Building and Applying Statistical Modeling Tools for an MLB Dataset

Michael Cowan

12/5/2017

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:

$$PA = \frac{total\ plate\ appearances}{162\ game\ season} = \frac{(walks)+(at\ bats)}{162} \geq 3.1. \qquad (1)$$

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

$$CRS = \frac{avg+R+HR+RBI+SB}{SO} \ . \qquad (2)$$

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

$$CRS_N = \frac{CRS}{CRS_{MaxOfYear}} \ . \qquad (3)$$

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

$$\{\hat{Y}\} = \{b\} \cdot [X] \mp \{\varepsilon\} \qquad (4)$$

where $\{\hat{Y}\}$ is a vector of the actual output data, $\{b\}$ are the coefficients of the model (which will later be solved for), $[X]$ is the input matrix of linear terms, and $\{\varepsilon\}$ 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

$$\{b\} = [X]^{-1} \cdot \{\hat{Y}\} \mp \{\varepsilon\}. \qquad (5)$$

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

$$\{b\} = ([X]^T \cdot [X])^{-1} \cdot [X]^T \cdot \{\hat{Y}\} \mp \{\varepsilon\}. \quad (6)$$

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

$$\min\{L(\{b\}) + \lambda \cdot [\alpha \cdot |\{b\}| + (1-\alpha) \cdot ||\{b\}||]\} \quad (7)$$

$$L(\{b\}) = \frac{1}{2N}\sum_N (y_N(\{b\}) - target_N)^2 \quad (8)$$

where $L(b_N)$ is the loss function based on coefficients, $\{b\}$, λ is defined as the effective constrain on the loss function from the elastic net balance, and α 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

$$\alpha = \begin{cases} \alpha = 0 & : Ridge\ Regression \\ 0 < \alpha < 1.0 & : Elastic\ Net \\ \alpha = 1 & : LASSO \end{cases} . \quad (9)$$

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 λ and even α if the elastic net model is being used. Though for simplicity, both λ and α 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.

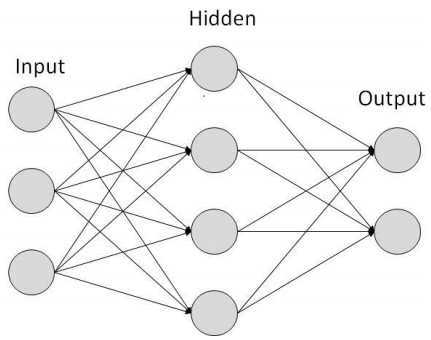

Figure 1: ANN Diagram[4]

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

$$\{H\} = [X] \cdot [W_1] + \{b_1\} \quad (10)$$

where $\{H\}$ is an array of the hidden layer values, $[X]$ is a matrix of inputs, $[W_1]$ is a matrix of weights, and $\{b_1\}$ 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

$$a(x) = \frac{1}{1+e^{-x}}. \quad (11)$$

Applying this function, we now have

$$\{NET\} = a(\{H\}). \quad (12)$$

The same method described above is applied to $\{NET\}$ in order to calculate the model's outputs. It is described by

$$[Y] = a(\{NET\} \cdot [W_2] + \{b_2\}) \quad (13)$$

Where $[Y]$ 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

$$\Delta W = -\gamma \frac{\partial E}{\partial W} \quad (14)$$

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

$$E = \frac{1}{2}([Y] - target)\,. \quad (15)$$

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

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial NET} \cdot \frac{\partial NET}{\partial W_2} \quad (16)$$

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial NET} \cdot \frac{\partial NET}{\partial H} \cdot \frac{\partial H}{\partial W_1} \tag{17}$$

An important note of this solution is that the bias terms, $b_{1,2}$, are not changed. Since they are randomly generated initially for each system, 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 analyzing the MLB dataset with OLS, an iterative approach was initially needed due to no given outputs within the data. Two main types of OLS were 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.

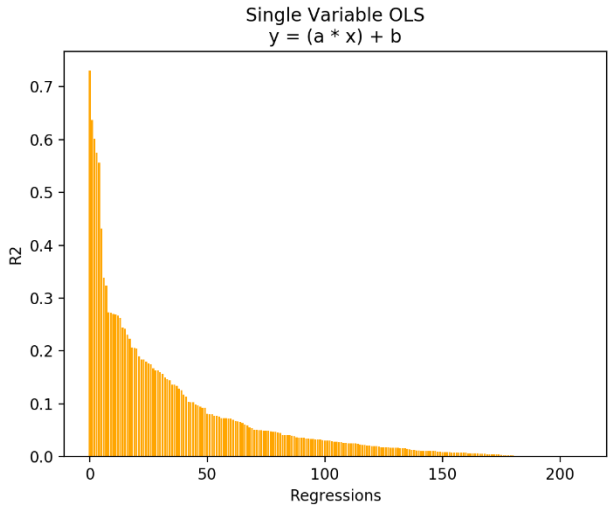

Figure 2: First order single-variable OLS

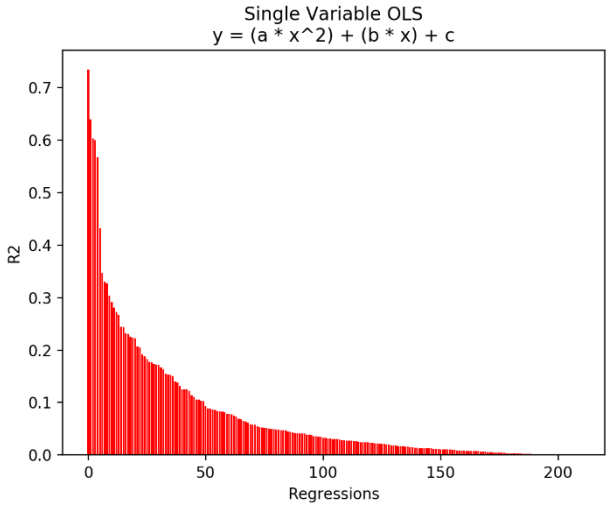

Figure 3: Second order single-variable OLS

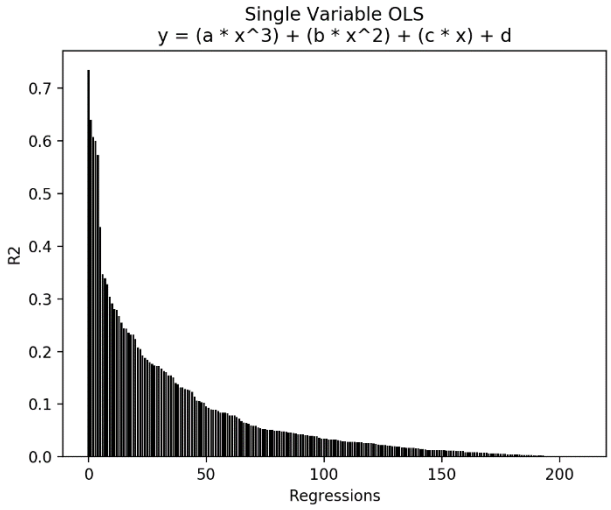

Figure 4: Third order single-variable OLS

As shown in figures 2-4 above, the highest R^2 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 was considered 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*.

