Expect to Win:

Predicting Winning Percentage in Major League Baseball

Rob Sanchez &
Sydney Brougham

STA631
F2018

Introduction

In this report we discuss the analysis of season-by season team data for professional Major League Baseball (MLB) teams. Using a variety of statistical procedures, including linear regression and penalized regression, in addition to some non-linear, dimensionality-reducing and distribution-fitting strategies, we modeled MLB teams' main playing season performance metric: winning percentage[1], the number of games won divided by the number of games played.

Ultimately, professional sports teams are in the business of making money. In order to stay relevant and maintain predictable levels of revenue, the primary goal of any professional team is to win games. Team owners, managers, players, fans and bookmakers all have an interest in a team's winning percentage throughout the season: player trades, gameday personnel decisions, casino odds and general fan interest all hinge on how well a team is performing and how it may be expected to perform in the future.

This project was inspired by analyses by Marchi & Albert (2014) and Miller (2005), which explored the predictive power of a non-linear equation called the Pythagorean Won-Loss formula or Pythagorean Expectation (PE). In that context, our project goals were three-fold: (1) construct the best linear model for winning percentage using the features in the Teams dataset, (2) estimate the exponent in the PE formula to best fit the data, and (3) produce visualizations of Weibull distribution fits against runs-scored and runs-allowed histogram data to further support the theory behind PE.

Data and Tools

The bulk of the data we analyzed were obtained from Sean Lahman's Baseball Database, a freely-available and open-source collection of complete batting and pitching statistics from 1871 to the present. Our analysis focused on one table of the Lahman database, the Teams table, which is detailed in Figure 1. Each Teams table observation is for a specific baseball season and for a specific team. Various offensive and defensive statistics for each team are collected in each column of the table. The number of teams, their names, and their locations have changed significantly since 1871, so this table provides a convenient look at the team-by-team and season-by-season history of Major League Baseball. Due to missing data, regression analyses focused on data since 1970, which was made up of 1324 season-team observations. Linear and non-linear Pythagorean expectation estimates were completed on all data since 1871, made up of 2865 observations. Season-specific team data detailing the frequency of runs scored and runs allowed for 2004 used in the Weibull fits were obtained from *baseball-reference.com*, an open-access website specializing in baseball statistics and analytics.

The procedures used for this project were implemented in three different programming languages: SAS, R, and Julia. SAS was used for variable selection and penalized regression, R was used for principal components regression (PCR) and partial least squares (PLS) regression, and Julia was used for the Weibull distribution fits.

Variable Selection Models

The first approach to modeling winning percentage was linear regression, using any number of the supplied variables from the Teams table (Figure 1). The initial phases were largely investigatory. There were originally upwards of 29 variables describing aspects of both a team's offensive and defensive capacity, so naturally the first order of business was to determine which of these variables were most useful in predicting winning percentage. As summarized in Figure 2, we applied three variable selection methods; backward, forward, and stepwise; in addition to more complex regression models; lasso, elastic net, principle component regression (PCR), and partial

[1] We recognize that it is more mathematically accurate to refer to this metric as a winning *proportion*, since as listed in the MLB standings it is never multiplied by 100 to produce a percentage. However, the statistic has been referred to as "winning percentage" throughout baseball history, and so we will refer to it that way here as well.

least squares (PLS) regression. These methods were applied on all data from 1970 onward (due to missing data before then) and used 5-fold cross validation to train (2/3 split train/test) the data.

yearID	Year	ERA	Earned run average
lgID	League	CG	Complete games
teamID	Team	SHO	Shutouts
franchID	Franchise	SV	Saves
divID	Team's division	IPOuts	Outs Pitched
Rank	Position in final standings	HA	Hits allowed
G	Games played	HRA	Homeruns allowed
GHome	Games played at home	BBA	Walks allowed
W	Wins	SOA	Strikeouts by pitchers
L	Losses	E	Errors
DivWin	Division Winner (Y or N)	DP	Double Plays
WCWin	Wild Card Winner (Y or N)	FP	Fielding percentage
LgWin	League Champion(Y or N)	name	Team's full name
WSWin	World Series Winner (Y or N)	park	Name of team's home ballpark
R	Runs scored	attendance	Home attendance total
AB	At bats	BPF	Three-year park factor for batters
H	Hits by batters	PPF	Three-year park factor for pitchers
2B	Doubles	teamIDBR	Team ID used by Baseball Reference
3B	Triples	teamIDlahman45	Team ID used in Lahman database 4.5
HR	Homeruns by batters	teamIDretro	Team ID used by Retrosheet
BB	Walks by batters		
SO	Strikeouts by batters		
SB	Stolen bases	calculated:	
CS	Caught stealing	run_diff	Run differential (R - RA)
HBP	Batters hit by pitch	SLG	Slugging
SF	Sacrifice flies	OBP	On-base percentage
RA	Opponents runs scored	OPS	On-base plus slugging
ER	Earned runs allowed	1B	Singles

