Building and Applying Statistical Modeling Tools for an MLB Dataset

*Michael Cowan*

*12/4/2017*

**Introduction**

In 2003, Michael Lewis released book titled *Moneyball*.1 The piece discussed Billy Beane, the general manager for the Oakland A’s, and his new and interesting approach on running a major league baseball (MLB) team. Mr. Beane was utilizing an analytics software known as *sabermetrics* to predict player’s talent based on their previous stats. In fact, the data-driven method was adopted by the Boston Red Sox who credited the technique as being a main reason for their World Series victory in 2004; a win which broke the long standing curse of the Great Bambino. The success of the Red Sox helped propel data science into the main stream of the MLB as it is today. Due to my love for the game (and especially the Red Sox), I found it fitting to apply my newly learned statistical modeling skills to an MLB batting dataset.

Baseball is truly a unique sport that favors endurance and consistency. Every team plays a 162 game season (not including post-season games). The games themselves are quite long with a minimum of nine innings being played out in a 9-on-9 fashion. The core rules of the game allow for many different stats to be recorded. Since baseball is played in a discretized pitch-by-pitch manner, it is easy to collect quantitative data on each player’s performance. In fact, the dataset chosen for this work only incorporates batting statistics. This excludes fielding, pitching, and even overall team stats that could also be analyzed. With so many stats being generated by the game, it is easy to see why the MLB is an excellent source to gather and study data.

The overall dataset chosen for this work consists of batting statistics ranging from 1955 to 2016 for both the American and National Leagues in the MLB. The data was refined such that only batting championship-qualifying seasons were looked at. The criteria to qualify for a batting award is having an average plate appearance (PA) no lower than 3.1.2 PA can be described as:

. (1)

Once the given criterion was applied, the dataset was found have 6863 rows and 30 columns. The specifications of the dataset have been summarized in Table 1 below.

Table 1: Summary of batting dataset

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Data Type** |
| Player | Unique player ID | Identifier |
| Year | year of the season | Integer |
| Team | player's team | String |
| League | American or National League | Boolean |
| G | games played | Integer |
| AB | at bats | Integer |
| R | runs scored | Integer |
| H | hits | Integer |
| 2B | doubles | Integer |
| 3B | triples | Integer |
| HR | homeruns | Integer |
| RBI | runs batted in | Integer |
| SB | stolen bases | Integer |
| CS | caught stealing | Integer |
| BB | walks | Integer |
| SO | strike outs | Integer |
| IBB | intentional walks | Integer |
| HBP | hit by pitch | Integer |
| SH | sacrifice hits | Integer |
| SF | sacrifice flies | Integer |
| GIDP | grounded into double plays | Integer |
| Avg. | batting average | Float |
| PA | plate appearances | Float |
| Age | age of player | Integer |
| Height | height of player | Integer |
| Weight | weight of player | Integer |
| Bats | left, right, or switch | Ternary |
| All Star | if made/played | Ternary |
| All Star Start | starting position | Integer |

With Table 1 alone, one can learn a lot about the dataset described. For instance, the majority of the parameters are integers which results in the data being highly discontinuous. This can cause complications for certain statistical modeling approaches. The only two continuous parameters are plate appearances and batting average. However, both PA and Avg. are not independent parameters. This will be important when fitting multivariate models since it is imperative that there are no strong dependencies between inputs. Besides integers, the dataset also contains some binary and ternary values. This allows for some interesting investigations into defining “cut-off scores”. For instance, one can ask what the minimum hits, batting average, and RBI’s are required to make the All Star team? One last important characteristic of this dataset is that there are no defined outputs. Since virtually any parameter can be considered an output, the modeling possibilities are almost endless. As great as it sounds in theory, one must be aware of the curse of dimensionality while attempting to iteratively fit different parameter combinations to models. For this reason, the more complex models applied to the dataset in this work are based on logical assumptions about the parameters and the game of baseball itself.

In order to reduce dimensionality of the system, a cumulative rate of success (CRS) was calculated for each player’s season. CRS was calculated using

*(2)*

To further compress the data, CRS was normalized using the max value of each season. This was solved by

*(3)*

CRS will later be used to build a predictive model to access a player’s success for the remainder of their career.

**Building a Statistical Model**

As I have grown to enjoy statistical modeling, I have also developed a strong interest in coding. Learning how to use a “black box” modeling package can absolutely be done. However, I instead decided to spend the time to code my own modeling tools and apply them to the chosen dataset. With my Python package, *statmod* 3, I have created tools to conduct single and multivariate ordinary least squares, regularized regression which include LASSO, Ridge, and Elastic Net, and a neural network optimized with gradient descent back-propagation. Writing the package has allowed for a more fundamental understanding of not only how statistical modeling techniques work, but also why they are effective.

*Ordinary Least Squares (OLS)*

The first tool built within *statmod* was ordinary least squares regression, or OLS. OLS can be a powerful technique due to its fast and explicit solutions. It is based on averages of the dataset and can solve both single variable and multivariate systems. Although it sounds great on paper, OLS has a couple drawbacks. One main problem is that not all data can be fit to a linear system. On top of that, increasing the order of the model can even lead to worse fits due to a rise in oscillation. This is otherwise known as Runge’s Phenomenon and will be critical when analyzing linear models on the batting dataset. Another issue with OLS is that it limits the model to only having a single output. Though, this is much smaller an issue in comparison to the first and is an inherent limitation of the modeling approach itself. In essence, it is something that must be lived with when using OLS.