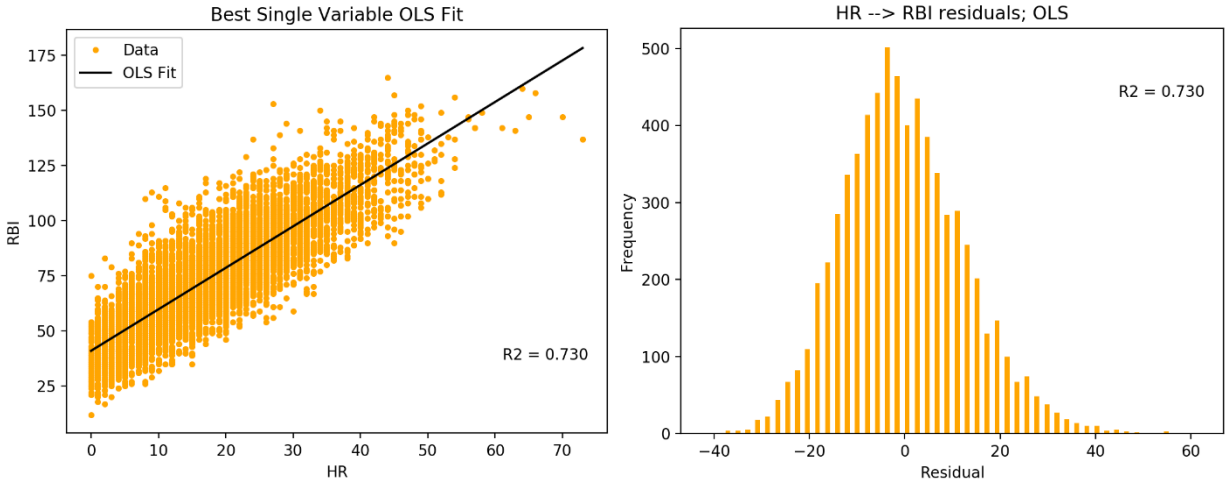

Figure 5: First order single-variable OLS: (left) data and fit; (right) residuals histogram

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

$$y = a(x_1 x_2) + b(x_2)^2 + c(x_1)^2 + d(x_2) + e(x_1) + f \tag{18}$$

where $a - f$ are coefficients, $x_{1,2}$ are the two inputs and y is the output. A summary of all regressions can be found in the figure below.

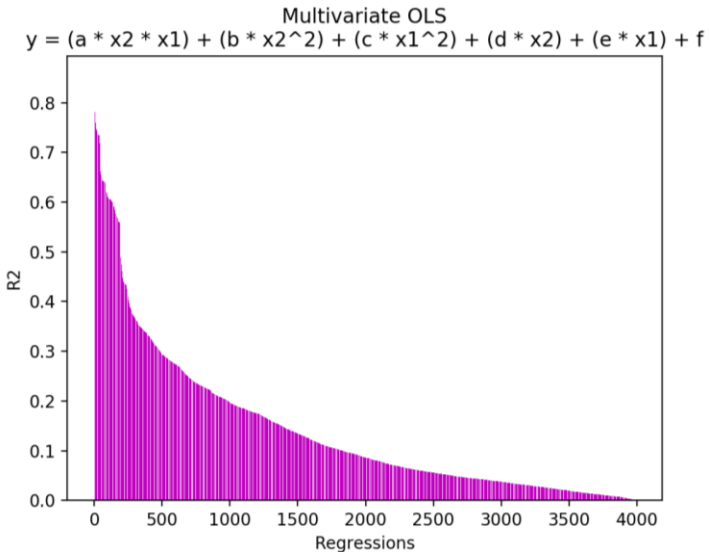

Figure 6: Multivariate OLS regressions

The results show that higher correlations were achieved for multivariate OLS when compared to the previous, single input studies. The best result, number of hits as a function of games played and batting average, had a correlation coefficient of 0.851 and is summarized in the plots below.

