Analysis of Major League Baseball Pitching in the Last Decade

August 8, 2021

Introduction

Over the last century or so, the sport of baseball has grown to become one of the most popular sports in the United States. The game, which is often referred to unofficially as an "American National Pastime", involves two teams that takes turns playing "offense" and "defense", that is, "batting" and "pitching/fielding". The objective of the game is to score the most runs, which is accomplished when playing offense by hitting balls into the field of play. Upon hitting a ball, an offensive player attempts to run bases and reach home plate before the defensive team can record outs. Today, the highest level of baseball in the United States is played in an organization called "Major League Baseball". The league was first founded in the late nineteenth century, and has evolved to include a total of 30 teams, which are split equally into two divisions: 15 teams in the American League, and 15 teams in the National League.

As Major League Baseball has increased in popularity, so have the salaries earned by the players that play within. Players make millions of dollars playing each year, and one of the highest paid player positions in baseball has often been the "pitcher". The pitcher's primary role is as a defensive player, whose job is to throw ("pitch") the baseball across home plate to a catcher. Ultimately, the pitcher's objective is to retire the offensive batter, who attempts to hit the ball or draw a walk when the ball is "pitched". Because the pitcher is involved in every play in the game of baseball, good pitchers are often important to helping teams win baseball games. But what makes a pitcher good? What pitching attributes are important to pitchers and translate to winning games? And ultimately, can pitching statistics be used to determine how often a pitcher will win games? The remainder of this report attempts to answer these questions, namely, how can different pitching statistics be used to develop an optimal model for prediction of "Win-Loss Percentage" in Major League Baseball? To answer this question, ten "important" pitching statistics from the game of baseball, and their effect on "Win-Loss Percentage", will be explored. These ten attributes are as follows:

1. Strike Outs (SO)
2. Innings Pitched (IP)
3. Age
4. League (American (AL) or National League (NL))
5. Hits (H)
6. Runs Allowed (R)
7. Earned Runs Allowed (ERA)
8. Stolen Bases Allowed (SB)
9. Batting Average Allowed (BA)
10. On-base Percentage Allowed (OBP)

At a high level, a fan of Major League Baseball may suspect that these ten attributes may be useful in finding a model to best optimize "Win-Loss Percentage". To ensure a relevant and recent data exploration that includes "up-to-date" analysis, the statistics pertaining to these ten "predictors" will be taken from the last decade of Major League Baseball. A data set is pulled from Stathead Baseball's online collection of baseball statistics, and is filtered to include single season totals for individual pitcher statistics from the following categories: pitchers from the both American and National Leagues, with at least 65 innings pitched in a given season from the years 2010 to 2020. A link to this collection of data is as follows: https://stathead.com/baseball/season_finder.cgi?type=p (https://stathead.com/baseball/season_finder.cgi?type=p). This data collection and set of filters provides for a complete dataset with 2428 records, which will be used in analysis to develop and optimize a “best” model for predicting pitching “Win-Loss Percentage”.

Methods

To begin analysis, the “raw” data file obtained as described in the introduction is first explored. At first glance, some additional data cleaning is required before lower level analysis can begin. Upon inspection, the raw data contains nearly 50 different variables, including the ten predictor variables to be explored in this report. As the variables beyond these ten predictors are beyond the scope of this analysis, the first step in data cleaning is to remove the “additional” variables. Upon removal, the dataset is next examined for missing or incomplete variables. One observation is found as “incomplete”, and this variable is removed from the data set. Now, the dataset contains 11 different variables: the response “Win-Loss-Percentage” and the ten predictors; and 2427 observations.

```
#load required libraries
library(MASS)
library(faraway)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(readr)
library(knitr)
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following objects are masked from 'package:faraway':
##
##     logit, melanoma
```

```
#read data from csv file
#data has been initially 'cleaned in Microsoft Excel to remove 'other'
#variables (i.e. those that aren't the response or one of the ten predictors)
pitching_data = read.csv("mlb_pitching_data_cleaned.csv", fileEncoding="UTF-8-BOM")

#remove observations with "NA"
pitching_data = na.omit(pitching_data)

#Check number of observations
nrow(pitching_data)
```

```
## [1] 2427
```

Next, because the varaible “Lg” is a catagorical variable, representing the American or National Leagues, it is coerced for use as a factor variable.

```
#set the categorical predictor "Lg" as a factor variable
pitching_data$Lg = as.factor(pitching_data$Lg)
```

Because variables “Win-Loss-Percentage, ERA, BA, OBP, OPS” are continuous numeric variables, and “SO, IP, AGE, H, R, and SB” are discrete numeric variables, no additional cleaning is needed, and the “cleaned dataset” is ready for analysis.