Figure 1. Teams table variables from Lahman's baseball database. Features in grey were included in regression models, those in yellow are relevant to the Pythagorean Expectation (PE)

For every applied selection and regression technique, the best way to compare accuracy is to refer back to the least squares regression model that has all 29 variables included. As expected, this full model results in the highest adjusted R-squared value and the lowest test root mean square error (RMSE), 0.9231 and 0.02008 respectively. The three selection methods each produced models that eliminated 10 or more variables, yet adjusted R-squared only decreased by a maximum of 0.0012 and RMSE only increased by a maximum of 1.8%. The three models were also highly similar in variable composition, and from the variables that they all share we have a rough idea of the features that are most useful in predicting winning percentage. These variables include but are not necessarily limited to: runs, at bats, walks by batters, stolen bases, caught stealing, on-base percentage, hit by pitch, runs allowed, complete games, earned run average, shutouts, saves, and outs pitched. These are the areas where teams should begin targeting their efforts to improve performance if they are not meeting expectations.

Both lasso and elastic net are penalized regression methods that were only included in this model-building phase for their potential to eliminate variables. Lasso eliminated 18 variables in total, the most thus far, while only decreasing adjusted r-squared by 0.0198 increasing test RMSE by 8.3%. Of the variables that remained in the model, the majority aligned with the variables chosen by the selection methods discussed previously. Elastic net, by comparison, performed the worst. This is because it eliminated the fewest variables out of all the methods (9), and it resulted in the highest test RMSE and the lowest adjusted r-squared.

Principal Components and Partial Least Squares Regression

The *pls()* package for R was used to perform principal components (PCR) and partial least squares regression (PLSR). Since our overall goal for modeling was to select variables and generally reduce the dimensionality of the dataset, we thought PCR and PLSR might be useful for this purpose.

	Least Squares	**Backward**	**Forward**	**Stepwise**	**Lasso**	**Elastic Net**
	(all)	R	R	R	R	R
		SB	SB	SB	SB	SB
		OBP	OBP	OBP	OBP	OBP
			SLG	SLG	SLG	SLG
		BB	BB	BB		BB
		CS	CS	CS		
		AB	AB	AB		
		HBP	HBP	HBP		
		1B				1B
		2B	2B			
		3B			HR	HR
						SO
		RA	RA	RA	RA	RA
		CG	CG	CG	CG	CG
		ERA	ERA	ERA	ERA	ERA
		SHO	SHO	SHO	SHO	SHO
		SV	SV	SV	SV	SV
		IPouts	IPouts	IPouts		BBA
		BBA	E	E	BBA	E
		SOA				SF
		FP				FP
						HA
						HRA
						ER
Num Vars:	29	19	16	15	11	20

Test RMSE:	0.02008	0.02017	0.02045	0.02045	0.02175	0.02346
Adj Rsq:	0.9231	0.9229	0.9221	0.9219	0.9033	0.9032

Figure 2. Variable selection summary. Selected variables are shown, with offensive and defensive features highlighted in blue and gray, respectively. Test RMSE and adjusted r-squared are listed for each model.

A random training/test split of 2/3 and 1/3 respectively was used to separate the data, and both the PCR and PLSR models were trained using cross-validation. After examining the summary of each trained model, we found the smallest number of components that explained 85% of the variance in the response. For PCR, this was possible with just seven components. For PLSR just two components were enough. Quite the reduction in dimensions indeed!

We chose the 85% threshold level because, as we will explain below, a model predicting winning percentage with only 1 variable, run-differential (RD = runs scored – runs allowed), was good enough to produce an adjusted R-squared value of around 0.86. We thought a comparison here might be instructive.