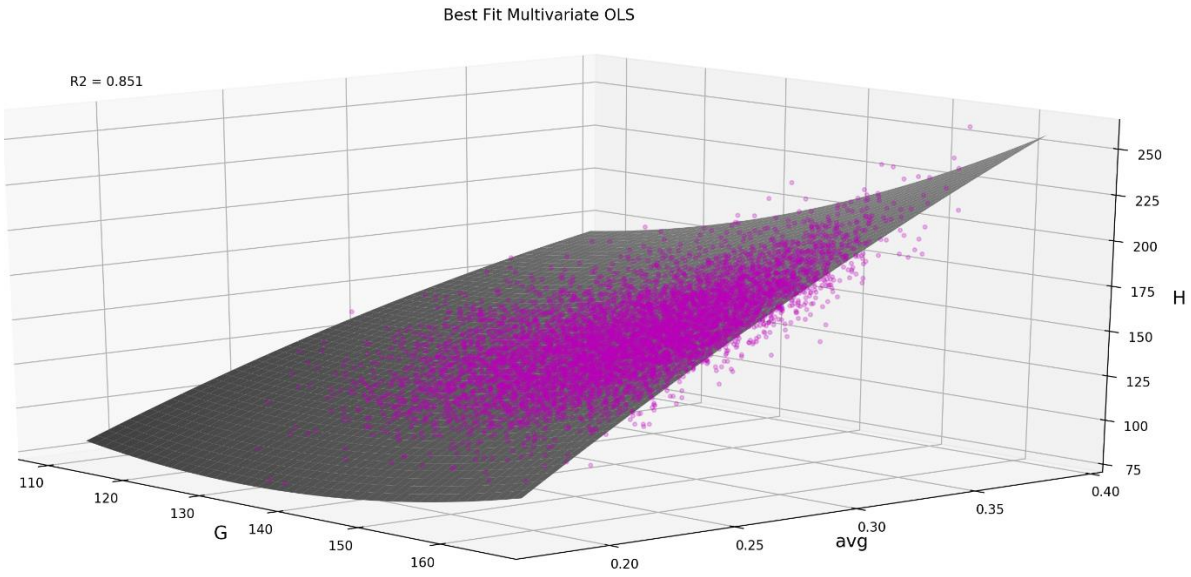

Figure 7: Surface plot of multivariate OLS results for $H = f(G, avg)$

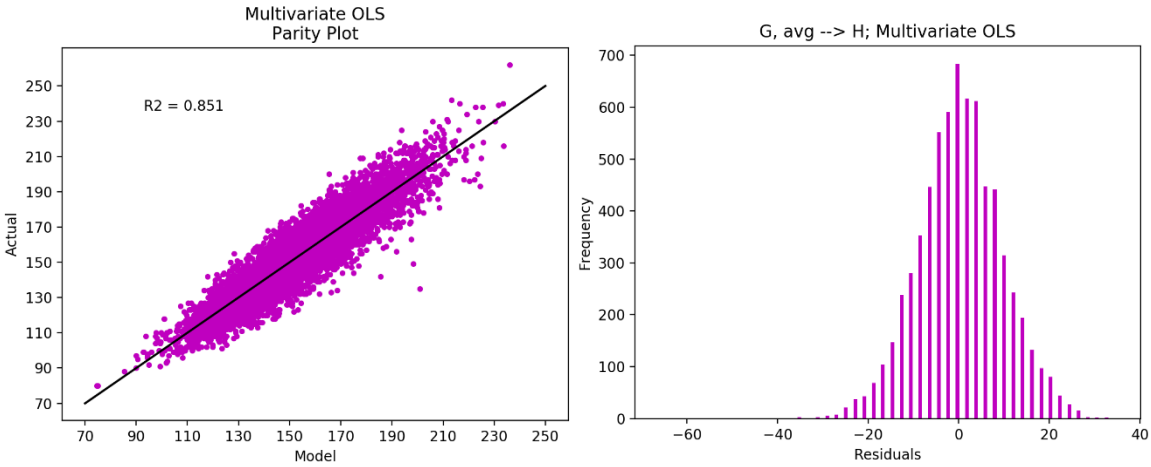

Figure 8: Multivariate OLS results: (left) parity plot; (right) residuals histogram

Figure 7 above shows a surface plot of both the data and model fit. The viewing 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: (X_1 = G; X_2 = avg)	*Coefficient*	*p-value*
$X_1 X_2$	*6.423*	*5.21×10^{-56}*
X_2^2	*-625.9*	*1.59×10^{-9}*
X_1^2	*0.01824*	*1.02×10^{-78}*
X_2	*57.69*	*0.481*
X_1	*-5.893*	*5.30×10^{-81}*
intercept	*392.5*	*9.74×10^{-45}*

Based on the table above, the X_2 term should not be included within the model due to its extremely high p-value of 0.481. When rerunning the calculations without this term, we the following results are achieved.

Table 3: p-values for revised multivariate OLS

Term: (X_1 = G; X_2 = avg)	*Coefficient*	*p-value*
$X_1 X_2$	*0.0001802*	*3.10×10^{-133}*
X_2^2	*1149*	*0*
X_1^2	*-0.05963*	*7.03×10^{-170}*
X_1	*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 0.840 which further supports that this is the model that should be used. The new resulting model is given below.

$$y = a(x_1 x_2) + b(x_2)^2 + c(x_1)^2 + d(x_1) + e \qquad (19)$$

To further study the linear models used above, multivariate regularized regression was applied 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.

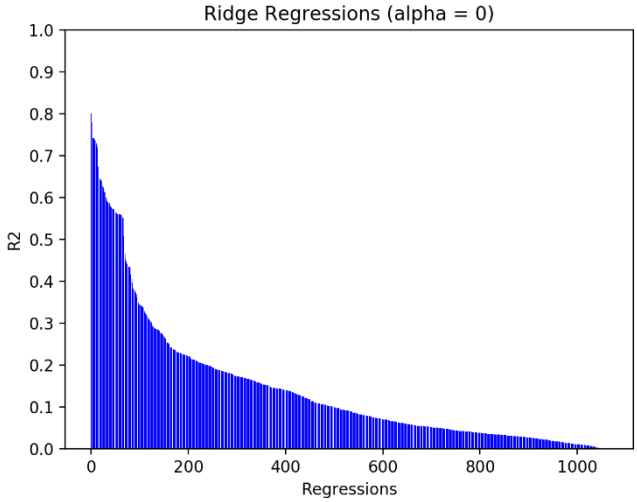

Figure 9: Ridge Regression correlations

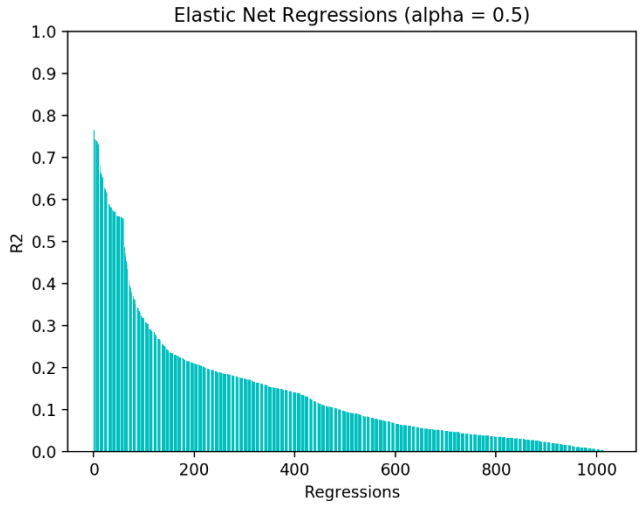

Figure 10: Elastic Net correlations

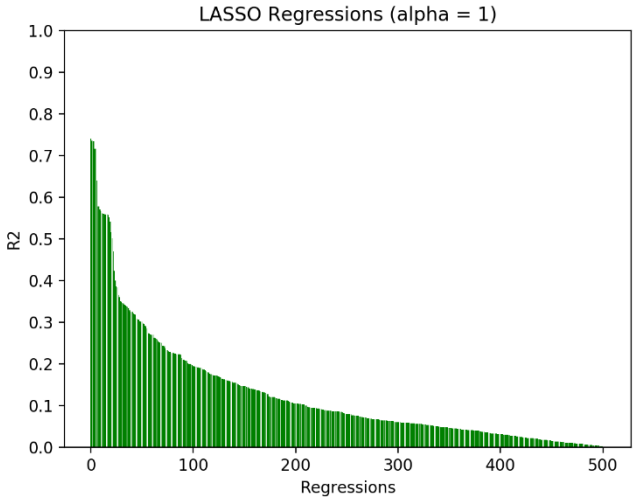

Figure 11: LASSO correlations

Figures 9-11 show that the best results for all three regression types have R^2 values ranging from 0.75 to 0.80 with Ridge regression producing the greatest results. The best model will be further analyzed below while plots for the other two models can be found in *Appendix A.2 – A.3*.

