Building and Applying Statistical Modeling Tools for an MLB Dataset

*Michael Cowan*

*12/2/2017*

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, most of the models applied to the dataset are based on logical assumptions about the parameters and the game of baseball itself.

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.

**Cumulative Rate of Success (CRS)**

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. 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.

**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)*

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

. *(8)*

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.

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.