Validation plots for PCR and PLSR can be found in Appendix A. In Figure 3 below, we show the component loadings for PLSR. While it is true that the loading matrix is not strictly speaking interpretable, it does highlight some interesting properties of the data. The matrix is highlighted at a feature contribution threshold level of 0.3. The bright-colored cells indicate loadings greater than the absolute value of the threshold level, and the dull-colored cells indicate loadings that are near but just below the absolute value of the threshold. The loadings can be examined generally for contributions from the various features present in the dataset, but to our eyes there is one that stands out: the first one, at the top of the figure, run-differential, or RD.

In the PLSR case, the very first component accounts for 84.2% of the variance in the response, with the largest contribution again coming from RD at 0.5021 in the 1st component. In terms of predictive performance, this dimensionally-reduced model achieves RMSE on the test data of 0.0245 with just two components. This is slightly worse but on-par with the penalized elastic net model identified in the previous section. A similar loading matrix can be found in Appendix D for PCR. The PCR model achieves an RMSE of 0.0258 with seven components. In that model, a significant jump in variance explained of the response can be at least partly attributed to RD between the 2nd and 3rd components.

The importance of RD in this dataset cannot be overstated. In order to win games, a team must score runs! RD is computed as the difference between R (runs scored) and RA (runs allowed), variables that were chosen for every single variable selection scenario in the previous section. In fact, RD is such a good predictor for winning percentage that for the rest of this report we will focus on it almost exclusively, particularly in the context of PE, the Pythagorean Expectation of winning percentage.

PLS Component Loadings, MLB since 1970

X	13.8	33.8	51.9	62.4	69.2	72.4	75.2	78.4	81.3
Wpct	84.2	87.1	88.3	89.4	89.9	90.3	90.6	90.8	91.0
	Comp 1	Comp 2	Comp 3	Comp 4	Comp 5	Comp 6	Comp 7	Comp 8	Comp 9
RD	0.5021	0.0913	0.0559	0.0316	0.0109	-0.0093	0.0122	-0.0229	0.0712
AB	0.0193	-0.0540	0.2604	-0.4621	0.4178	-0.1571	0.0375	-0.2675	0.0524
X1B	0.0164	0.1957	0.2123	-0.4186	0.2388	0.3258	-0.0312	-0.4179	0.0333
X2B	0.1533	-0.2523	0.2845	-0.2609	-0.0519	-0.0220	-0.3160	0.1546	0.2004
X3B	-0.0092	0.1871	0.0120	-0.2150	0.0798	0.4348	-0.5509	0.3224	0.0676
HR	0.1682	-0.2252	0.4001	0.0970	0.0254	-0.2691	0.0968	-0.1483	-0.0451
SF	0.1572	0.1067	0.1218	-0.4098	-0.2228	0.3066	0.0131	0.1502	-0.1150
BB	0.1940	0.0461	0.2583	-0.2492	-0.1371	-0.5524	0.5269	0.0658	-0.1390
HBP	0.1022	-0.3208	0.1741	-0.0739	0.0365	0.0917	0.1309	0.0863	-0.1306
SO	0.0421	-0.3805	0.0855	-0.1001	0.2477	-0.4015	0.1548	0.4199	-0.0206
SB	0.0468	0.1579	-0.0230	-0.1751	0.1224	0.1828	-0.2499	0.9472	-0.6396
CS	-0.0848	0.3029	-0.0226	-0.1334	0.1004	-0.0673	-0.3733	0.5394	-0.4194
OBP	0.2639	0.0416	0.3736	-0.0726	-0.3630	0.1286	0.0759	0.0124	-0.0462
SLG	0.2243	-0.1601	0.4236	0.0737	-0.1849	0.0973	-0.1943	-0.0071	0.0372
ER	-0.2543	-0.1977	0.3901	-0.2159	-0.0699	0.1289	-0.0025	0.0422	-0.0176
ERA	-0.2826	-0.1993	0.3425	-0.0342	-0.2752	0.2540	-0.0452	0.1409	-0.0259
CG	-0.0248	0.3905	-0.0967	-0.0258	0.0248	0.1170	0.4898	-0.3039	0.2843
SHO	0.2544	0.0841	-0.2817	-0.1357	0.1856	-0.1827	0.1699	-0.2953	0.1988
SV	0.2632	-0.1436	0.1375	0.0205	0.3843	0.1178	-0.2155	0.4299	-0.1746
HA	-0.2042	-0.0744	0.3568	-0.3642	0.1362	-0.0296	-0.0364	0.0789	0.2214
HRA	-0.0859	-0.3074	0.3360	-0.0861	-0.0377	0.1212	0.0644	-0.0069	-0.1369
BBA	-0.2480	-0.0046	0.2683	-0.1776	0.1280	0.0104	0.3070	-0.2202	-0.2683
SOA	0.1572	-0.3593	0.0340	-0.1397	0.2510	-0.2270	0.2311	0.0895	-0.1325
IPouts	0.0291	-0.0249	0.1752	-0.4688	0.4829	-0.2659	0.1125	-0.2012	0.0198
E	-0.2224	0.3455	0.0733	-0.0778	0.1820	-0.5146	0.1648	0.1742	-0.0362
DP	-0.0887	0.0492	0.2594	-0.1882	0.1656	-0.0170	-0.2849	-0.2834	0.5994
FP	0.2384	-0.3491	-0.0262	-0.0615	-0.0456	0.4568	-0.1748	-0.2686	0.0859
	Highlight Loadings with absolute value greater than:					**RMSE**	**0.0245**		
	Threshold	**0.30**					**2 comps**		