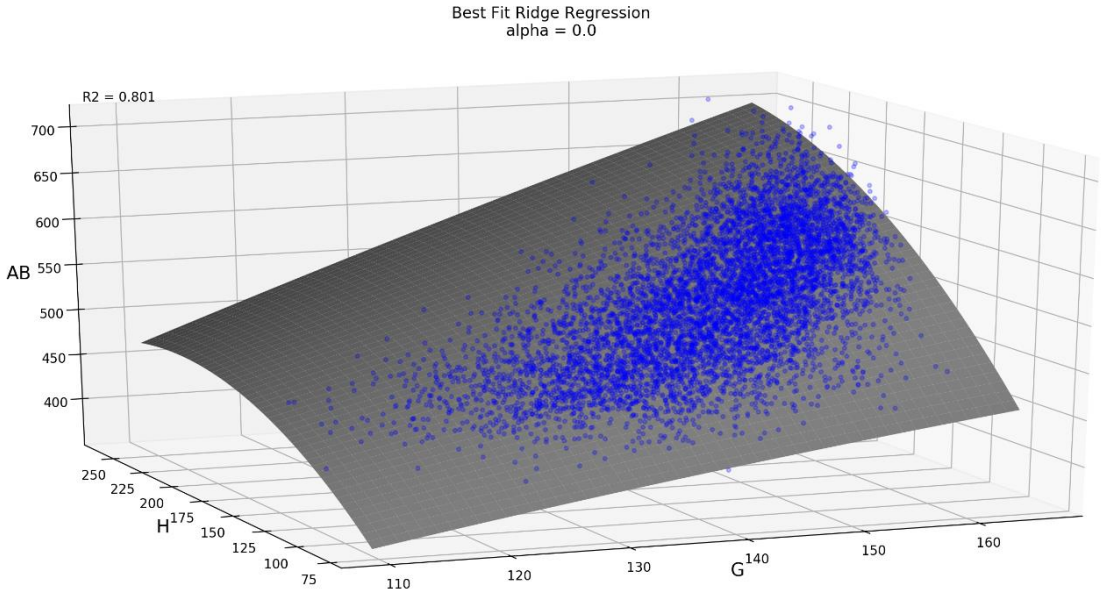

Figure 12: Ridge regression surface plot: $AB = f(G, H)$

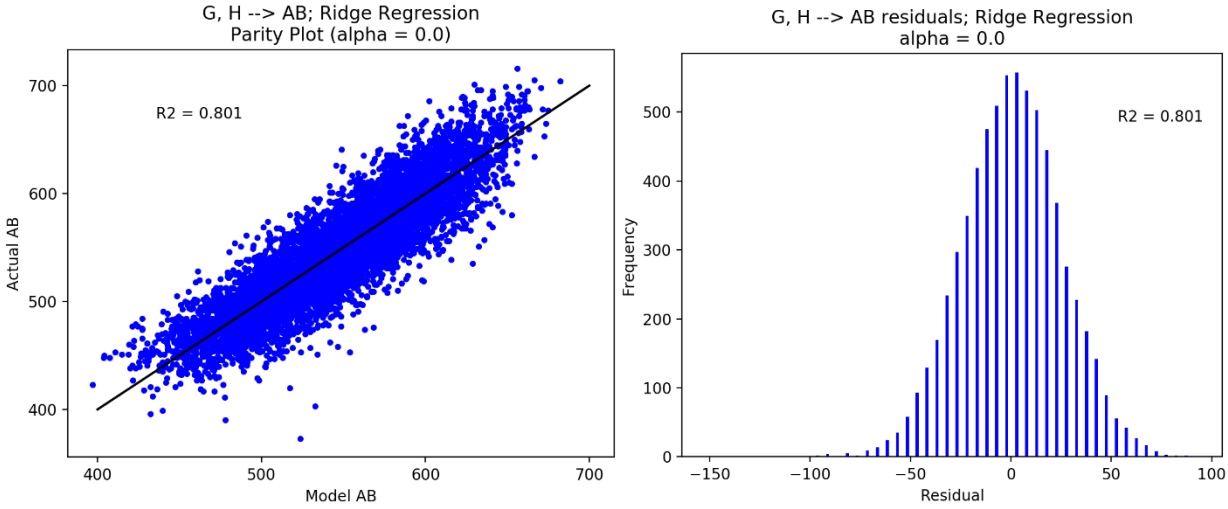

Figure 13: Ridge regression results: (left) parity plot; (right) residuals histogram

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 CRS_N based on a number of their previous season's CRS_N. In total, nine different models were trained. Each model split its data into training, validation, and testing sets at a ratio of $8:1:1$, respectively. The networks were then solved using the *Levenberg-Marquardt* training algorithm. Results of the work are summarized in the table below.

Table 4: Neural Network CRS_N Results

Name	Previous Seasons	Data size	Hidden Neurons	Mean Square Error (MSE)			Total R^2
				Training	*Validation*	*Test*	
CRS_{N-2}	2	2172	30	1.56E-02	1.81E-02	1.76E-02	0.68677
CRS_{N-3}	3	1579	30	1.56E-02	1.63E-02	1.81E-02	0.70946
CRS_{N-4}	4	1128	10	1.78E-02	1.48E-02	1.21E-02	0.70344
CRS_{N-5}	5	795	35	1.63E-02	1.82E-02	1.93E-02	0.73175
CRS_{N-6}	6	556	15	1.67E-02	1.97E-02	1.00E-02	0.73333
CRS_{N-7}	7	384	10	1.70E-02	2.45E-02	1.86E-02	0.70858
CRS_{N-8}	8	259	7	1.73E-02	1.70E-02	1.50E-02	0.72338
CRS_{N-9}	9	172	6	1.74E-02	1.18E-02	1.30E-02	0.72231
CRS_{N-10}	10	105	3	1.60E-02	1.21E-02	5.34E-03	0.73433

Overall, the training of the models showed little success with correlation coefficients ranging from $0.686 - 0.734$. However, all results were found to have relatively low mean square errors. Table 4 shows the various number of hidden neurons that were used for each of the networks. Generally speaking, more neurons can lead to an overfitting model while too few can cause an under-fit. To combat this concern, the number of neurons for each network was found empirically by training a system multiple times in order to achieve maximum correlation with minimal MSE. For further analysis of the results, plots summarizing the six-input network (CRS_{N-6}) are given below.