OLS is based on a simple, closed-form, linear equation and can be written as

*(4)*

where is a vector of the actual output data, are the coefficients of the model (which will later be solved for), is the input matrix of linear terms, and is the explicit error vector term. Within equation 4, the only unknown is the coefficient vector. The equation can then be manipulated to solve for the unknown, given as

. *(5)*

If the input matrix is not square, it does not have an inverse which means the equation above cannot be solved. To fix this, the Moore-Penrose Pseudo-Inverse will be used which gives a final solution to the OLS model as

. *(6)*

*Regularized Regression*

The next statistical model built in *statmod* allows for the implementation of regularized regression. Essentially, regularized regression can be split into three main subtypes: ridge regression, least absolute shrinkage and selection operator (LASSO), and elastic net. The generalized formulation that envelopes all three types can be written as

*(7)*

*(8)*

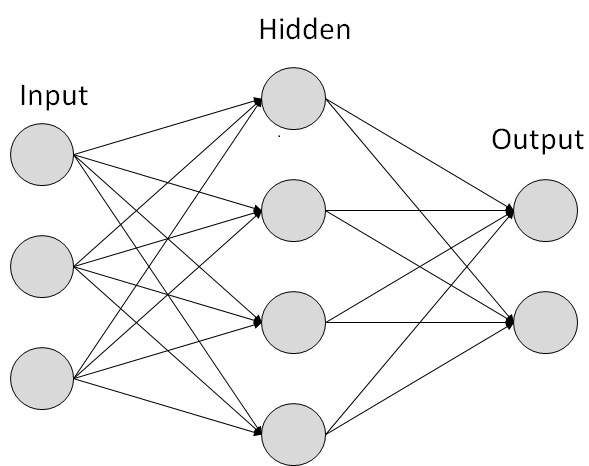
where is the loss function based on coefficients, , is defined as the effective constrain on the loss function from the elastic net balance, and is the regularized regression parameter which determines the subtype of the model. This last parameter determines the type of regression utilized and can be further defined by

. *(9)*

For optimization of the objective function, the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) was used. BFGS is a gradient descent-based method which relies on an explicitly derivable objection function. On top of finding optimal coefficients for the system, BFGS can also optimize and even if the elastic net model is being used. Though for simplicity, both and will be held constant for this work.

*Neural Network*

In order to investigate the use of more sophisticated machine learning methods, an artificial neural network (ANN) tool was built in *statmod*. Inspired by the synapse connections within the brain, ANNs use a hidden layer of neuron “nodes” which link the inputs to the outputs by use of weights, bias terms, and an activation function. The schematic below displays the general process of ANNs.



*Figure 1: ANN Diagram4*

In the simple model shown above, inputs are connected to hidden layers by

*(10)*

where is an array of the hidden layer values, is a matrix of inputs, is a matrix of weights, and is a vector representing bias. Before propagating the hidden layer to the outputs, an activation function must first be applied. An activation function can be any differentiable function that compresses the data (usually between 0 and 1). For the tool built, the sigmoidal function was implemented. It can be written as

. *(11)*

Applying this function, we now have

. *(12)*

The same method described above is applied to in order to calculate the model’s outputs. It is described by

*(13)*

Where is the final array of outputs. An important note when using the sigmoidal activation function is that all outputs will be between 0 and 1. For this reason, the target data used must be normalized before training the model.

To optimize the model, gradient descent by means of back-propagation was used.5 The generalized solution can be written as

*(14)*

where regulates the gradient descent, is the change in weights and

. *(15)*

In order to solve the partial derivative within equation 14, the chain rule was applied resulting in

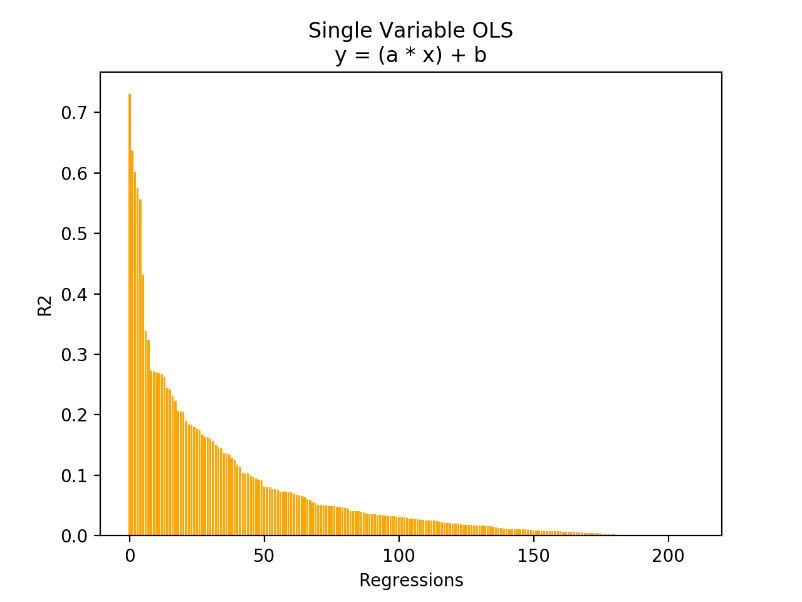
*(16)*

(17)

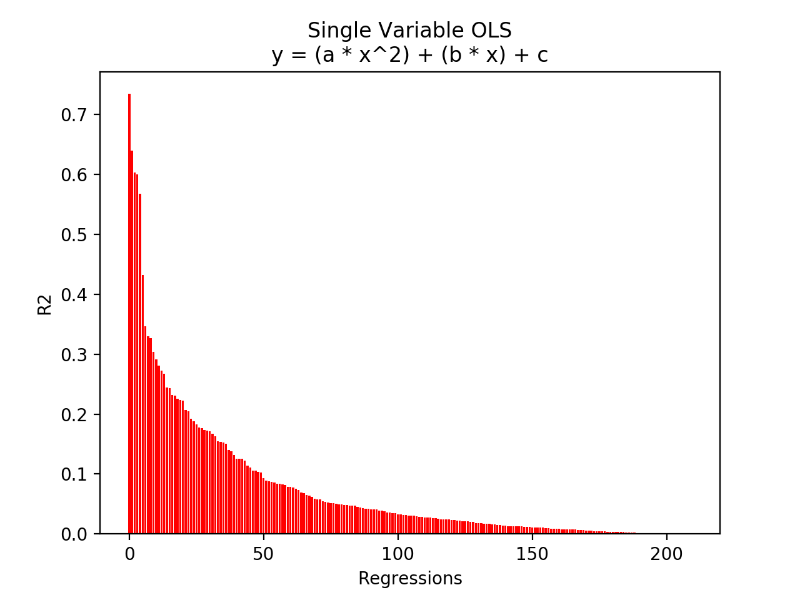
An important note of this solution is that the bias terms, , are not changed. Since they are randomly generated initially for each model, each optimized model can have different weights for the same dataset. By iteratively solving equation 14 and changing the weights of the system, the error can be minimized and a solution can be found.

