Data Mining Final

Akhil Jonnalagadda (saj2538), Isaac Hulsey (idh285) & Tristan Robins (tjr2695)

5/8/2020

Abstract

The goal of this project is to assess the effectiveness of different prediction models on baseball data through the scope of minimizing out-of-sample RMSE. The end goal is to create the best performing model in terms of minimized RMSE for predicting strikeouts. The data collected was webscraped from MLB.com and contains all pitching stats on every pitcher between the years of 1990 until 2019 regular season games—this excludes spring training, the All-Star game, playoffs, and the World Series. We begin using a PCA for exploratory data analysis to get an idea of what sort of trends there are in the data and branch out into prediction using random forest, lasso, and OLS regression. We conclude the project by comparing the RMSE of all the regressions run.

Baseball

Baseball is a sport that has an abundance of metrics that can be assessed from a data mining perspective. As such, modeling the data is a unique and intriguing challenge—is it possible to find a model that most accurately describes the most pertinent metrics for baseball's pitchers? Further, is it possible to broadly characterize the best players through utilization of statistical modeling techniques?

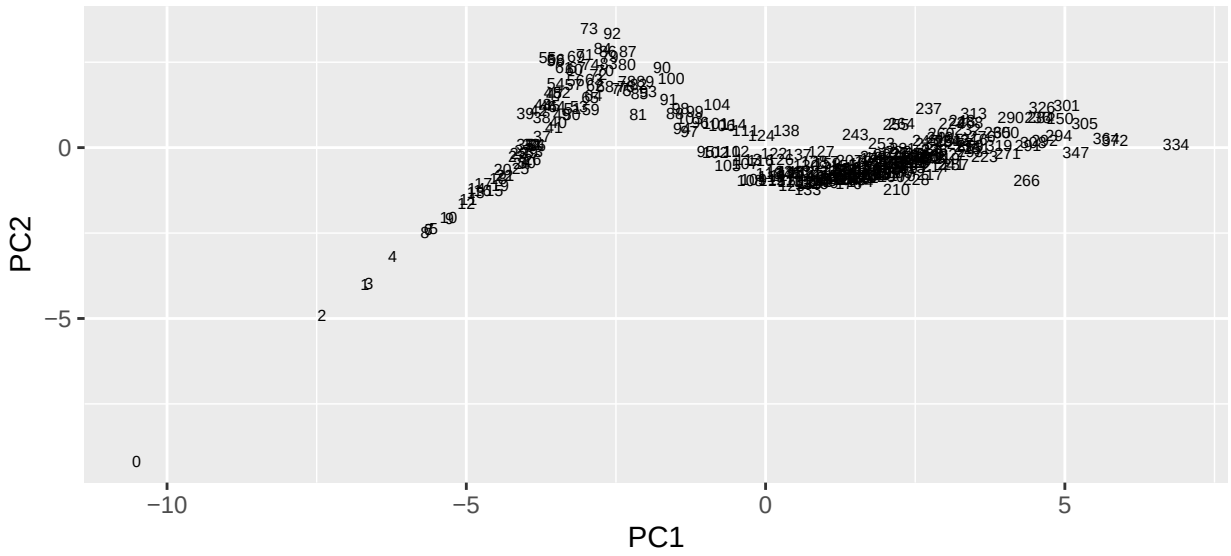
By breaking down baseball's most widely discussed defensive position, pitchers, we are able to better understand the metrics that define greatness through analysis of a pitcher's strikeouts (K's) and walks & hits per inning (whip).

Pitching

Pitchers need to limit earned runs—this means getting many strikeouts or preventing hits and walks. As such, pitchers with a high count of strikeouts are perceived as better and pitchers with low whips are seen as especially elite. To begin, examination of PCA was utilized to characterize the features of the best pitchers.