We first define two functions:

1. “model_diagnostics”: check linearity, constant variance, normality of errors assumptions. It displays fitted vs residual plot and Q-Q plot. It also reports bp-test and Shapiro-Wilke test.
2. “calc_loocv_rmse”: calculate LOOCV rmse of a model, which we will use it as a metric to compare models.

```
#define a function to check linearity, constant variance, normality of errors assumptions
model_diagnostics = function(model){
  #fitted vs residual plot
  plot(fitted(model), resid(model), col = "grey", pch = 20,
  xlab = "Fitted", ylab = "Residuals", main = "Data from Model")
  abline(h = 0, col = "darkorange", lwd = 2)
  #bp-test
  print(bptest(model))

  #Q-Q plot, Shapiro-Wilke Test
  qqnorm(resid(model), main = "Normal Q-Q Plot", col = "darkgrey")
  qqline(resid(model), col = "dodgerblue", lwd = 2)
  #Shapiro-Wilke Test
  print(shapiro.test(resid(model)))
}

#define a function to calculte LOOCV rmse
calc_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}
```

Then, we explore data at a high level and look at pairs plots.

```
pairs(pitching_data)
```

All of the response variables seem to have some sort of correlation with the response. Age is somewhat questionable, but we will include it for our initial model.

We choose a simple additive model as our start point.

```r
additive_model = lm(Win.Loss.Percentage ~ ., data = pitching_data)
summary(additive_model)
vif(additive_model)
#There is huge multicollinearity issue as many of the predictors have a VIF greater than 5.

#Calculate LOOCV rmse and perform model diagnostics
calc_loocv_rmse(additive_model)
model_diagnostics(additive_model)
```

Next, we perform both aic backward and bic backward on the additive model and compare their LOOCV rmse.

```r
#using LOOCV rmse as a metric to compare models
#aic, backward
aic_model = step(additive_model, direction = "backward")
summary(aic_model)
#bic, backward
n = length(resid(additive_model))
bic_model = step(additive_model, direction = "backward", k = log(n))
summary(bic_model)



calc_loocv_rmse(aic_model)
calc_loocv_rmse(bic_model)
```

Since aic_model has the lowest LOOCV RMSE, we will try to improve based on aic_model. We decide to perform boxcox transformation to aic model and log transformation to the reponse in aic model. Since a Win.Loss.Percentage equal to zero would cause errors, we will exclude them for our transformation.

```r
#Apply box-cox transformation
boxcox(lm(formula = Win.Loss.Percentage ~ IP + Lg + R + ERA + SB,
       data   = pitching_data, subset = Win.Loss.Percentage > 0),
       plotit = TRUE,lambda = seq(0.7, 1.1, by = 0.1))

#Choose a lambda of 0.88 according to the boxcox plot
box_model = lm((((Win.Loss.Percentage^0.88)-1)/0.88)~SO + IP + Lg + R + ERA + SB,
               data = pitching_data, subset = Win.Loss.Percentage > 0)

summary(box_model)
calc_loocv_rmse(box_model)

#Apply log transformation to response
log_model = lm(log(Win.Loss.Percentage) ~SO + IP + Lg + R + ERA + SB ,
               data = pitching_data, subset = Win.Loss.Percentage > 0)

summary(log_model)
calc_loocv_rmse(log_model)
```

However, both transformations do yield a better LOOCV rmse. Therefore, we decide to add two-way interactions to aic model.

```
# Adding interaction terms between variables
int_model = lm(Win.Loss.Percentage ~ (SO + IP + Lg + R + ERA + SB) ^ 2,
               data = pitching_data)

summary(int_model)
calc_loocv_rmse(int_model)
```