**Results**

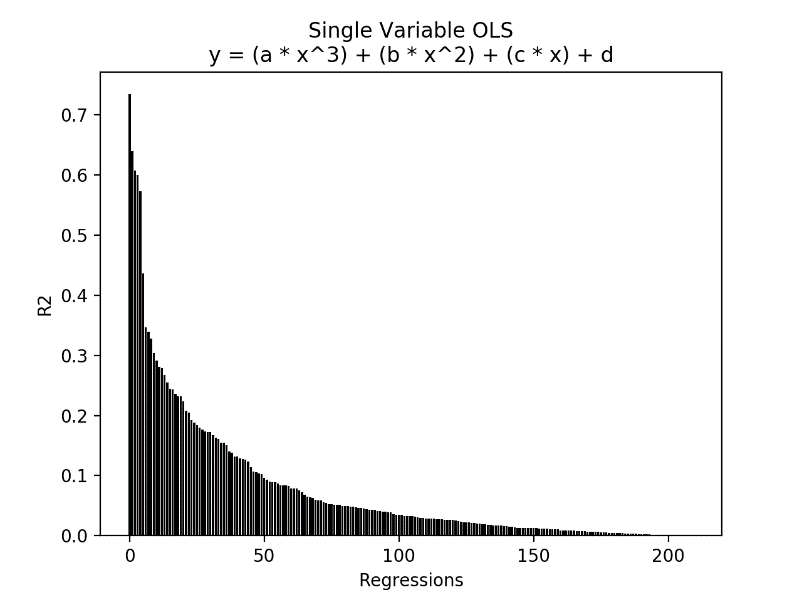
When the analyzing the MLB dataset, an iterative approach was initially needed due to no given outputs within the data. Two main types of OLS was used: single input (single variable OLS) and two inputs, or multivariate OLS. Within each type, fitting equations ranging from first order to third order were used. The results for single variable OLS can be seen below.



*Figure 2: First order single-variable OLS*

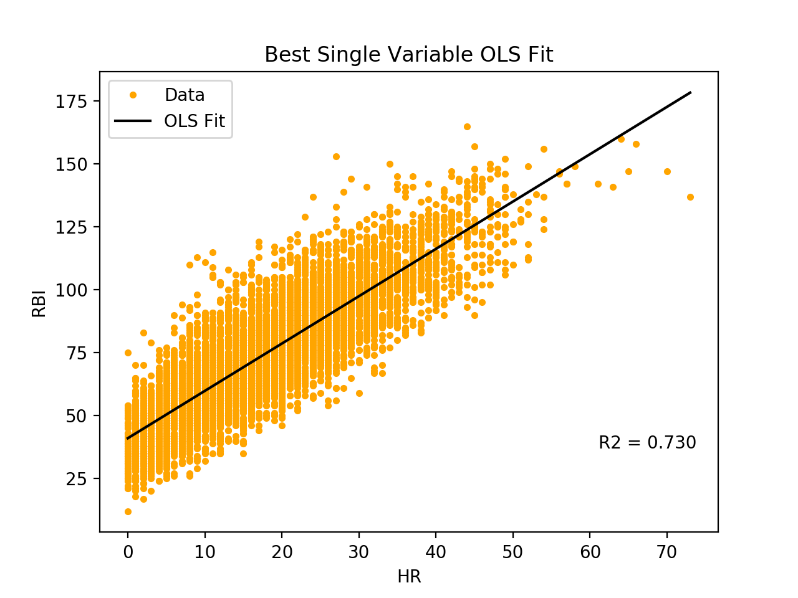
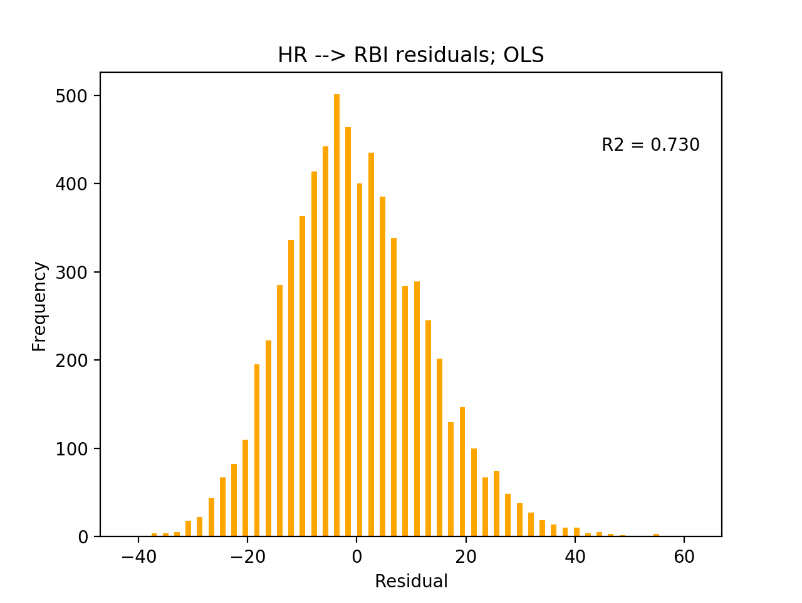
**

*Figure 3: Second order single-variable OLS*



*Figure 4: Third order single-variable OLS*

As shown in figures 2-4 above, the highest R2 value for all single-variable OLS was found to be about 0.73. This is not a great fit. It shows that there are no variable-variable correlations within the data that can be accurately represented by a linear equation. Since all three models share the same max correlation coefficient, the principle of Occam’s Razor will be applied such that only the first order OLS model will be further analyzed. This can also be done since all three types found homeruns and RBIs to be most correlated. The top results for the first order model are further displayed in the figures below. Figures for the second and third order models can be found in *Appendix A.1*.