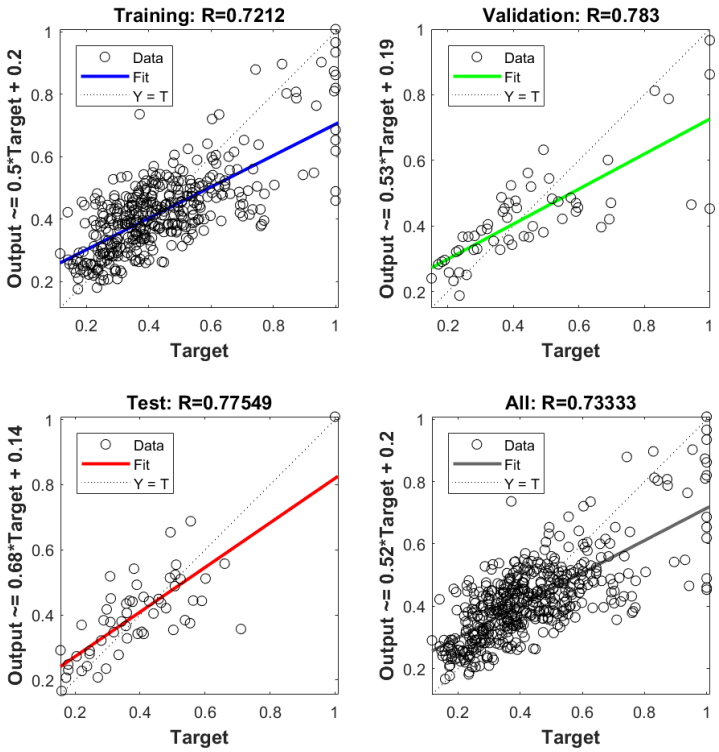

Figure 14: CRS_{N-6} neural network, parity plots

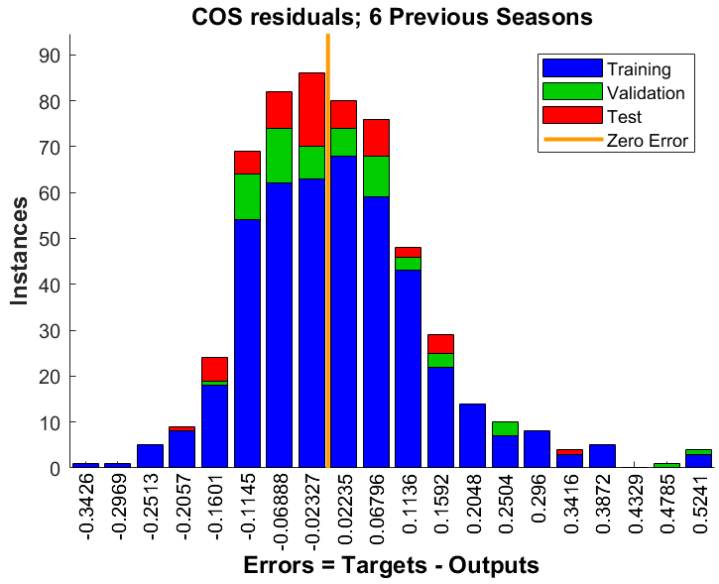

Figure 15: 6 CRS_{N-6} neural network, residuals histogram

Figure 14 shows the parity plots for all subsets of the CRS_{N-6} neural network model. The dotted line indicates a perfect correlation between model and target where the solid lines show a linear fit of the results. These plots on the figure help visualize just how far off the model is when attempting its predictions. Though, the model still has a normal distribution of residuals, as shown in Figure 15, which is highly desirable for neural networks.

Unfortunately, developing a strong predictive model with CRS_N values was unable to be achieved with a neural network. However, this is absolutely not due to the modeling capabilities of ANNs themselves. In order to prove this statement, a neural network was trained to predict number of hits as a function of games played and batting average. This is the same dataset that yielded the top results for multivariate OLS (Figures 7-8). A depiction of the ANN's results can be seen below.