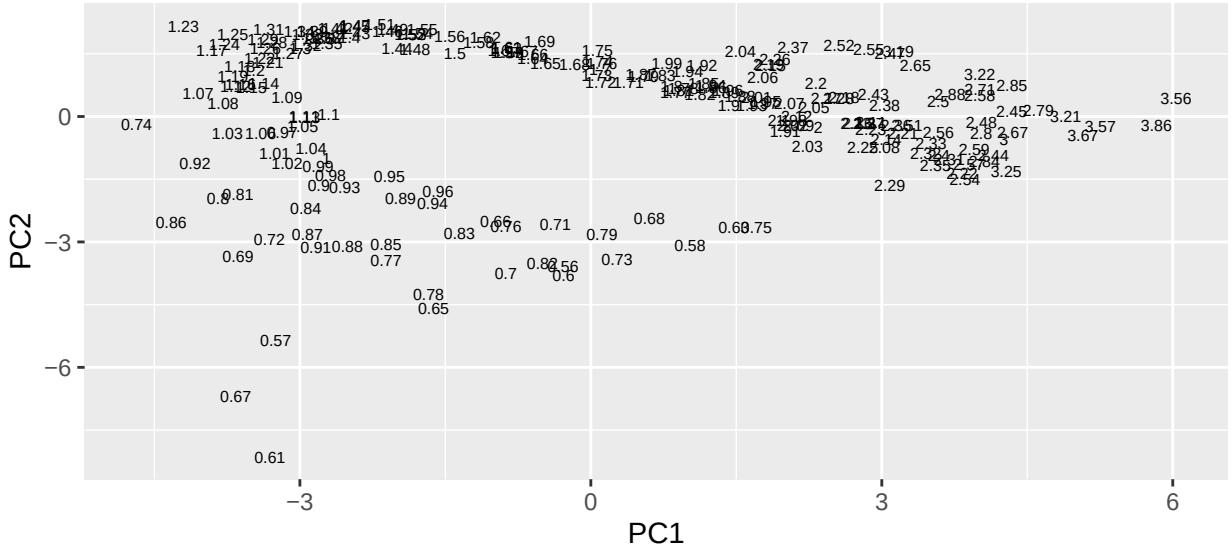
Principal Components Analysis

First, PCA on strikeouts will be examined. Ideally, it would be possible to distinguish between high and low strikeout counts—if so, there is confidence that principal component analysis works to sort bad pitchers from great pitchers.



The PCA of strikeouts showcases an increasing trend in the axis of (PC1, PC2); there is a trend that the best pitchers have PC1 values greater than 1. The the worst pitchers have PC1 values less than zero. This is a visually good model, but isn't perfect. Ideally, there would be a higher degree of sorting on PC2 between, but it's still clear where the best pitchers sort. Despite the lack of perfection, the average great pitcher contains many of the same traits, according to PC1. Although PC2 isn't a great variable for sorting, the best pitchers seem to cluster around 0 (while retaining the characteristics from PC1). Using PC1 and PC2 accounts for 77.71% of the variation in strikeouts [see Appendix PCA Strikeouts Breakdown].

Although it is possible to sort on strikeouts , it isn't very easy to idnetify the best pitchers using two PC's. To get a better feel for whether this is always the case, a pitcher's whip was also assessed using PCA.



This PCA very clearly lays out the best and worst features from the first two principal components (which comprise 72.68% of the variation in whip) [see Appendix PCA Whip Breakdown]. The best—nay, elite—pitchers have whips with values less than one. We clearly see that under these principal components, these kinds of pitchers tend to have PC1 and PC2 values that are negative. Specifically, PC1 on average seems to be around -3 while PC2 is around -2.5. For great pitchers (with whip values less than 1.5), PC1 is in the range [-6, -1.5] and PC2 values are in the range of [-7, 2.5]. When these happen simulataneously, great pitchers are found.

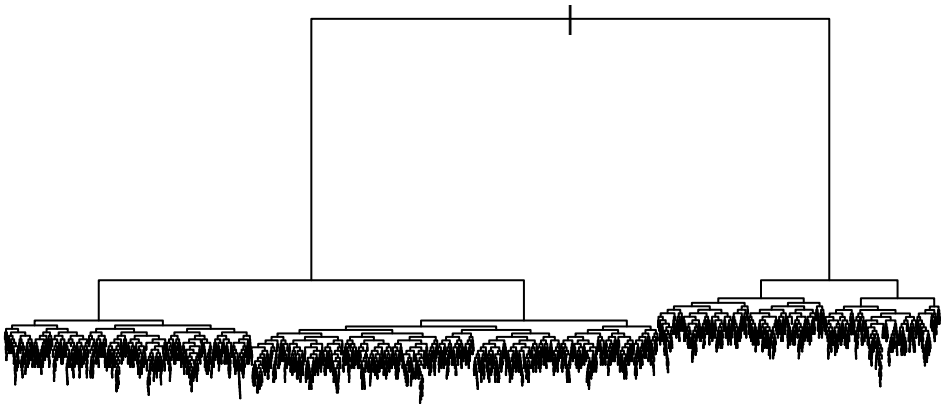
Clearly it is possible to sort and filter the best traits using PCA which tells us that certain variables may be

more significant than others (and that there is some dependence between variables), but is it possible to find an accurate model that predicts strikeouts from those variables? To do this, random forests allow for the aggregation of important traits to predict outcomes, and lasso can select the best metrics that can then be sorted into an OLS output.

