**SI 601: Final Project**

**Analyzing 2014 Batting Data: Baseball-Reference and Fangraphs**

**Alex Czarnik - 2/24/2016**

**Motivation**

In this report, I will discuss my findings from my research leveraging data from Baseball-Reference.com and Fangraphs.com. Baseball-Reference and Fangraphs both specialize in the collection of baseball data from all leagues (major league baseball, minor league, etc.). When looking at both sites “leaderboards”, the two two sites display a collection of dashboard stats which focus on different metrics. Baseball-Reference’s display focuses more on traditional statistics like “Homeruns”, “Batting Average”, and “Walks”, giving fans an idea of the basic metrics for a given player. Fangraphs’ display, however, contained more analytic statistics known as “sabermetric” statistics. These stats provide fans a better understanding of some of the nuances of a player’s performance besides the obvious statistics that can be easily observed during a game.

The reason I chose this topic was similar to why I decided to come to the School of Information. With an established passion for data analytics, I decided I wanted to pursue a career in the sports industry performing analytics for baseball teams, relatable to the popular featured film “Moneyball”. I decided to take this project as an opportunity to work some popular baseball data, leading me to the two popular websites Baseball-Reference and Fangraphs.

The biggest goal from this project was to merge the data and construct a large dataset containing a large index of data for any major league batter that qualified for over 100 “Plate Appearances”. For context, a “Plate Appearance” is defined as an instance that a batter has been given the chance to bat during an official major league game. Regardless of outcome, Homerun, Strikeout, etc., the batter is credited with a Plate Appearance which becomes important when calculating stats like Batting Average or On-Base Percentage.

My hope is that by creating this merged dataset I can find correlations between the statistics provided by the differing sites. One of the more popular sabermetric statistics is the “WAR” statistic, also known as Wins-Above Replacement. In recent years, this statistic has become a staple for fans to evaluate players for their contributions batting, pitching, and playing defense, in an effort to find a more unanimous way to judge a player by more than his Batting Average. I hope that my joined dataset can show how stats from both data sources attribute to this statistic WAR. My goal, however, is not to prove WAR’s significance in accurately evaluating players, but rather research whether a collection of generic and sabermetric stats can show correlation to a popular ranking stat.

**Data Sources**

Originally, in my initial proposal I anticipated leveraging three different datasets, two csv files and a sql file. My research led me to believe that the sql file would be similar to the database files we were manipulating in class. However, after further evaluation, the sql file was different and incompatible with sqlite that led to my questioning of the dataset for this project. Continuing my evaluation, I realized that the data contained within the file was identical to the Baseball-Reference dataset. After this realization I decided to continue with only the Baseball-Reference and Fangraphs dataset, that I will go into more description.

**Baseball-Reference**

Baseball-Reference has become a very good database of information when looking for any data regarding the sport. The dataset I am working with specifically is in regards to the 2014 Major League Baseball season. To download this dataset, one can go to the link I have provided, scroll down to the table “Player Standard Batting”, and click the “Export” button above the table. This will produce a csv file of roughly 1,600 rows and 30 columns. Of these 30 columns, I used 28 of them for my project which I have listed below the link to the dataset. Along with the 28 columns, I cut my dataset down to roughly 450 rows based on my qualifier of 100 plate appearances that I explained in the introduction. The reason I did this was to eliminate plate appearances that were credited to pitchers and players that typically would not bat. When it comes to batting evaluation, it is unnecessary to account for pitchers because they ultimately do not carry enough significance to a team with their batting abilities. The dataset and my qualifier also handles players that may have been traded during the season that may have statistics logged for two separate teams. A player with this condition appears multiple times in the original dataset, but only once in my joint set with all statistics from the full season. This dataset required minimal effort accessing the data, but it provided a great dataset that I will most likely use in future projects.