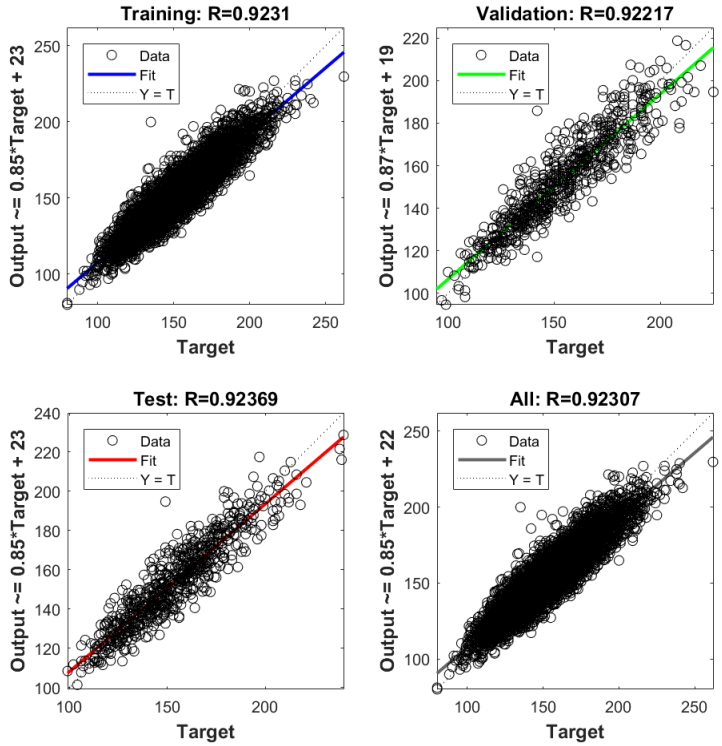

Figure 15: (G, avg --> H) neural network, parity plot

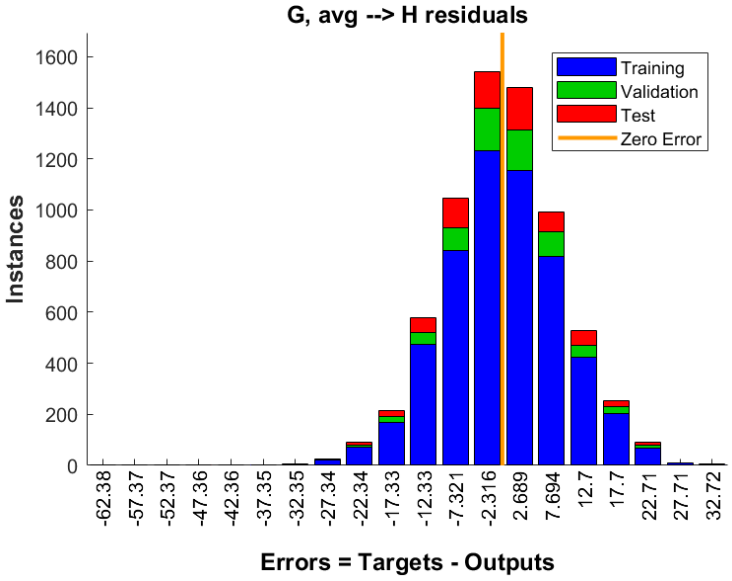

Figure 16: Figure 15: (G, avg --> H) neural network, residuals histogram

Figure 15 shows that the network achieved an excellent overall correlation of 0.92307. This exceeds the multivariate OLS model's fit while still maintaining a normal distribution of residuals (displayed in Figure 16). Compared to the OLS model, the neural network was able to better capture the correlation of inputs to outputs and is in fact the best fitting model within this work.

Methods Comparison

Even though none of the models had great success, the results can still be used to compare the capabilities of the three modeling methods applied to the dataset. OLS was found to yield solutions quickly, however its easily understandable modeling equations fare poorly when it comes to capturing complex correlations. This could be a main reason why OLS had a difficult time finding strong fits within the batting data. Comparing OLS to regularized regression, one will find that the latter does a much better job at differentiating between necessary and unneeded coefficients within the model function. This is due to regularized regression penalizing models that have too many coefficients. It is done by constraining the loss function. As for the MLB dataset, multivariate OLS did in fact produce better results than regularized regression. This could be due to the fact that there were already a low number of coefficients in the model so no penalization was required. The last and most sophisticated approach, ANN, generated the best fitting models for the dataset. Neural networks utilize a matrix-based weighting and bias method which results in capabilities of capturing complex characteristics within a dataset. This is further highlighted when its results are compared to the other two statistical models studied. On top of that, it emphasizes the other model's downfall of lacking robustness due to their stiff linear equations. However, since ANNs can fit virtually all mathematical relationships, it is imperative that care was taken when setting up the model to minimize overfitting of the data. This was achieved by splitting the data into training, validation, and testing sets and varying the number of hidden layer neurons. Although all of the models have both pros and cons, it is apparent from this work that neural networks do the best job at capturing the complicated characteristics of the MLB batting dataset.

Future Work

In order to further the results of the work done and approach more practical insights, it will be necessary to focus on improving the cumulative rate of success model. This is due to the fact that the neural network was the most successful in model formation for the batting dataset. Also, CRS has the most potential of providing valuable information to a scout or manager when attempting to predict a player's future success in the league. One main way to improve the system is to reevaluate the formulation of the CRS equation itself (equation 2). When looking at the equation, one can see that almost all batting stats included are evenly weighted. One option for a future study would be to better weight the parameters. For instance, when analyzing a player's batting performance, it does not make much sense to evenly weight their homeruns to their stolen bases. By weighting each of the parameters such that the homeruns account for more of the overall score, a better representation of CRS could be achieved. One last way to improve the rate of success equation would be to first normalize all parameters and then weight them. By normalizing each parameter based on the season, the data would give a better representation of where each player's skills lineup when compared to all other players within that season.

With one of the newly revised CRN equations, it is the hope that a new neural network model could be trained resulting in better overall fitting and predictive capabilities.

Conclusion

Baseball is not only America's pastime, but also an excellent source of data with interesting characteristics. From the start of the 21st century, the sport saw a boom in the application of data analytics to further improve scouting and coaching. Motivated by this relatively new approach, the statistical modeling toolset, known as *statmod*, was built in order to study relationships within an MLB batting dataset. Unfortunately, almost all results yielded weak correlations. This was not too surprising for the OLS and regularized regression models due to the limitations of linear models (i.e. Runge's Phenomenon). When attempting to create a predictive cumulative rate of success model using a neural network, the results were found to be poor as well. Though, the poor fitting was most likely due to the formulation of CRS itself. Overall, the work showed that not only can MLB batting data be modeled statistically, but it should for the further advancement of the game.