Random Forest

Random forests are resourceful for aggregating many trees to develop a best model. To best understand how this is done, an initial tree can be examined.

Tree

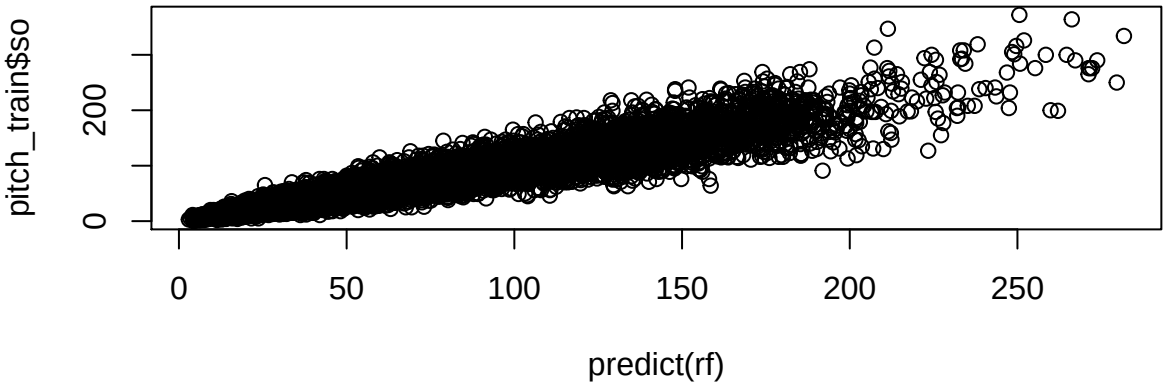


This tree is just one possibility of how the data of pitchers could be presented and sorted. The RMSE for the single tree is:

```
## [1] 24.32252
```

Pruning the tree could yield an even better (lower) RMSE, but a random forest should also generate a better model. When trees are randomly aggregated many times, the best outcomes are given more significance which results in a better model. This process creates a 'random forest' wherein the model is able to be estimated through every tree produced.

Random Forest



This random forest demonstrates the trends from 500 randomly generated trees. The resulting RMSE is:

```
## [1] 16.85469
```

Clearly, the random forest outperforms the tree due to the added variation across multiple random trees. This is a fairly good model that is easily able to be produced. It's possible that using lasso and OLS may yield less error due to less randomness in choosing varibales like trees do.

Lasso and OLS

```
##   (Intercept)          wins        losses           era             g
##  1.612227e+01  7.572497e-01 -4.422773e-01 -5.909468e-01  1.152493e-01
##            gs           svo            ip             h             r
##  7.655325e-02  2.618235e-01 -4.059244e-05 -1.227878e+00 -3.994056e-01
##            er            hr            bb           avg          whip
##  3.714465e-01  2.171921e-01 -4.546473e-01 -2.373033e+02  1.146403e+01
##            cg           sho            hb           ibb            gf
## -4.952045e-01 -1.036509e+00 -2.676411e-01 -7.891112e-01 -1.032084e-01
##            hl          gidp            go            ao            wp
##  2.647419e-02  1.920584e-01 -1.805267e-01 -1.282781e-01  8.641692e-01
##            sb            cs           tbf            np         go_ao
##  6.109450e-01  1.267580e+00  4.855258e-01  1.884884e-02 -2.584483e-02
##           obp           slg           ops           h_9          p_ip
## -6.960309e+01 -1.539090e+01  3.703404e+01  5.987068e+00 -1.118160e+00
##          wpct
## -7.810755e+00
```