Baseball-Reference Link: <http://www.baseball-reference.com/leagues/MLB/2014-standard-batting.shtml>

Baseball-Reference Important Columns:

[Name Age Tm Lg G PA AB R H 2B 3B HR RBI SB CS BB SO BA OBP SLG OPS OPS+ TB GDP HBP SH SF IBB]

**Fangraphs**

Fangraphs has been one of the websites leading the charge of the sabermetric era in major league baseball. The advanced analytics, statistics and articles by fangraphs have become widely respected and accepted by the baseball community. Like the Baseball-Reference dataset, the Fangraphs dataset focuses on baseball players that qualified with 100 plate appearances during the 2014 Major League Baseball season. The dataset started with 22 columns, but I only extracted columns that were unique to this dataset compared to the Baseball-Reference set. This left me with 10 important columns, listed at the bottom of this section, from this set that were merged together with the other dataset. This dataset also began with 1000 rows, with a qualifier that the player needs to have started the season in the major league, as opposed to the Baseball-Reference dataset which included any player that hit during the season. I reduced this 1000 rows to match the 450 rows that was produced from filtering the Baseball-Reference data. The only qualifier I applied to this dataset was in the join based on player name, any players in the Fangraphs data that was not in my processed Baseball-Reference dataset was excluded.

Fangraphs Link:

<http://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=8&season=2014&month=0&season1=2014&ind=0&team=&rost=&age=&filter=&players=>

Fangraphs Important Columns:

[BB% K% ISO BABIP wOBA wRC+ BsR Off Def WAR]

**Data Manipulation Methods**

As mentioned in the Data Sources section, I applied a few different qualifiers to my data to filter out and manipulate my data. However, in this section I will go into more detail as to what went into each function and dataset to completely explain the data manipulation process that went into this project. I will try to emulate reading each line of code to properly understand the program.

**Baseball-Reference Process**

I began my program by reading the Baseball-Reference data with a CSV reader and extracting the header to produce keys for a dictionary. While doing this I had to account for any punctuation such as quotation marks, extracting only the words within the string. Once I had a list of column names, that I named keys, I then began processing the rows within the dataset. As I did with the keys, I accounted for any punctuation within a string, but allowing periods for numeric values. Once I validated that the strings were void of unnecessary punctuation, I compared the row’s ‘Name’ attribute to a list of names created later in the program. The reason for this comparison is to ensure to skip duplicates that may appear later in the dataset. After I validate that the row in question has a unique name that is not in my list of names, I account for any repeating headers that appear in the data. For the Baseball-Reference dataset specifically, the header appears multiple times so a user does not have to constantly check the top row, but for my use, I skipped these rows when adding the data to my variables. This then brings in the qualifier mentioned earlier about requiring 100 plate appearances to be entered into the variables. I accomplish this filter with a simple if statement that checks the row’s ‘PA’ attribute.

After the row makes it through all of these conditions, I add the row’s ‘Name’ attribute to the name list and begin filling in my dictionary for that player. I accomplish this by looping through the list of keys that was generated earlier, and adding the row’s attributes for each key to the dictionary. I apply regular expression filters for floats, ints and strings, and convert the values into its appropriate data type. The final processing I apply is removing the ‘Rk’ attribute from my dictionaries and key list. This attribute was not important to my analysis because the Baseball-Reference ranking was alphabetical order and beared no indication to a player’s performance. Once I removed this variable, I appended the player dictionary to a list and looped to the next row. This function then returned the list of keys, list of names, and list of dictionaries. Below I provide a brief workflow with pseudo code.