This time we are seeing a slightly lower LOOCV rmse. Then, we add quadratic terms along with two-way interactions to aic model.

```
#second order model, include first-order terms, interaction and quadratic terms
#Since Lg is not numeric, we would not include quadratic term for Lg
so_model = lm(Win.Loss.Percentage
              ~ (SO + IP + Lg + R + ERA + SB)^2 + I(SO^2)
              + I(IP^2)+I(R^2)+I(ERA^2)+I(SB^2),
              data = pitching_data)

summary(so_model)
calc_loocv_rmse(so_model)
```

Model with interactions and quardratic terms yields the best LOOCV rmse. Due to high number of terms, we decide to perform aic backward algorithm on this model.

```
aic_so_model = step(so_model, direction = "backward")
summary(aic_so_model)

calc_loocv_rmse(aic_so_model)
model_diagnostics(aic_so_model)
```

This time we get a lowest LOOCV rmse.

During the process, we also check for leverages, outliers, influentials. However, in general we don’t want to remove those points unless there is a good reason to.

```
#check for unusual observations
#check for points with high leverage
sum(hatvalues(aic_so_model) > 2 * mean(hatvalues(aic_so_model)))

#check for outliers
#expect the standardized residual to be greater than 2 approximately 5% of the time
#assume > 2 is an outlier
#don't necessarily need to remove outliers, it diagnostics look good
sum(abs(rstandard(aic_so_model)) > 2)

#check for observations with large "influence"
sum(cooks.distance(aic_so_model) > 4 / length(cooks.distance(aic_so_model)))
```

Let’s see how removing points of high influence would affect aic_so_model.

```r
cutoff = cooks.distance(aic_so_model) <= 4 / length(cooks.distance(aic_so_model))

aic_so_model_lf = lm(formula = Win.Loss.Percentage ~ SO + IP + Lg + R + ERA + SB
                     + I(R^2) + I(ERA^2) + SO:Lg + IP:R + IP:ERA + Lg:R + R:ERA,
                     data = pitching_data, subset = cutoff)

summary(aic_so_model_lf)
calc_loocv_rmse(aic_so_model_lf)
model_diagnostics(aic_so_model_lf)

#Getting a lower LOOCV rmse and higher adjusted R-squared with influentials removed
#Also, we failed to reject normality of errors at alpha = 0.05
```

We are also interested in how aic_so_model performs when we split the data into training/test sets.

```r
#Split into 8:2 training/test datasets
set.seed(42058)
trn_idx = sample(1:nrow(pitching_data),
                 floor(nrow(pitching_data)*0.8), replace = FALSE)

pitching_train = pitching_data[trn_idx,]
pitching_test  = pitching_data[-trn_idx,]

#fit aic_so_model using training data only
aic_so_model_trn = lm(formula = Win.Loss.Percentage ~ SO + IP + Lg + R + ERA + SB +
    I(R^2) + I(ERA^2) + SO:Lg + IP:R + IP:ERA + Lg:R + R:ERA,
    data = pitching_train)

#Calculate rmse on both training and test dataset
train_rmse = sqrt(mean(residuals(aic_so_model_trn)**2))

obs_dif = pitching_test$Win.Loss.Percentage - predict(aic_so_model_trn, pitching_test)
test_rmse  = sqrt(mean((obs_dif))**2)
```