**

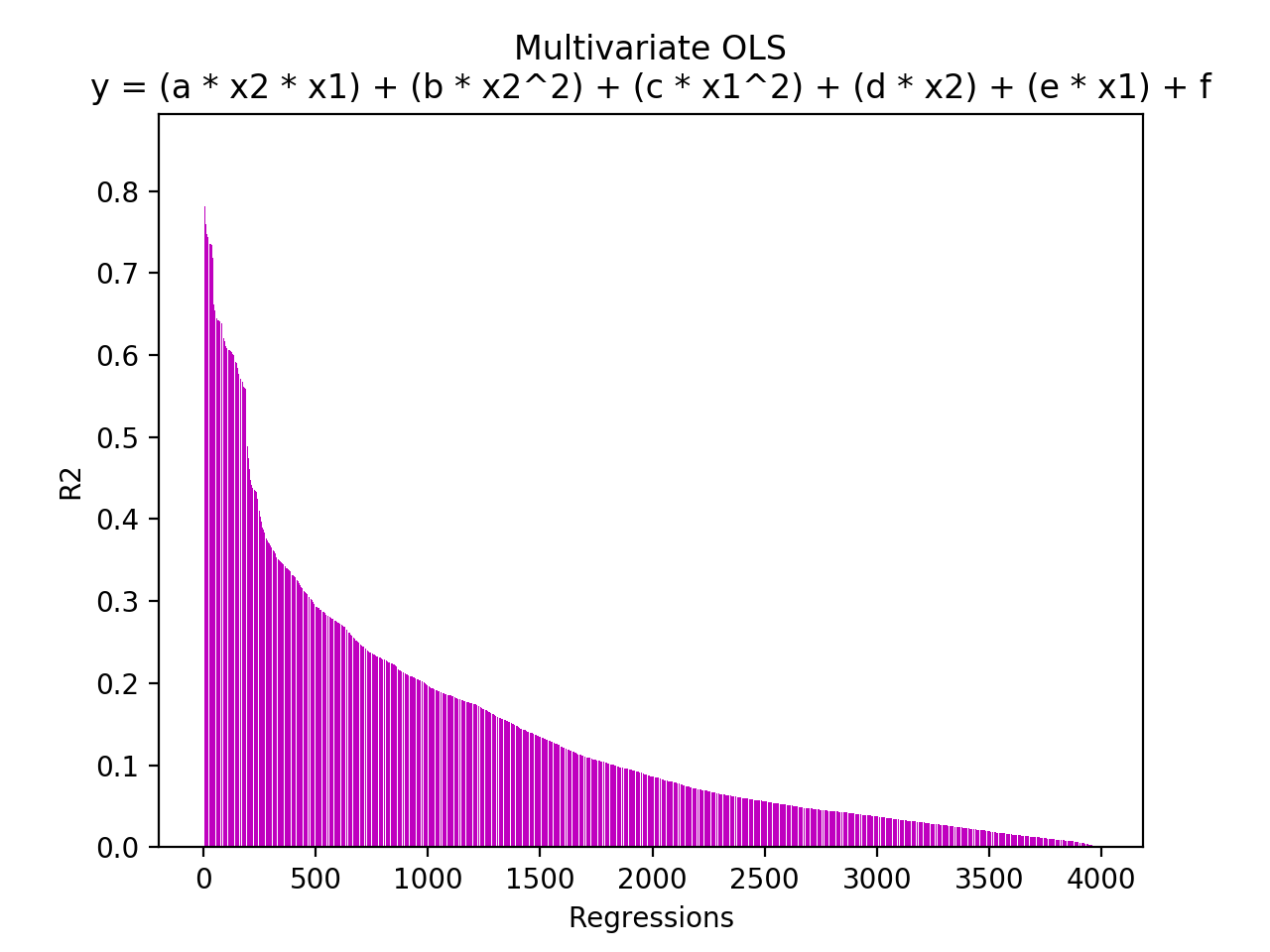
*Figure 5: First order single-variable OLS: (left) data and fit; (right) residuals*

When looking at the left plot in Figure 5, one can see that there is a general proportional relationship between homeruns and RBIs. However, the data appears to have a large variance due to the smaller range of homeruns players hit per season when compared to runs batted in. A varying range is difficult to capture for OLS due to it being a method based on averages. Though, the model can still give a rough prediction of a player’s RBIs based on number of homeruns hit due to the variance remaining close to constant throughout the data. The left plot in Figure 5 shows that the model achieved a normal distribution of residuals. This will be a desirable characteristic for all models analyzed in the work. One other important note of the model is that the p-values for both coefficients was calculated to be 0.