Baseball-Reference Workflow:

1. Extract Dataset Keys
   1. Remove any unnecessary punctuation (quotation marks)
2. Extract Data
   1. For each row, create a dictionary
   2. Remove any unnecessary punctuation (asterisks, pound symbols)
   3. Validate row name is unique
   4. Validate row as a player row, not a header row
   5. Filter row based on condition: Plate Appearances > 100
   6. Append row name to list of names
   7. For each key, add row attribute to created player dictionary
   8. Convert values to appropriate data type based on regular expression
   9. Remove ‘Rk’ attribute from key list and dictionaries
   10. Append player dictionary to list of dictionaries
3. Return List of Keys, List of Names, and List of Dictionaries

The challenges I mainly encountered with this dataset was verifying that the data extract was actually data that I wanted. Filtering out header rows, removing punctuation and casting the appropriate data type to each attribute all came from roadblocks that occurred while coding. These challenges, however produced a better product overall, producing a better dataset that would then be merged and analyzed.

**Fangraphs Process**

Once I processed the Baseball-Reference data, I returned the end product, which became a parameter for my fangraphs function. I then created variables (Keys, Names, Data), based on the parameter provided. I began reading the Fangraphs csv with a csv reader similar to the Baseball-Reference dataset. I extract all the keys from the header row, and appended only unique keys that were not provided by Baseball-Reference. These keys were stated in the Data Source section of this report. Due to some problems with this dataset, I had to account for anything that was not in ‘UTF-8’. Once I had all the keys from both datasets, I began reading the data rows. As I did with the previous set, I filtered out any players that did not appear on the list of names, and this reduced the data to about 475 rows. Once I verified that the ‘Plate Appearances’ attribute was also above 100, I ran a for loop going through the list of dictionaries provided by the Baseball-Reference processing. If the Fangraph row’s ‘name’ matched a name in the list of dictionaries, I then began adding the Fangraph attributes to the player’s dictionary. Due to the fact that I was using a full list of keys, I had to start my loop at the 28th attribute, which is when the keys from Fangraphs began. This then provided a challenge because while my loop began at 28, the row’s data still began at 0 and I only wanted specific data points from the row. I debugged and found the data I wanted, and I was able to extract the data I was interested in. Some of the data in this set had floats that appeared as strings with ‘%’ symbols, that I had to remove and then convert the data to a float and divide it by 100 to accurately show the decimal format. I then casted the values to floats and added the dictionary to the list again. After I did this I returned the list of keys and list of dictionaries.

In a different function, I then accounted for any rows that only had data from one of the sites. While in the final hours of my coding, I realized that some of the players that were featured in the Baseball-Reference dataset did not appear in the Fangraphs set. After further research, I came to the conclusion that Fangraphs had excluded any major league players that began the 2014 with a minor league team. In baseball, a major league team can “call-up” a player from the minors at any point in the season, so while there were players that logged over 100 Plate Appearances, Fangraphs excluded them for not beginning the year with a major league team. I accounted for these datasets through the extra function by removing any dictionaries in the list that did not have any Fangraphs statistics in the dictionary. Once I filtered out these last rows, I sorted the list by the attribute ‘WAR’, a key statistic in my analysis. After finalizing the list, I wrote this data to a csv for further analysis. Below is a brief workflow of this process.

Fangraphs Workflow:

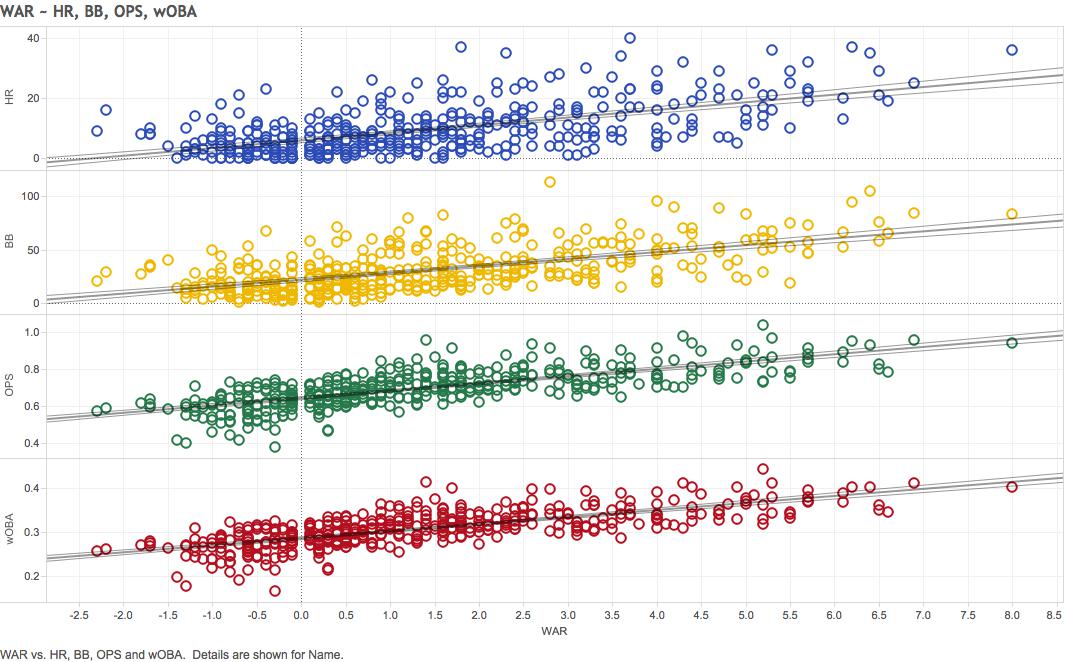
1. Extract Dataset Keys
   1. Filter any keys that were already in the Baseball-Reference key list
   2. Remove any unnecessary punctuation (quotation marks)
2. Extract Data
   1. For each row, provide the Baseball-Reference dictionary for that player
   2. Validate row name is unique and not a duplicate
   3. Filter row based on condition: Plate Appearances > 100
   4. Validate row name appears in an element in the list of dictionaries
   5. For each Fangraphs key, add row attribute to provided player dictionary
   6. Remove any punctuation (percentage symbols)
   7. Convert values to float data type
   8. Append player dictionary to list of dictionaries
3. Return List of Keys, and List of Dictionaries
4. Send List of Keys and List of Dictionaries to Extra Function
   1. Filter any rows that did not feature Fangraphs data
   2. Sort list by attribute ‘WAR’
   3. Return list of keys and list of dictionaries
5. Write Data to CSV

One of the biggest challenges I faced was properly extracting the data points in this dataset that were unique. Due to the difference in indices, I had to try various iterations to properly extract the data points I wanted. I solved this through iterations of ‘trial and error’ which proved to be successful. The other challenge I faced was accounting for the players that began their year in the minor league system. This challenge arose very late in the coding process, but after careful research I realized why the data was not processing properly. I implemented the necessary functions as described, and managed to overcome this challenge.

**Analysis**

Once I merged the data together, the analysis for this project was fairly straightforward. My motivation for this project was to compare the two datasets, as they differed in standard statistics and sabermetrics statistics, and as I began looking at the data combined together I started comparing these statistics to the statistic WAR. As I discussed, WAR is a popular metric that is used to evaluate a player’s overall performance both on offense and defense. Most of my data revolved around offensive data, and so my analysis focused on well known offensive statistics (Homeruns, Walks, On-Base Plus Slugging, and Weighted On-Base Average). Two of these statistics are standard, common, baseball stats (Homeruns and Walks) that any average baseball fan is familiar with and the other two (On-Base Plus Slugging and Weighted On-Base Average) are more analytically focused.

What I found, visualized with Visualization 1, is an interesting comparison between the four statistics and the statistic WAR. It is important to note that none of these four statistics are leveraged to calculate WAR.