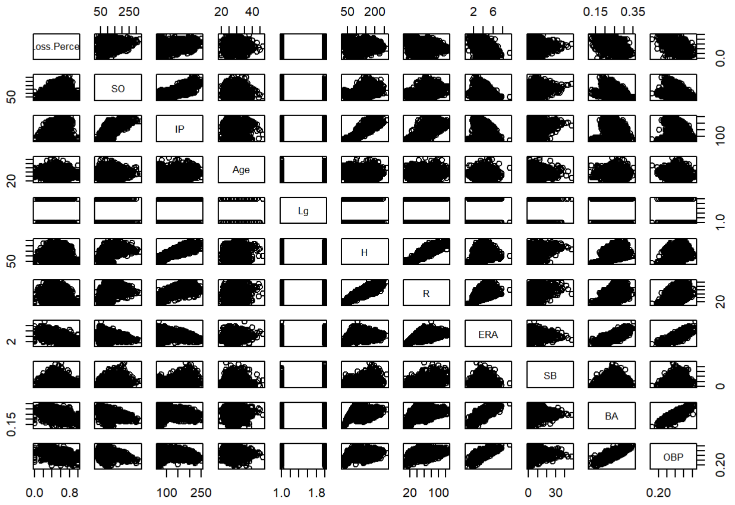
Results

By the end of our experimentation with the data and after evaluating a number of uniquely generated models, we identified a wide array of models which achieve our initial goal for prediction at a baseline level. After evaluating each model based on a number of numerical criteria based on our end goal of creating a model for prediction, we identified a pair of "best models". These models happened to be the "aic_so_model" and "aic_so_model_lf" model found in the methods section. Each model is made up of the same predictors, but the second model uses a datset that has been modified such that it contains no influential points. The following charts and graphs illustrate the performance of these models (listed as the "AIC Second Order" and "AIC Second Order (No Influential)" models), versus the performance of other candidate models. In general, these models were chosen due to their high R-squared value as well as their low LOOCV-RMSE values. The second model was generated and chosen as a final model in addition to the first model out of both consideration and caution for the effect of removing data.

Correlation

The Following pairs plot shows that we might have multicollinearity issues.

```r
pairs(pitching_data)
```



VIF result shows that we have multicollinearity issues for variables: SO, IP, H, R, ERA and BA

```
additive_model = lm(Win.Loss.Percentage ~ ., data = pitching_data)
vif(additive_model)
```

```
##      SO      IP     Age    LgNL       H       R     ERA      SB      BA     OBP
##   5.758  56.036   1.042   1.031 101.578  37.439  10.839   1.328  15.270   4.962
```

Model Performance Table

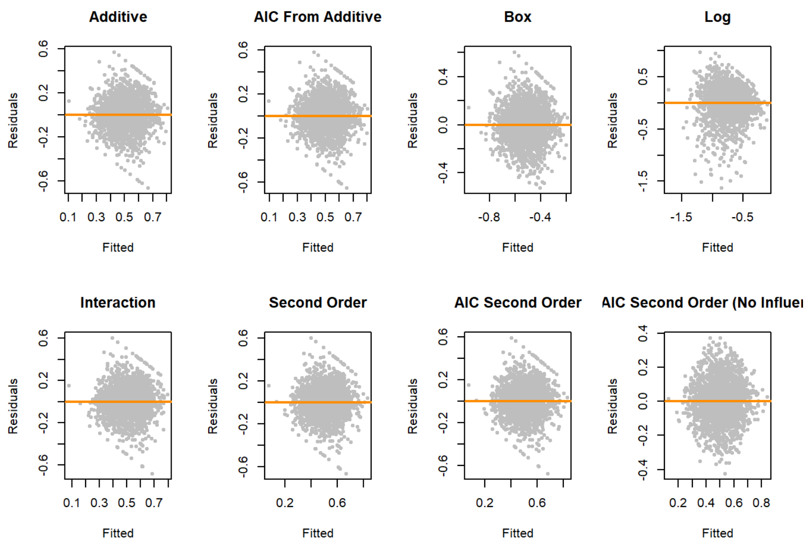
Model	R-Squared	Adjusted R-Squared	LOOCV-RMSE	Shapiro-Wilk Test P Value	BP Test P Value
Additive	3.221e-01	3.193e-01	1.387e-01	6.373e-13	1.990e-44
AIC From Additive	3.212e-01	3.196e-01	1.386e-01	8.334e-13	8.426e-47
Box	3.367e-01	3.351e-01	1.460e-01	1.949e-07	4.423e-46
Log	3.179e-01	3.162e-01	3.088e-01	6.941e-30	1.329e-32
Interaction	3.310e-01	3.252e-01	1.384e-01	1.013e-13	1.956e-45
Second Order	3.332e-01	3.260e-01	1.384e-01	8.979e-14	1.182e-43
AIC Second Order	3.324e-01	3.288e-01	1.378e-01	1.149e-13	1.063e-49