For multivariate OLS, only a second order model with cross-correlation terms was studied. It can be written as

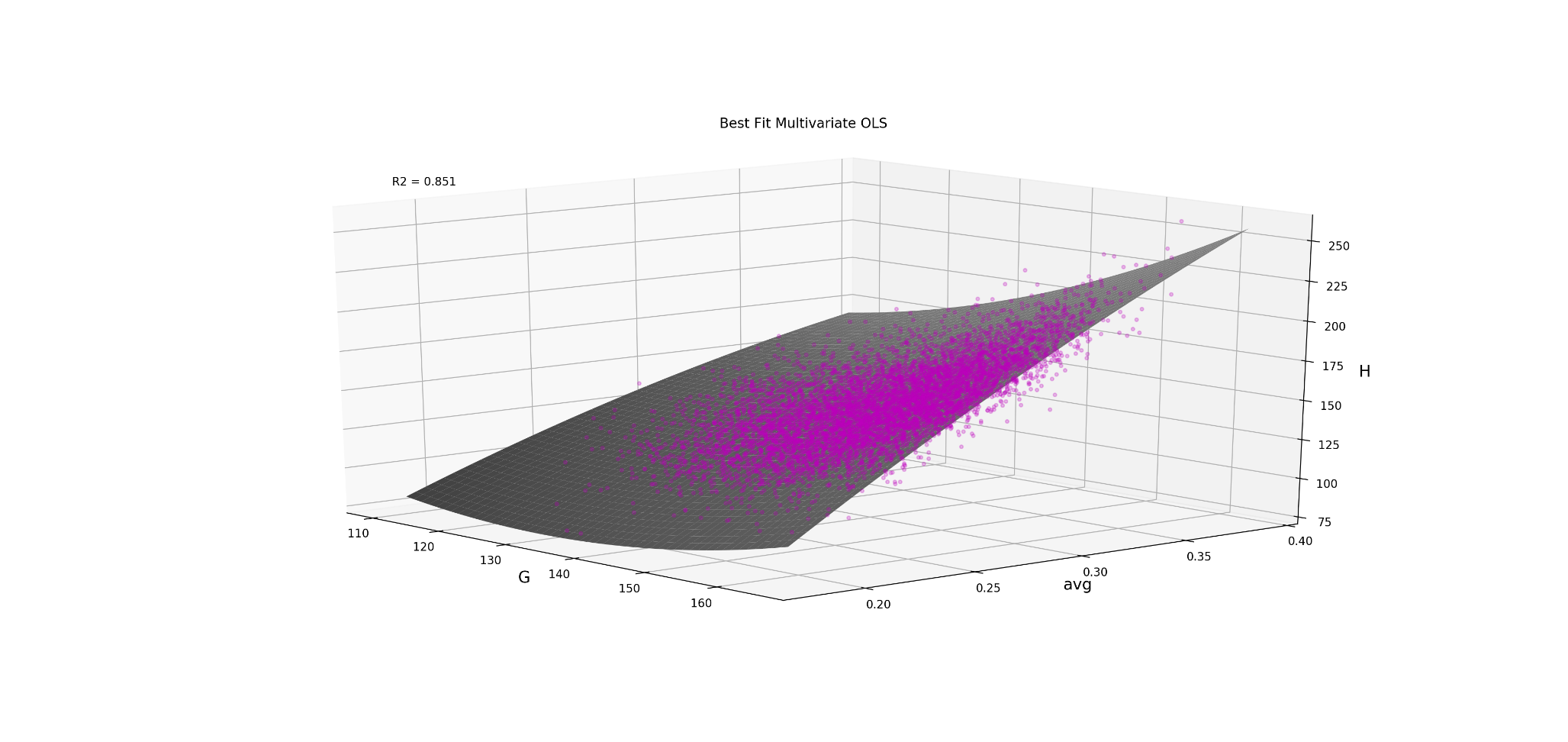
*(18)*

where are coefficients, are the two inputs and is the output. A summary of all regressions can be found in the figure below.

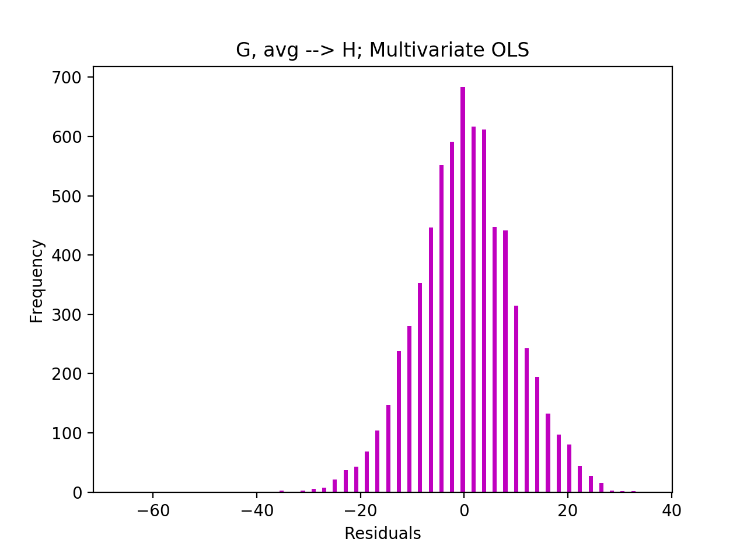
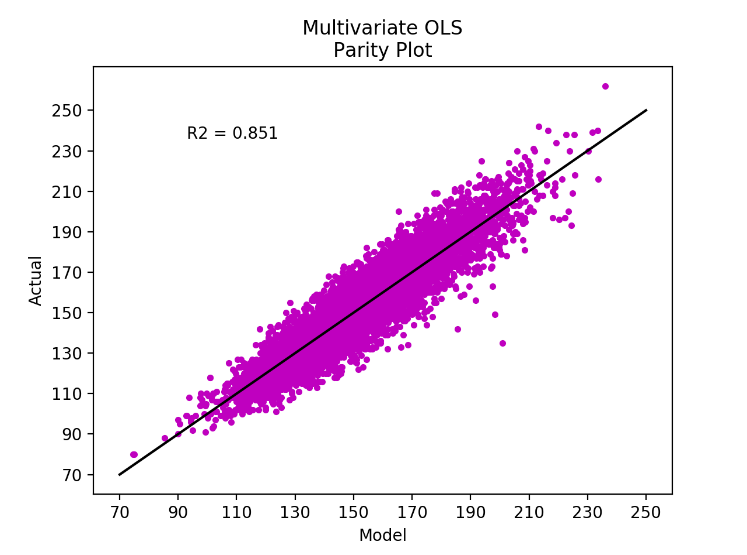


*Figure 6: Multivariate OLS regressions*

The results show that higher correlations were achieved for multivariate OLS when compared to the previous, single input studies. The best result, hits as a function of games and batting average, had a correlation coefficient of and is summarized in the plots below.