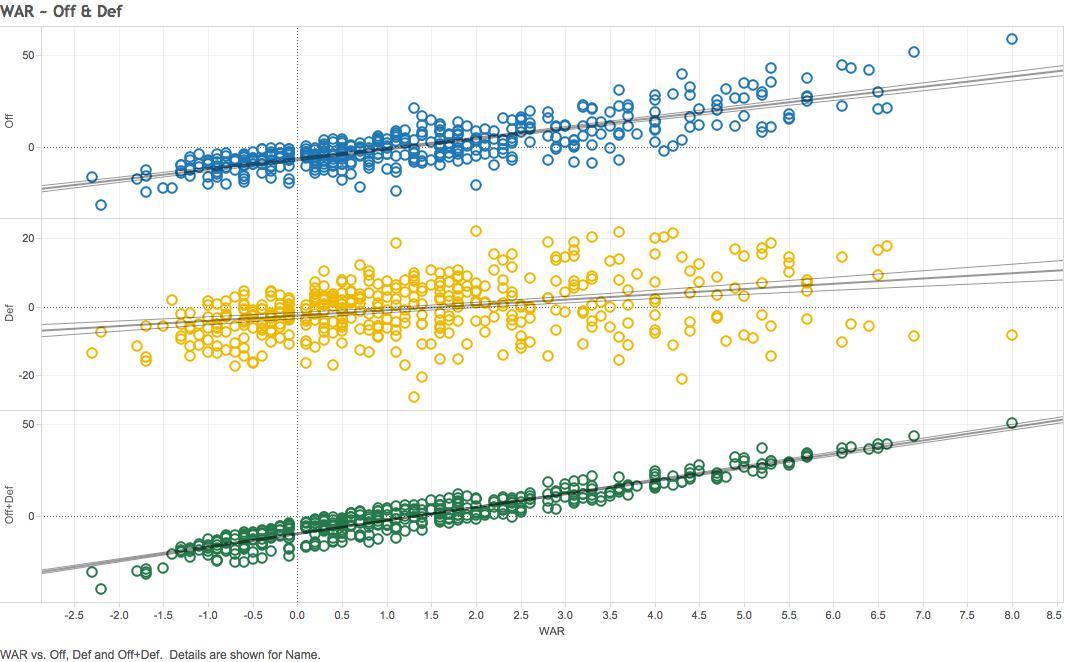


**Visualization 1**

What this visualization shows is scatter plots of all the player’s statistics compared to their WAR and a trend line showing the overall relation. When looking at the residuals of each scatter plot, I found that the basic statistics ranged, showing little correlation to the WAR metric, as opposed to the more finely-tuned sabermetric stats. It appears that Homeruns and getting “on base” do not count as much for a player’s overall WAR, which shows that player evaluation has taken further steps towards a more sabermetric mindset.

Another interesting piece of analysis I found while doing my research was the correlation between the metrics Off, Def when compared to WAR. Both metrics are used similarly to WAR, by evaluating a player’s performance overall on Offense of Defense. What I found interesting was that while this is true, the Off metric was highly correlated to WAR, where Def was all over the plot. Below I provided scatterplots, again with trend lines, to visualize this correlation.

**Visualization 2**



Based on these scatter plots, it is apparent that the Off metric is a better predictor when predicting a player’s WAR. However, it was not surprising to me me that when I added the two metrics together, it produced a very high correlation to the WAR statistic. As shown in the green scatter plot, the residuals for this comparison are very close to the trend line, indicating a higher correlation. What was most interesting, though, was how different the correlations the two metrics were separately, with the offensive metric being much more correlated.

**References**

1. *Baseball-Reference.com. 2014 Major League Baseball Standard Batting. Web. 27 Jan. 2016.* [*http://www.baseball-reference.com/leagues/MLB/2014-standard-batting.shtml*](http://www.baseball-reference.com/leagues/MLB/2014-standard-batting.shtml)
2. *Fangraphs.com. Major League Leaderboard >> 2014 >> Batters >> Dashboard. Web. 27 Jan. 2016.*[*http://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=8&season=2014&month=0&season1=2014&ind=0&team=&rost=&age=&filter=&players=*](http://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual=0&type=8&season=2014&month=0&season1=2014&ind=0&team=&rost=&age=&filter=&players=)