Appendix

A.1: OLS Model Results

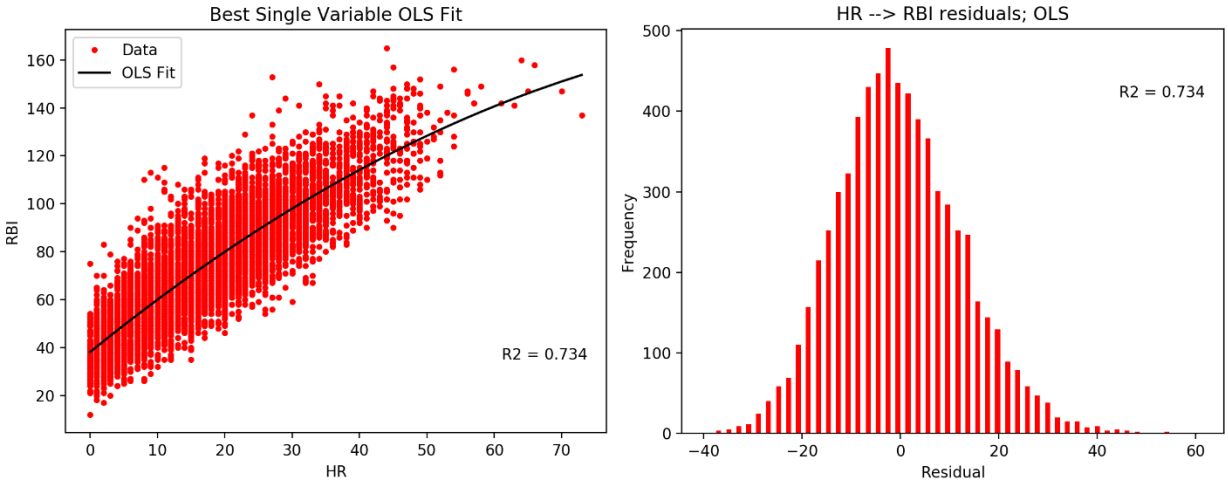

Figure A.1.1: Second order OLS: (left) data and fit; (right) residuals histogram

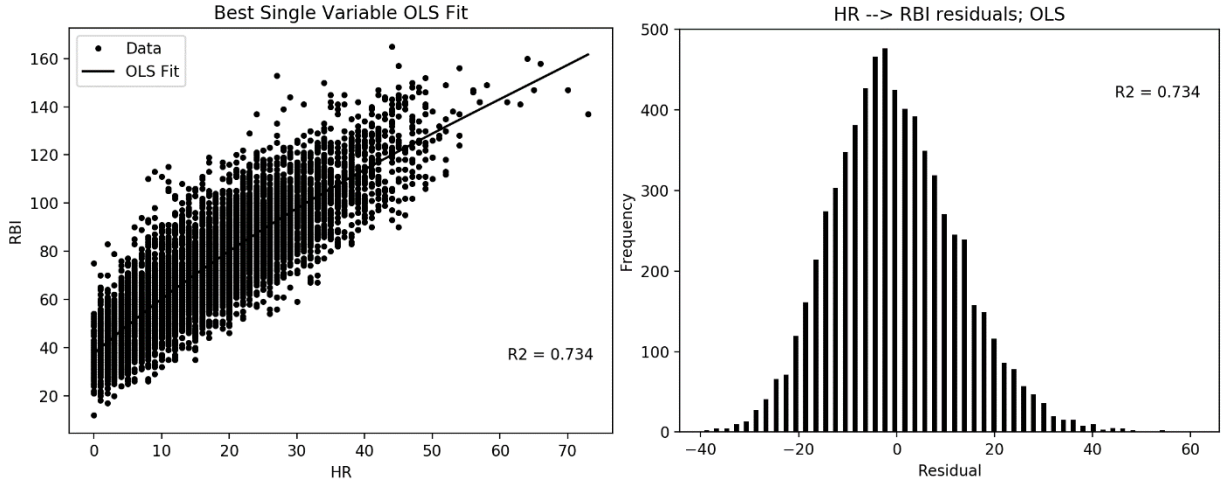

Figure A.1.2: Third order OLS: (left) data and fit; (right) residuals histogram

A.2: Elastic Net Regression Model Results

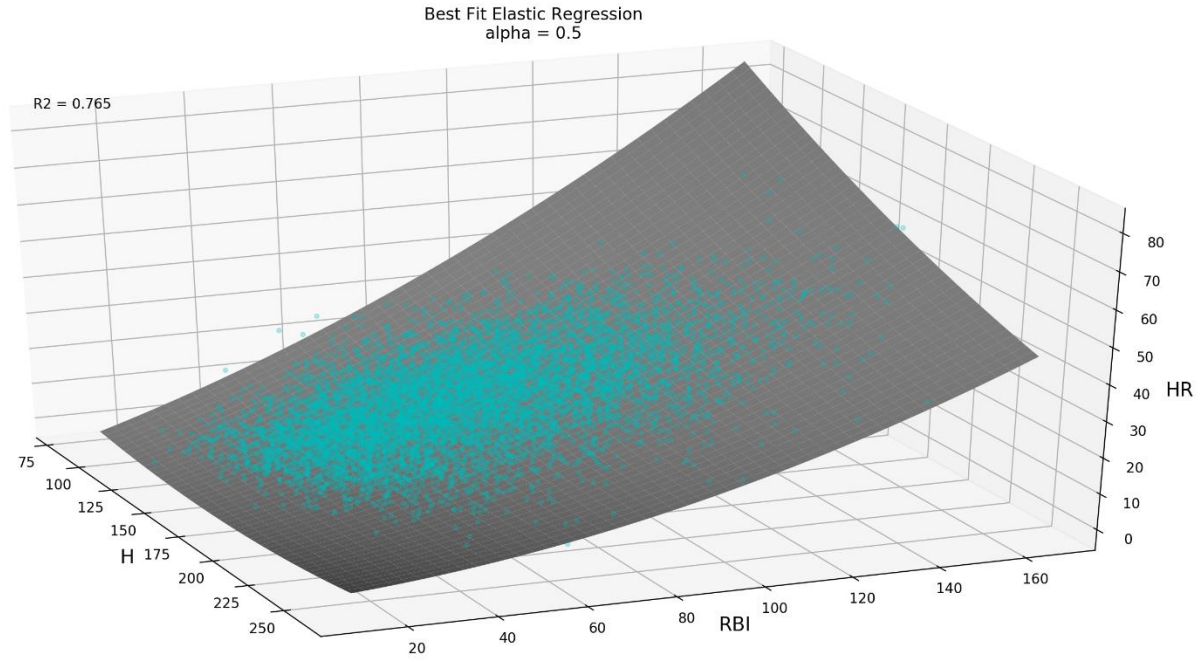

Figure A.2.1: Elastic net regression surface plot: $HR = f(H, RBI)$