**

*Figure 7: Surface plot of multivariate OLS results for*

**

*Figure 8: Multivariate OLS results: (left) parity plot; (right) residuals histogram*

Figure 7 above shows a surface plot of both the data and model for number of games played as a function of hits and batting average. The angle of the plot allows for the slopes of the edges to be seen which gives insights into how each input affects the output. It is clear that batting average plays a much stronger role in determining total number of hits. This makes sense due to batting average being directly proportional to hits. Figure 8 displays both the parity plot and a histogram of residuals. The plots confirm that the model has a relatively good fit with a normal distribution of residuals. To confirm that all terms within the linear model are needed, p-values were calculated.

*Table 2: p-values for multivariate OLS*

|  |  |  |
| --- | --- | --- |
| *Term: (X1 = G; X2 = avg)* | *Coefficient* | *p-value* |
| *X1 X2* | *6.423* | *5.21 x 10-56* |
| *X22* | *-625.9* | *1.59 x 10-9* |
| *X12* | *0.01824* | *1.02 x 10-78* |
| *X2* | *57.69* | *0.481* |
| *X1* | *-5.893* | *5.30 x 10-81* |
| *intercept* | *392.5* | *9.74 x 10-45* |

Based on the table above, the term should not be included within the model due to its extremely high p-value of . When rerunning the calculations without this term, we the following results are achieved.

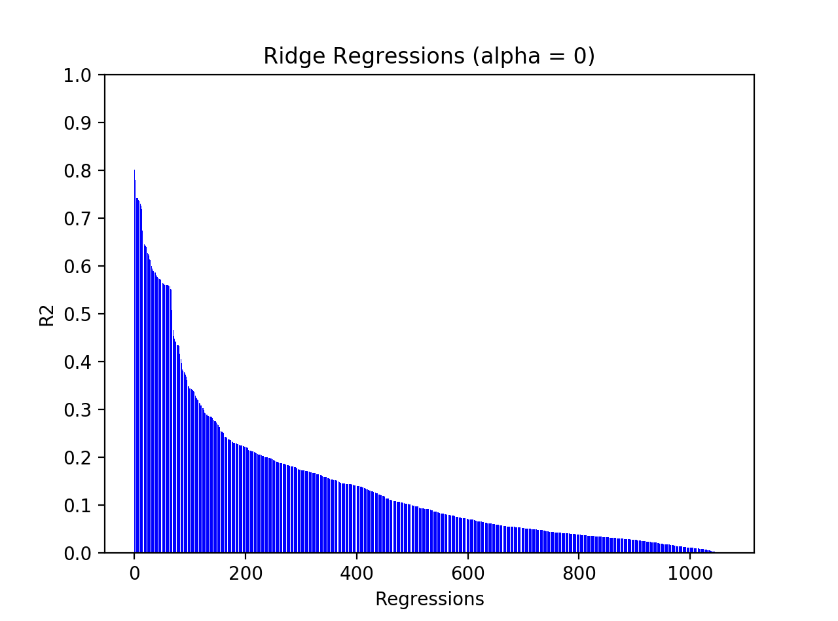
*Table 3: p-values for revised multivariate OLS*

|  |  |  |
| --- | --- | --- |
| *Term: (X1 = G; X2 = avg)* | *Coefficient* | *p-value* |
| *X1 X2* | *0.0001802* | *3.10 x 10-133* |
| *X22* | *1149* | *0* |
| *X12* | *-0.05963* | *7.03 x 10-170* |
| *X1* | *7.0888* | *0* |
| *intercept* | *-265.6* | *0* |

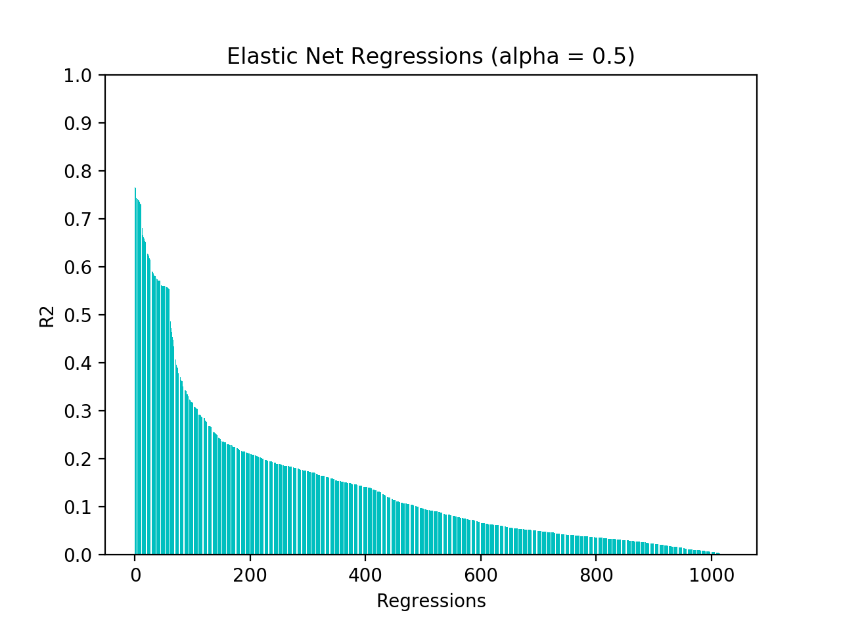
As one can see, the resulting p-values are significantly better showing that the optimal coefficients have been found for this model. On top of that, the new correlation coefficient only showed a slight decrease by changing to which further supports that this is the model that should be used. The new resulting model is given below.

