e+05 | 2.19e+05 | -2.411 | 0.016 | -9.57e+05 | -9.73e+04 |
| BPAB | -8.432e+05 | 1.38e+06 | -0.612 | 0.541 | -3.55e+06 | 1.87e+06 |

| | | | |
|---|---|---|---|
| Omnibus: | 99.727 | Durbin-Watson: | 1.652 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 187.125 |
| Skew: | 1.378 | Prob(JB): | 2.32e-41 |
| Kurtosis: | 4.892 | Cond. No. | 4.87e+04 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.87e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

*Figure(12) WAR MLR*



OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | WAR | R-squared: | 0.512 |
| Model: | OLS | Adj. R-squared: | 0.509 |
| Method: | Least Squares | F-statistic: | 208.9 |
| Date: | Tue, 10 Dec 2019 | Prob (F-statistic): | 8.46e-63 |
| Time: | 19:32:58 | Log-Likelihood: | -700.26 |
| No. Observations: | 402 | AIC: | 1407. |
| Df Residuals: | 399 | BIC: | 1419. |
| Df Model: | 2 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | -4.9058 | 0.980 | -5.005 | 0.000 | -6.833 | -2.979 |
| TBB | 0.0025 | 0.000 | 18.433 | 0.000 | 0.002 | 0.003 |
| BPAB | 1.1300 | 0.331 | 3.410 | 0.001 | 0.478 | 1.781 |

| | | | |
|---|---|---|---|
| Omnibus: | 41.598 | Durbin-Watson: | 1.653 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 94.439 |
| Skew: | 0.544 | Prob(JB): | 3.11e-21 |
| Kurtosis: | 5.110 | Cond. No. | 1.86e+04 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.86e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

*Figure 13*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   BPAB   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.516
Date:                Tue, 10 Dec 2019   Prob (F-statistic):              0.219
Time:                        20:42:58   Log-Likelihood:                 44.424
No. Observations:                 402   AIC:                            -84.85
Df Residuals:                     400   BIC:                            -76.86
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.0337      0.014    216.137      0.000       3.006       3.061
salary      2.162e-09   1.76e-09      1.231      0.219   -1.29e-09    5.61e-09
==============================================================================
Omnibus:                      131.163   Durbin-Watson:                   1.277
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              533.487
Skew:                           1.390   Prob(JB):                    1.43e-116
Kurtosis:                       7.911   Cond. No.                     1.04e+07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.04e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
Correlation Coefficient:  [[1.         0.06144511]
 [0.06144511 1.        ]]
```

*Figure 14*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    TBB   R-squared:                       0.147
Model:                            OLS   Adj. R-squared:                  0.145
Method:                 Least Squares   F-statistic:                     69.07
Date:                Tue, 10 Dec 2019   Prob (F-statistic):           1.49e-15
Time:                        20:42:55   Log-Likelihood:                -3054.6
No. Observations:                 402   AIC:                             6113.
Df Residuals:                     400   BIC:                             6121.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        966.2084     31.281     30.888      0.000     904.714    1027.703
salary      3.252e-05   3.91e-06      8.311      0.000    2.48e-05    4.02e-05
==============================================================================
Omnibus:                       16.372   Durbin-Watson:                   0.239
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                8.262
Skew:                           0.130   Prob(JB):                       0.0161
Kurtosis:                       2.348   Cond. No.                     1.04e+07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.04e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
Correlation Coefficient:  [[1.         0.3837375]
 [0.3837375 1.       ]]
```

*Figure 15*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    WAR   R-squared:                       0.036
Model:                            OLS   Adj. R-squared:                  0.034
Method:                 Least Squares   F-statistic:                     14.93
Date:                Tue, 10 Dec 2019   Prob (F-statistic):           0.000130
Time:                        20:42:58   Log-Likelihood:                -836.90
No. Observations:                 402   AIC:                             1678.
Df Residuals:                     400   BIC:                             1686.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0954      0.126      8.714      0.000       0.848       1.343
salary      6.075e-08   1.57e-08      3.863      0.000    2.98e-08    9.17e-08
==============================================================================
Omnibus:                       89.286   Durbin-Watson:                   0.923
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              168.177
Skew:                           1.223   Prob(JB):                     3.03e-37
Kurtosis:                       5.014   Cond. No.                     1.04e+07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.04e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
Correlation Coefficient:  [[1.         0.18966693]
 [0.18966693 1.        ]]
```

*Figure 16*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   BPAB   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.1012
Date:                Tue, 10 Dec 2019   Prob (F-statistic):              0.751
Time:                        21:37:24   Log-Likelihood:                 43.715
No. Observations:                 402   AIC:                            -83.43
Df Residuals:                     400   BIC:                            -75.44
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.0746      0.094     32.559      0.000       2.889       3.260
Age           -0.0009      0.003     -0.318      0.751      -0.007       0.005
==============================================================================
Omnibus:                      125.332   Durbin-Watson:                   1.267
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              483.541
Skew:                           1.342   Prob(JB):                    1.00e-105
Kurtosis:                       7.655   Cond. No.                          285.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Correlation Coefficient:  [[ 1.         -0.01590783]
 [-0.01590783  1.        ]]
```

*Figure 17*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    TBB   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.3464
Date:                Tue, 10 Dec 2019   Prob (F-statistic):              0.556
Time:                        21:37:12   Log-Likelihood:                -3086.5
No. Observations:                 402   AIC:                             6177.
Df Residuals:                     400   BIC:                             6185.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1264.4736    227.393      5.561      0.000     817.438    1711.509
Age           -4.0961      6.959     -0.589      0.556     -17.778       9.586
==============================================================================
Omnibus:                      288.496   Durbin-Watson:                   0.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               25.210
Skew:                           0.009   Prob(JB):                     3.36e-06
Kurtosis:                       1.773   Cond. No.                          285.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Correlation Coefficient:  [[ 1.         -0.02941544]
 [-0.02941544  1.        ]]
```

*Figure 18*

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    WAR   R-squared:                       0.023
Model:                            OLS   Adj. R-squared:                  0.021
Method:                 Least Squares   F-statistic:                     9.431
Date:                Tue, 10 Dec 2019   Prob (F-statistic):            0.00228
Time:                        21:37:26   Log-Likelihood:                -839.58
No. Observations:                 402   AIC:                             1683.
Df Residuals:                     400   BIC:                             1691.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.9970      0.850      4.703      0.000       2.326       5.668
Age           -0.0799      0.026     -3.071      0.002      -0.131      -0.029
==============================================================================
Omnibus:                       83.145   Durbin-Watson:                   0.831
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              145.616
Skew:                           1.185   Prob(JB):                     2.40e-32
Kurtosis:                       4.754   Cond. No.                          285.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Correlation Coefficient:  [[ 1.         -0.15176875]
 [-0.15176875  1.        ]]
```