Figure 3. PLSR component loadings

Run Differential

Straying into more complex and less interpretable models is largely unnecessary. Even reducing the number of predictors in the model by one half, as in the case of forward and stepwise selection, there was minimal loss of predictive power. In fact, there is a linear combination of just two variables, known as run differential, which can describe 86.3% of the variation in winning percentage on its own (Figure 4). Run differential is the number of runs scored minus the number of runs allowed. Not only is such a model very simple, it is also easy to interpret. It is no stretch to say that run differential is useful in predicting winning percentage, because a comparison of runs to runs allowed is ultimately what determines the winner of any one given game. Note that the intercept is precisely 0.5000, suggesting that if a team scores as many runs as they allow, then they are expected to win half of their games. The slope can even be manipulated to indicate how many additional games out of the typical 162 a team could expect to win if the run differential were increased by a specified amount. This is consistent with Marchi and Albert's findings, where they also assert that run differential is a strong predictor of winning percentage (2014).

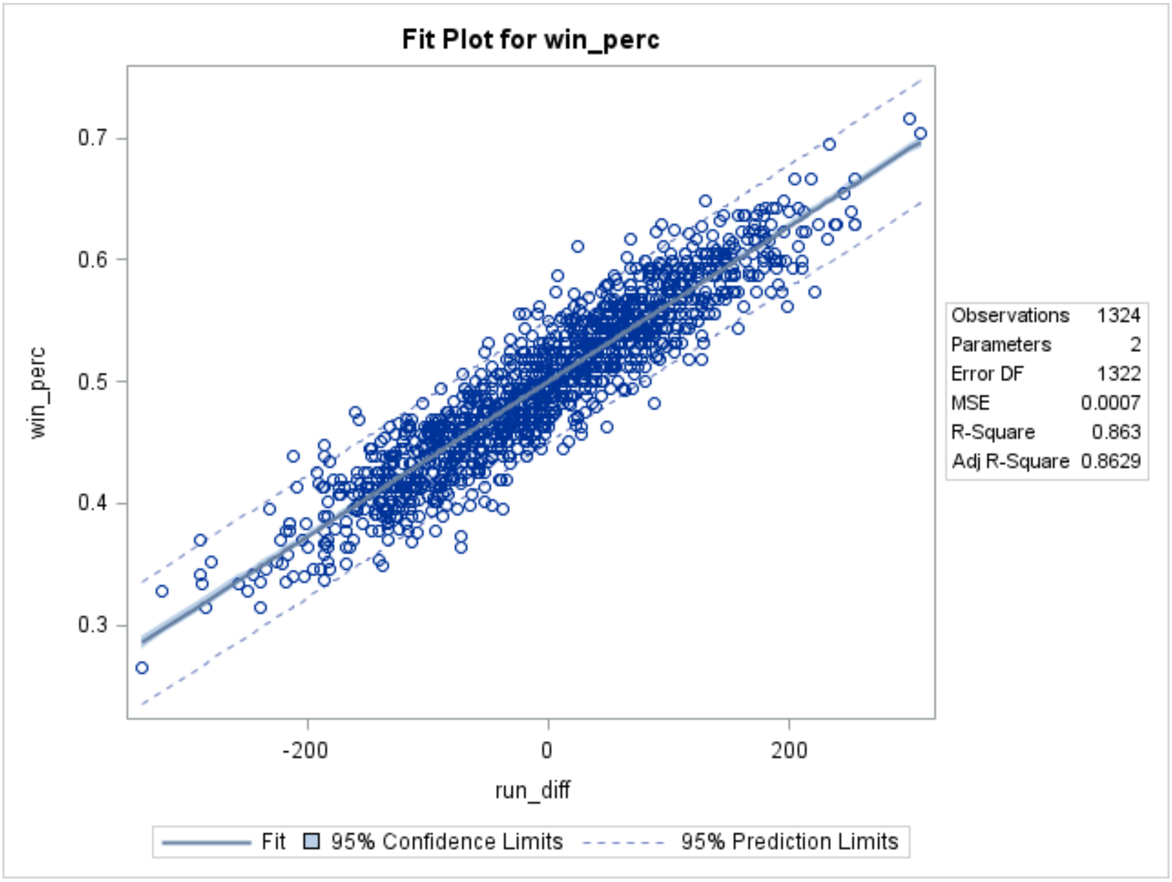


Figure 4. Winning percentage as a function of run differential.

Pythagorean Exponent Estimation

The original Pythagorean Win-Loss formula was derived empirically by famed baseball statistician Bill James in the 1980s (Marchi & Albert, 2014). James's original formulation is based solely on runs scored (RS) and runs allowed (RA) and due to the exponent value of 2 he initially chose, was nicknamed the so-called "Pythagorean Expectation" (PE) formula. It was originally specified as follows:

$$W\% = \frac{RS^k}{RS^k + RA^k} = \frac{RS^2}{RS^2 + RA^2}$$

More recent research (Marchi & Albert, 2014) has revealed that the actual exponent in the MLB context is closer to the 1.8-1.9 range. Our goal was to estimate this exponent with our dataset, and we took two approaches to doing so: simple linear regression and nonlinear regression. In order to use simple linear regression to estimate the PE exponent, the PE formula was rearranged as follows:

$$log\left(\frac{W}{L}\right) = k \cdot log\left(\frac{R}{RA}\right)$$

With the equation in this form, it is plain to see that in a model where the dependent variable is the log of the win-loss ratio and the independent variable is the log of the runs to runs allowed ratio, the estimate for the slope would represent the PE exponent. Note that the model requires the intercept to be zero. Running this model using all available observations in the dataset results in an exponent estimate of 1.8775 (95% CI: 1.8554 to 1.8996).

The second method for estimating the exponent was a nonlinear regression. The SAS procedure, `proc nlin`, is known for its utility in very specific scenarios where certain variable relationships are defined as part of a known mathematical function, and the goal is to converge upon some unknown constant(s) to best complete the model. In our case, using this procedure was a matter of inputting the unaltered PE expression, and initializing the exponent parameter with a value of two. From that point, the exponent parameter underwent two iterations to reach a PE exponent estimate of 1.8681 (95% CI: 1.8447 to 1.8915) using the Gauss-Newton method. The two approaches produced results that were largely in agreement, evidenced by the degree of overlap in the 95% confidence intervals of each estimate. This comes as no surprise because both methods effectively use the minimized sum-of-squares as a criterion.

The exponent estimates reported above were determined in the context of all the data, which dates back to 1871. It goes without saying that the nature of the game has changed since MLB's inception. This raises the question: has the exponent changed significantly over time? The first approach to answering this was to independently determine the exponent for each decade (Figure 5a). In the plot, the bars represent the 95% confidence interval for the exponent estimate for that decade. The figure suggests there is no obvious pattern, but the exponents typically remain in the range of 1.75 to 2.0.