*(19)*

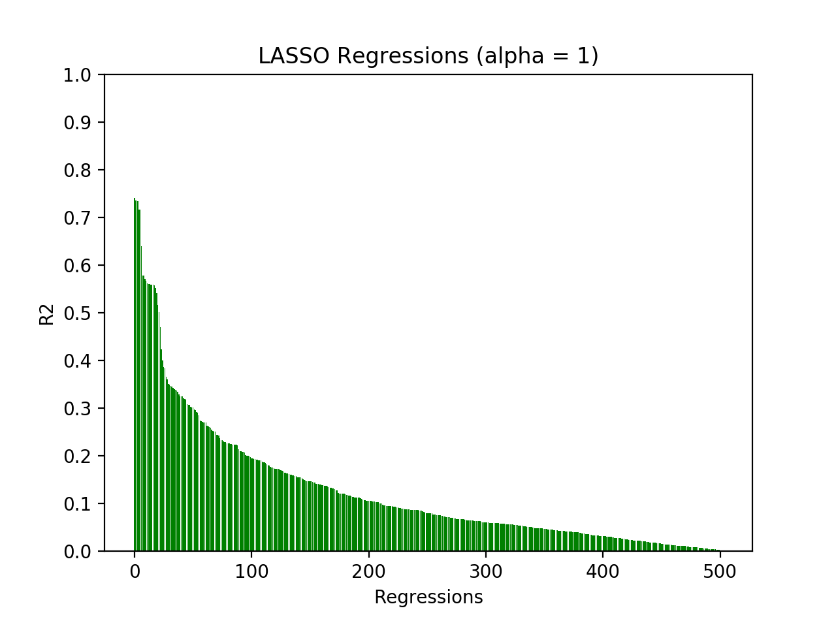
To further study the linear models used above, multivariate regularized regression was utilized to seek improved correlations. An iterative approach was again used to study all possible combinations between two inputs and a single output within the dataset. Equation 18 was used for all models. A summary of the iterative results can be seen in the figures below.