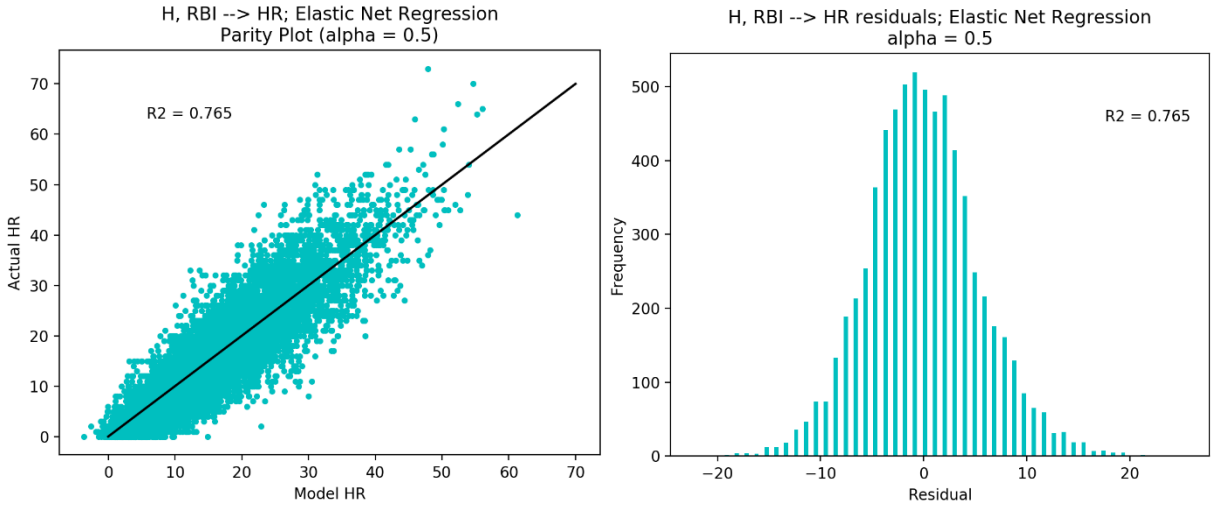

Figure A.2.2: Elastic net regression results: (left) parity plot; (right) residuals histogram

A.3: LASSO Regression Model Results

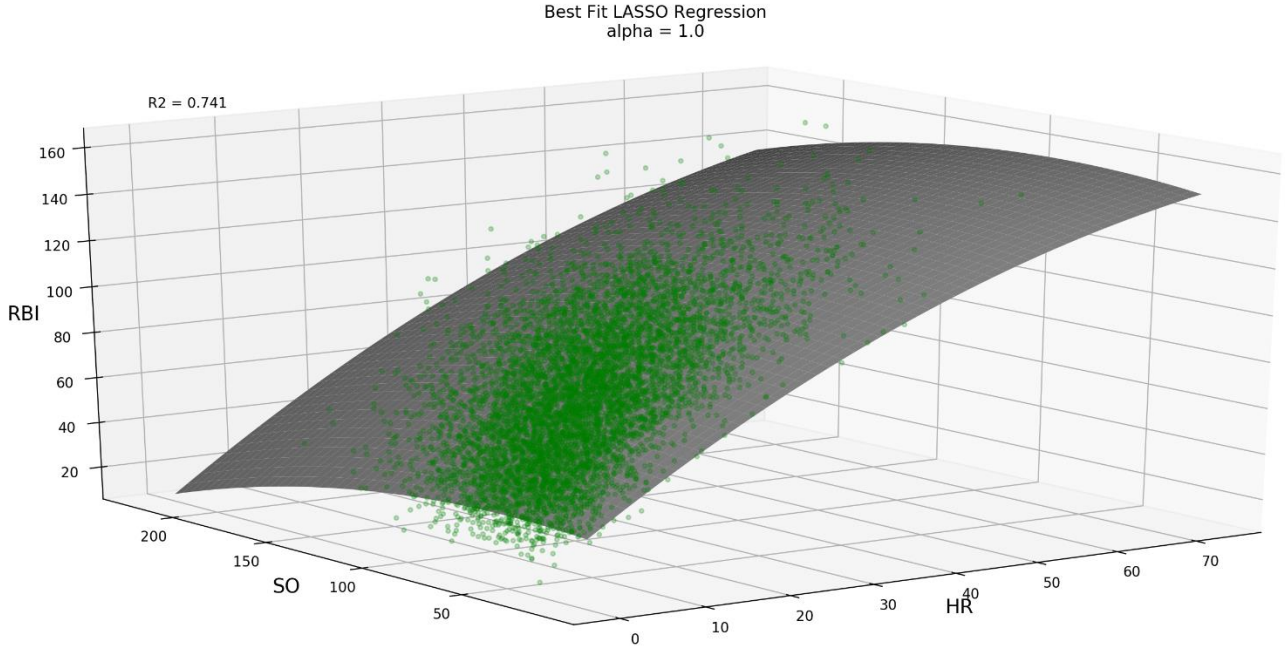

Figure A.3.1: LASSO regression surface plot: $RBI = f(SO, HR)$

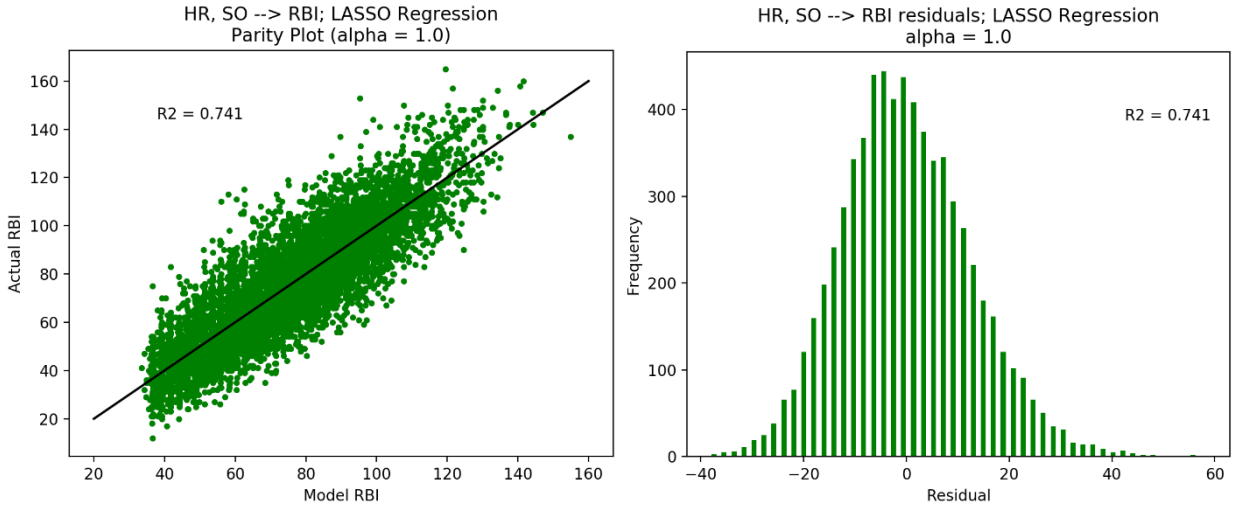

Figure A.3.2: LASSO regression results: (left) parity plot; (right) residuals histogram

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/.