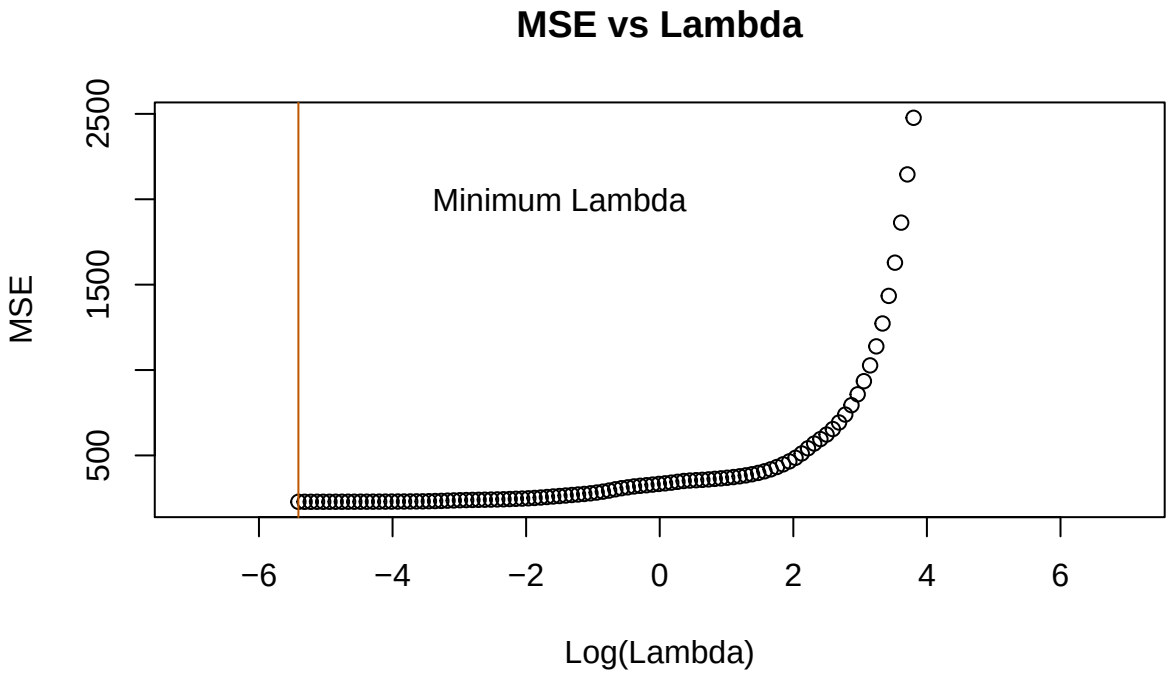
```
## [1] -5.409597
```

```
## [1] 0.004473443
```

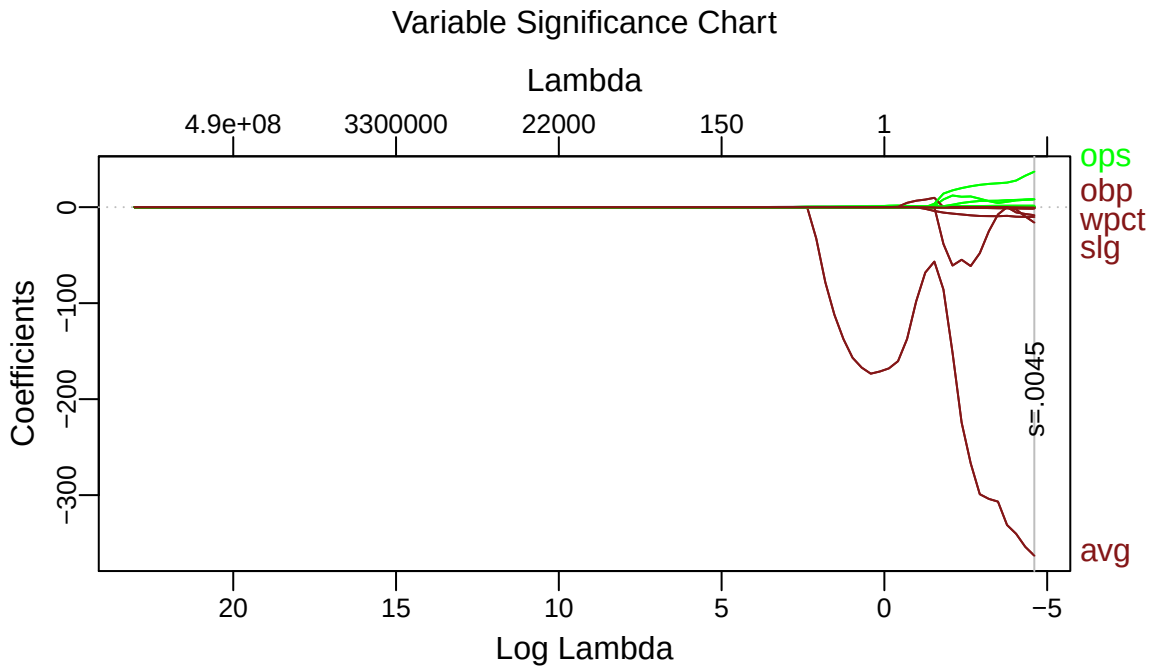
Here we are looking at creating a model to predict strikeouts. Simply put, the goal of a pitcher is to strike the batter out in baseball. Modeling this is useful for teams to understand a pitcher's ability to do their job.