*Figure 9: Ridge Regression correlations*

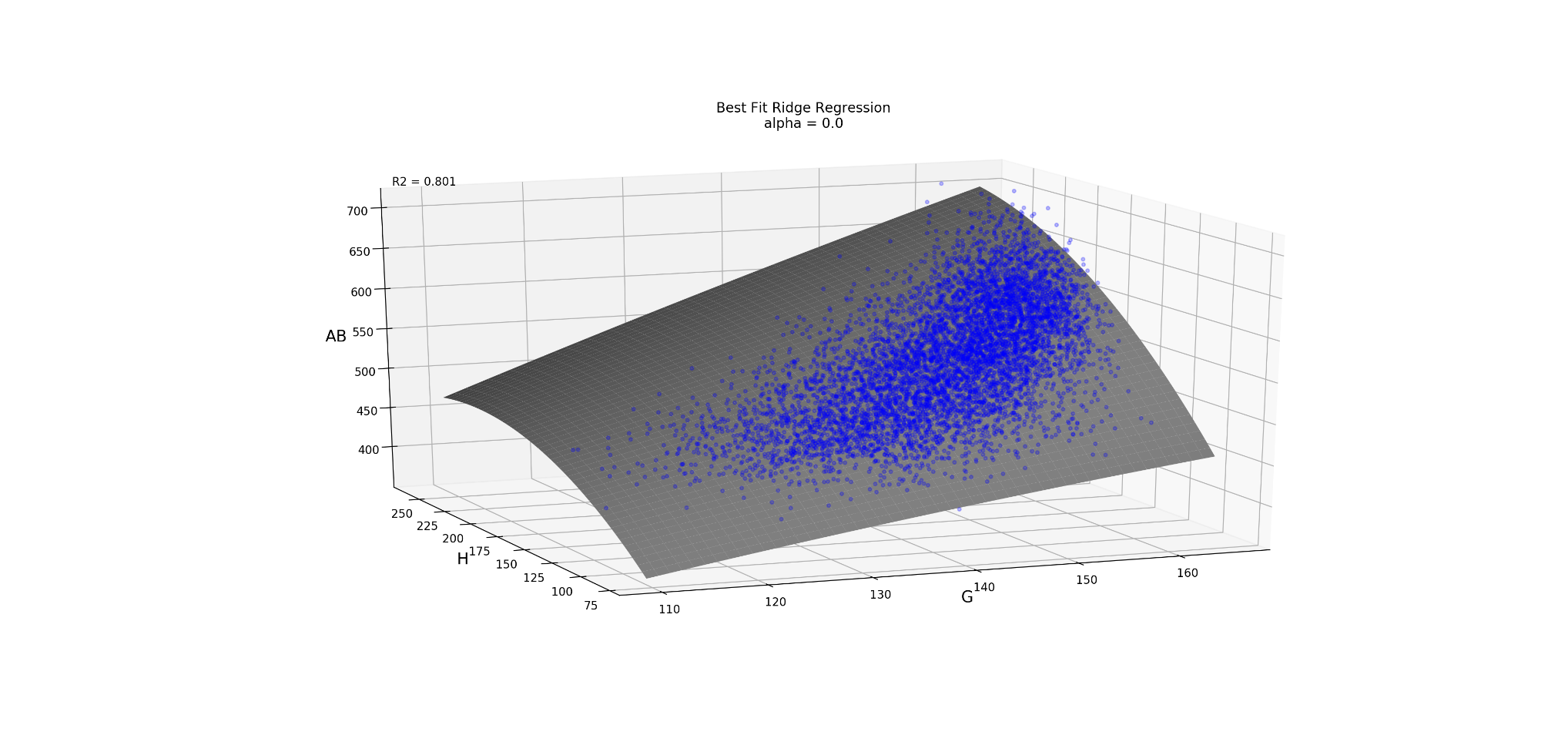
**

*Figure 10: Elastic Net correlations*

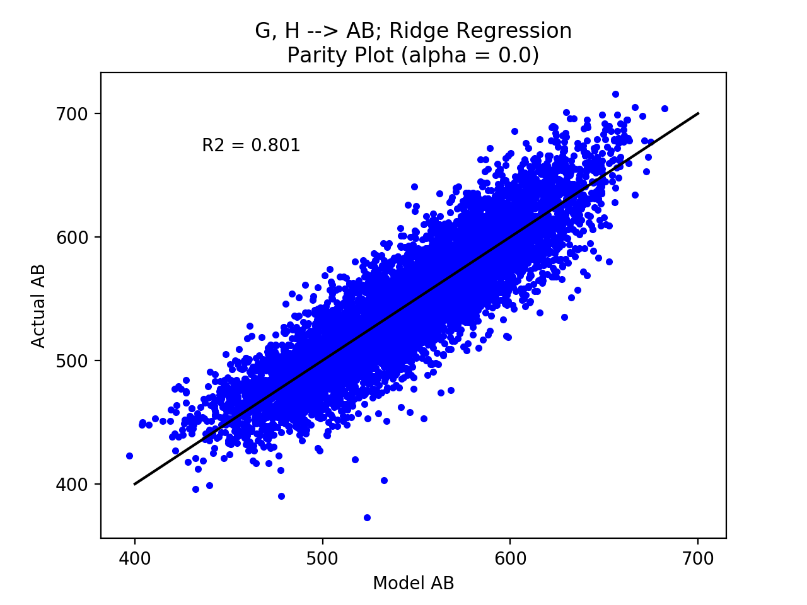
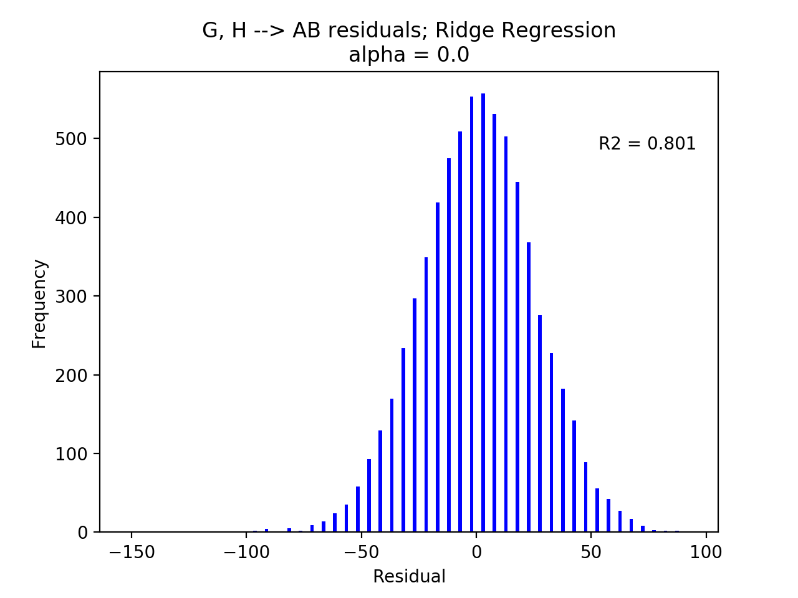
**

*Figure 11: LASSO correlations*

Figures 9-11 show that the best results for all three regression types have R2 values ranging from to with Ridge regression producing the greatest results. The best model will be further analyzed below, but plots for the other two models can be found in *Appendix A.2*.



*Figure 12: Ridge regression surface plot:*

** **

*Figure 13: Ridge regression results: (left) parity plot; (right) residuals histogram*

Figure 12 shows that both games and hits have a positive correlation to number of at bats. Also, it shows that having many games played is imperative for achieving a high number of at bats. This is a logical prediction due to MLB players consistently having very similar number of at bats per game. Similar to Figures 5 left and 8 left, the left plot in Figure 13 shows a large, yet constant variance throughout the model. Lastly, the ridge regression continues the trend of all previous models by having a normal distribution of residuals. Even though regularized regression is usually used as means to penalize functions with many coefficients, there appear to be no problems with the number of coefficients used for the model above. The model overall gives a decent prediction of number of at bats based on games played and hits.

For all previous tools applied, the results have been lack-luster. Moreover, the models do not give any meaningful data when it comes to predicting a player’s future success. To attempt to discover correlations that have more practical value, an artificial neural network was used to study the normalized cumulative rate of success (given in equation 3). Although a neural network tool was created in *statmod*, it will not be used for this work due to its slow optimization times. Instead, MATLAB’s neural network fitting tool was utilized in order to create predictive models of a player’s COSN based on a number of their previous season’s COSN.

**Appendix**

**A.1: OLS Model Results**

**A.2: Regularized Regression Model Results**

**References**

1. Lewis, Michael. Moneyball. *W.W. Norton*. **2003**.
2. Major League Baseball 2017 Official Rules. MLB.com. *MLB Advanced Media, LP*. **2017**. p 137.
3. http://www.github.com/michael-cowan/statmod.
4. https://www.tutorialspoint.com/artificial\_intelligence/artificial\_intelligence\_neural\_networks.htm.
5. Mazur, Matt. A Step by Step Backpropagation Example. **2015**. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

To do this, a neural network was implemented using a player’s CRSN from their three previous seasons in order to predict the outcome of the next upcoming year.