Model	R-Squared	Adjusted R-Squared	LOOCV-RMSE	Shapiro-Wilk Test P Value	BP Test P Value
AIC Second Order (No Influential)	3.989e-01	3.955e-01	1.201e-01	1.098e-01	2.083e-45

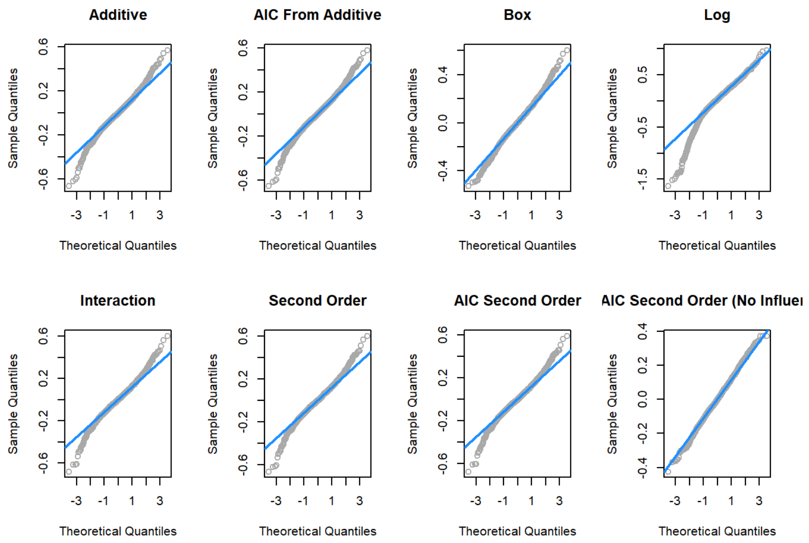
Model Performance Table For Split Data

Model	Test Dataset RMSE	Training Dataset RMSE
Additive	0.1362	0.1405
AIC From Additive	0.1365	0.1402
Box	0.1457	1.0288
Log	0.3116	1.2589
Interaction	0.1348	0.1407
Second Order	0.1344	0.1406
AIC Second Order	0.1349	0.1398
AIC Second Order (No Influential)	0.1173	0.1401

Fitted Versus Residuals Plots



Normal Q-Q Plots



Discussion

For this analysis, our main goal was to build a useful model for predicting “Win-Loss Percentage” in Major League Baseball. We started with a baseline model that included all predictors, and improved by choosing methods that best achieved this goal. At the end of model exploration, we found that the AIC from second order (aic_so_model_lf) model was the most useful for making predictions.

The “aic_so_model_lf” took the following form:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 (x_4^2) + \beta_8 (x_5^2) + \beta_9 x_1 x_3 + \beta_{10} x_2 x_4 + \beta_{11} x_2 x_5 +$$

where

$x_1 = StrikeOuts$

$x_2 = InningsPitched$

$x_3 = NationalLeague(dummyvariable)$

$x_4 = RunsAllowed$

$x_5 = EarnedRunsAllowed$

$x_6 = StolenBasesAllowed$

Before creating our first model, we used the pairs plot function to understand the relationships between all pairs of variables. Based on these plots, we noticed that there was a slight correlation between H and R as well as BA and OBP. This makes sense and is as expected: OBP is equal to BA plus walks plus hit-by-pitches, the most common way to score runs is baseball is by getting base hits. At this point in the analysis, we were not too concerned with correlation, but knew it was something to keep in mind while performing additional analysis.

As stated above, our first approach was to create an additive model with all 10 variables as our starting point. We obtained a small adjusted R^2 of 0.32 as shown in the model performance table. We noticed that the F-test for regression significant was very small. However, when looking at each individual predictor, we saw that a large portion of them were not significant. To further investigate correlation, we used the VIF function to quantify the effect of collinearity on our model. As suspected, many of the predictors had a VIF greater than 5, which further confirmed the issue of multicollinearity. In looking at the fitted versus residuals plot, we noticed that the residuals were roughly centered at 0. This was a good sign as it meant our linearity assumption was not violated. However, we noticed that we had heteroscedasticity in our model. To confirm, we performed the Breusch-Pagan Test which returned a small p-value as shown in the model performance table. To further diagnose this model, we created a Normal Q-Q plot. We noticed that the points were diverging from the line towards the "tails" of the plotted line. We suspected that the errors were not normally distributed. To verify, we performed the Shapiro-Wilk normality test, and the low result value confirmed our suspicion.