Before any analysis we had to clean and address the data. This included removing all categorical variables and non trend specific data such as ID numbers. After this all metrics using strikeouts in their calculations were removed. The remaining variables for these models included those such as home runs, saves (pitcher ending a winning game), and an assortment of others.

The first method used was a lasso (least absolute shrinkage and selection operator) to predict strikeouts. First we had to create training data and testing data. Using an 80/20 split on our sample that was done. The next part of this method including creating a matrix of all variable combinations and using a feature section method to choose the most significant variables to use in our analysis. This part also included finding the optimal "lambda" our penalty vector in this regression.



In this graph we see across these lambda values we plot the variables we selected and their respective coefficients. As not all metrics are the same we highlighted the 5 most significant according to our section model. The red being for negative impact and green for postie. OPS, the players on base and slugging average is the most significant positive variable and batting average being the most negative significant variable.



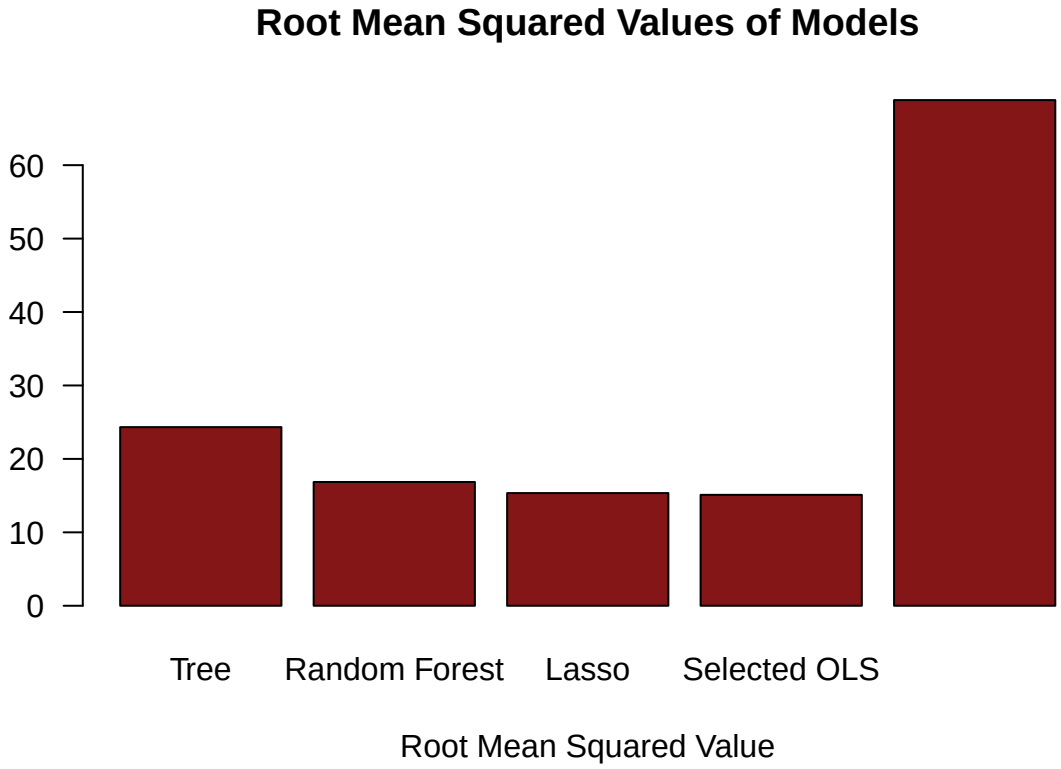
The next part of this method was to lock down the lambda. This was done by plotting the mean squared error and a series of lambda values to find the lowest mean squared error. As seen about the lambda was .0044845 or 5.407 in absolute value logarithmic form.

Using these variables and lambda we can compute our lasso regression. Using this model and the test data we compare our predictions to our actual outcomes to find out error rate. The root mean squared error (RMSE) was 15.46 for this model. Considering that the min/max of strikeouts is 0/372 this is an understandable error

rate.

For a comparison of another regression technique we used our selected variables in an OLS. Using the same prediction procedure we calculated a 15.10375 RMSE. Again a fair error rate for the model and very similar to our lasso regression error.

While these regressions seem to be doing a decent job at estimating strikeouts we need to compare against some standard to understand that its a "better" method in some sense. The simplest way was to run OLS with all variables (post data clean). OLS is the simplest regression technique and the go to in most economists toolbox. Running the prediction and comparing to our test sample we had a RMSE of 69.4515. Which is far worse than our past error rates. Now this begs the question of why OLS is so much worse without feature section. Simply put is that OLS without section control will over fit the model. Our in sample error rate was 14.92575 which is the best error rate of this group of models. In practice(test data) we see this does not hold as we over fit our model. Some variables had more influence than they should have creating increased error rates. The section method in our LASSO then OLS avoids this issue. In summary, these methods are "better" than our dry cut OLS.

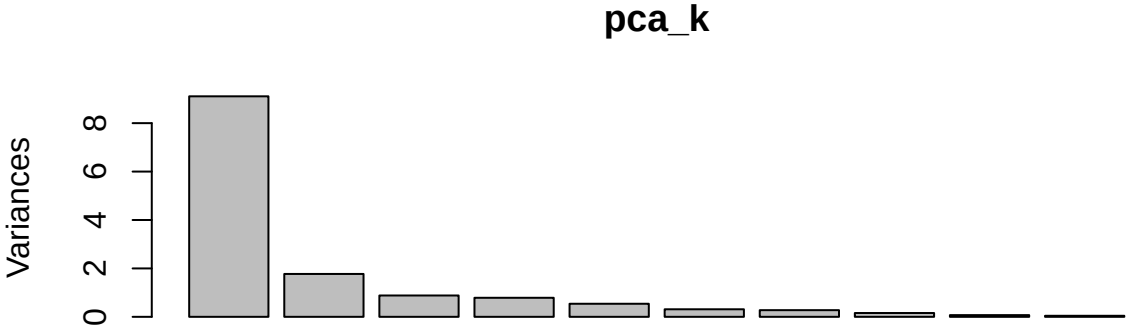


Conclusion

The results in the tree and random forest are initially impressive. The discrepancy between these two results from the random forest running many more regressions than the singular tree and is a stochastic result. The tree and random forest are comparable since the random forest is an aggregation of trees. Further, running a standardized lasso regression was utilized for feature selection and, as expected, its results are very similar to the OLS regression with the same features selected. Running all of the variables in the final OLS (a 'kitchen sink regression') returns the worst results which is not surprising. The reason why random forest and the tree marginally underperform the OLS type regressions probably has to do with the trees and random forests overfitting noise in the data. The data ran through the lasso only had first order interaction terms and could be missing out on quite a lot of the geometry of the data. Overall, the big surprise was that the OLS selected feature regression minimized RMSE with very little work done with feature engineering.