Our next approach was to use the step function to perform an AIC backwards stepwise search on our additive model. The goal was to remove variables that were causing collinearity issues such as H and BA. The model that AIC selected no longer contained such variables (H and BA) that were causing coliniarty issues. We also noticed that the Age variable, which did not have any collinearity issue, was removed. Intuitively, this made sense since we did not believe age was a strong indicator for a team's "Win-Loss Percentage". We saw improvements in each individual predictor significance, as well as a small F-test for regression. Our adjusted R^2 increased slightly, but we believed there was still room for improvements. We obtained a LOOCV RMSE of 0.1386 which was slightly better than our simple additive model, so we preferred the AIC model.

We believed that adding interaction to all the predictors would make the AIC model more flexible. As you can see from the model performance table, we obtained a slightly increased LOOCV RMSE score. However, the adjusted R^2 remained the same. To improve the adjusted R^2, we tried to introduce quadratic terms to the following variables: SO, IP, Lg, R, ERA and SB. We were able to slightly improve the adjusted R^2 to 0.328. Since we did not want to overfit our model, we did not introduce any higher order terms. Seeing that quadratic terms yield the best results for both adjusted R^2 and LOOCV RMSE, we decided to use AIC backward search again. The aim was to select a model with a smaller number of parameters and a good fit.

Our new model that was chosen by AIC (aic_so_model_lf) was the most useful for making predictions. This model has the lowest LOOCV RMSE and the highest adjusted R^2 (note that this adjusted R^2 value is still rather small). As you can see from the model performance table, when we removed influential points, our new model passed the normality test at a alpha = 0.05 level. Unfortunately, the test for homoscedasticity still failed at the same alpha level. The fitted versus residuals plot, as shown in the results section, appears to display a larger variance at the center of the plot and a smaller variance at either ends. We had hoped that our model would do a better job at explaining the data. However, based on our low adjusted R^2 and heteroscedasticity, we do not believe this model is a good candidate for making inference. Note, we compared the additive model RMSE with our chosen model and noticed that the results were very close. Generally, we would have preferred the smaller model, but since none of our models are useful for making inference, we decided to go with our more flexible model.

In summary, the analysis in this report is driven towards the goal of finding a "best" model for predicting pitching "Win-Loss Percentage" in Major League Baseball. In performing the analysis, models were explored, evaluated, and then improved such that model performance metrics improved and the prediction models became "better". Because the primary goal of the modeling was to find a model that is best at predicting, in the analysis process, model assumptions and variable relationships defer to the goal of minimizing the error. The model that gives highest confidence in prediction is the model that makes the smallest amount of error, and emphasis is given to analysis that supports this case (such as LOOCV-RMSE).

After answering the initial question posed in this report and finding a model that is "best" for predicting "Win-Loss-percentage", consideration must be given towards determining how useful the "best" prediction model will be. To do so, the model user must first pose some questions to themselves. Is the user comfortable removing observations of high influence? Doing so may give a data set that is slightly less representative than the original, but may also improve model assumptions. Is the user of the models a general manager of a Major League Baseball team trying to decide whether to execute a trade for a pitcher that may "make-or-break" the season? If so, the models might not yet be "useful", and the manager may desire to continue to improve models and further reduce error. Note that modeling was based on only ten "initial" predictors. There exist many more baseball statistics that have not been considered in modeling, and in this case, the user might attempt to further reduce error by adding additional predictors into the model. On the other hand, the model user may be a casual baseball fan looking to use the derived models to find a higher level prediction of their favorite team's "win-loss" record for an upcoming season. In this case, the models might be more than sufficient in accomplishing this, and already provide immediate use to the user. While the model analysis is successful in providing a “best” predicting model relative to other predictive models, ultimately, the question of usefulness is best defined by each specific user on a case by case basis.

Appendix

No additional code has been included with the report.