Appendix

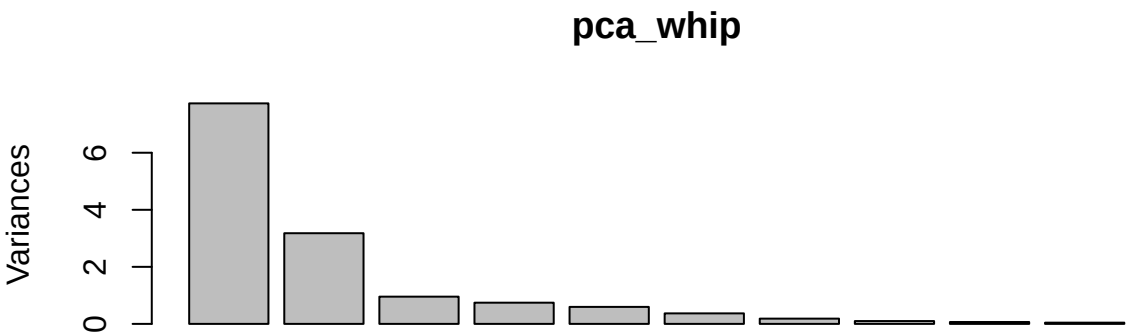
PCA Strikeouts Breakdown




```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     3.0180 1.3310 0.9384 0.8870 0.73441 0.55834 0.52445
## Proportion of Variance 0.6506 0.1265 0.0629 0.0562 0.03853 0.02227 0.01965
## Cumulative Proportion  0.6506 0.7771 0.8400 0.8962 0.93475 0.95701 0.97666
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.39751 0.26554 0.20645 0.17752 0.12596 0.08963 0.01408
## Proportion of Variance 0.01129 0.00504 0.00304 0.00225 0.00113 0.00057 0.00001
## Cumulative Proportion  0.98795 0.99298 0.99603 0.99828 0.99941 0.99999 1.00000
```

```
##        PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
## so    0.32 -0.01  0.06 -0.10  0.03 -0.11  0.23 -0.23
## wpct  0.27 -0.03  0.36 -0.32  0.11 -0.23  0.65  0.29
## era  -0.27 -0.38 -0.02  0.15  0.14 -0.12  0.21 -0.24
## sv   -0.17  0.56 -0.14  0.22  0.34 -0.22  0.23 -0.04
## ip    0.32 -0.04 -0.13  0.08  0.02 -0.20 -0.01  0.04
## hr    0.28 -0.06 -0.37  0.26 -0.18 -0.35  0.10 -0.49
## bb    0.28 -0.06 -0.33  0.23  0.25 -0.04 -0.18  0.63
## bk    0.18  0.05  0.46  0.77 -0.20  0.29  0.17  0.04
## tbf   0.32 -0.04 -0.18  0.12  0.04 -0.20 -0.05  0.08
## obp  -0.28 -0.33 -0.09  0.18  0.25 -0.01  0.04  0.13
## sho   0.24 -0.05  0.48  0.00  0.59 -0.16 -0.48 -0.27
## hb    0.26 -0.03 -0.30 -0.08  0.41  0.72  0.25 -0.23
## gf   -0.23  0.52 -0.03  0.09  0.22 -0.07  0.08 -0.03
## whip -0.27 -0.37 -0.02  0.18  0.29 -0.16  0.23  0.04
```

PCA Whip Breakdown




```
## Importance of components:
```

```
##                            PC1    PC2     PC3    PC4    PC5     PC6     PC7
## Standard deviation     2.7804 1.7829 0.97708 0.8622 0.7714 0.60670 0.42955
## Proportion of Variance 0.5522 0.2271 0.06819 0.0531 0.0425 0.02629 0.01318
## Cumulative Proportion  0.5522 0.7792 0.84743 0.9005 0.9430 0.96932 0.98250
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.3153 0.25636 0.20662 0.12994 0.10796 0.09207 0.01469
## Proportion of Variance 0.0071 0.00469 0.00305 0.00121 0.00083 0.00061 0.00002
## Cumulative Proportion  0.9896 0.99429 0.99734 0.99855 0.99938 0.99998 1.00000
```

```
##        PC1   PC2   PC3   PC4   PC5   PC6   PC7
## whip  0.31  0.20 -0.04  0.13 -0.41  0.13 -0.09
## wpct -0.21 -0.31  0.32 -0.16 -0.20  0.77  0.11
## sv   -0.17 -0.40 -0.39  0.21 -0.33 -0.16 -0.04
## ip   -0.34  0.14  0.01  0.09 -0.06  0.14 -0.19
## hr   -0.26  0.35 -0.07  0.16  0.05  0.06 -0.30
## so   -0.35 -0.02  0.06  0.11 -0.16  0.05 -0.06
## bb   -0.22  0.41 -0.19  0.11  0.13  0.09 -0.13
## tbf  -0.33  0.21 -0.02  0.11 -0.04  0.14 -0.19
## obp   0.30  0.26 -0.06  0.13 -0.32  0.10 -0.04
## sho  -0.22  0.01  0.66 -0.05 -0.39 -0.51 -0.15
## hb   -0.27  0.27  0.00  0.22 -0.09 -0.08  0.88
## gf   -0.21 -0.37 -0.39  0.19 -0.23 -0.10 -0.04
## era   0.30  0.16  0.01  0.22 -0.45  0.13  0.02
## bk   -0.14  0.22 -0.34 -0.83 -0.34 -0.07  0.05
```