We applied the same technique to estimate the exponent for any given year and represented it with a time series plot in Figure 5b. The plot reveals changes to the exponent at a higher resolution, providing an understanding of how much fluctuation can occur from season to season. While the exponent may appear to be slightly downward-trending upon first glance, a linear regression reveals that the slope is non-significant (p=0.1024). This result required that the 1878 outlier (present in the time series as the left-most high spike) be removed, for it had a high amount of leverage and a Cook's D value well above the threshold. With this in mind, the data suggests the exponent varies but not predictably. Therefore, we are safe to assume that our original PE exponent estimate of about 1.87 is as relevant to predicting a modern-day winning percentage as it is a historical one.

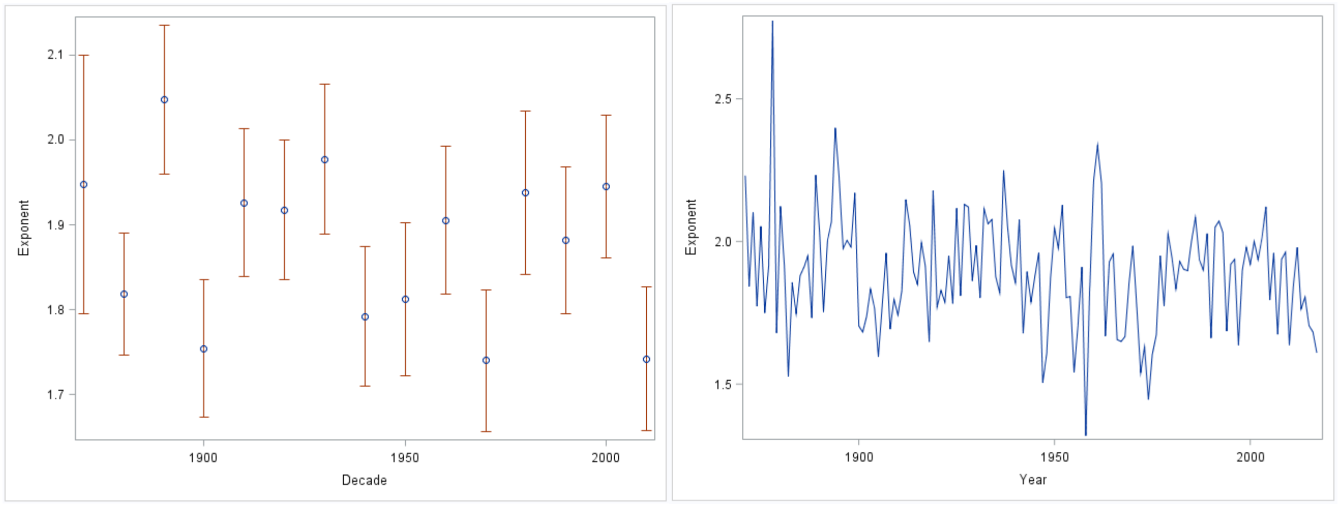


Figure 5. Exponent estimation by decade (a) and year (b).

Weibull Distribution Fits

In 2005, Steven Miller introduced a theoretical framework for estimating the PE exponent. His paper provided a justification for the PE formula and its associated exponent value by modeling a specific team's runs-scored and runs-allowed statistics for a season as independent random variables drawn from a Weibull distribution.

Examining the mathematics of the derivation in detail is beyond the scope of this project, but Miller did provide visualizations of the Weibull fits in his paper for all of the American League teams from the 2004 season. These were presented as histograms of team runs-scored and runs-allowed frequencies plotted on appropriately-shaped Weibull distributions, using the shape parameter γ estimated with ordinary least squares (OLS) and maximum likelihood estimation (MLE). Examples for two 2004 AL teams are shown in Figure 6.

Miller estimated the Weibull fits using runs-scored and runs-allowed data for each team structured as shown in Table 1. Using the 2004 Detroit Tigers as an example, each column in the table represents a discrete bin of runs scored or runs allowed, as well as the number of times (games) the team scored or allowed that number of runs during the season. For instance, as shown in the table, the 2004 Tigers scored eight runs in 14 games, while allowing twelve runs in two games. These data were obtained for each 2004 MLB team from *baseball-reference.com*, a baseball statistics website.

Miller used ordinary least squares (OLS) and maximum likelihood estimation (MLE) to converge on parameter estimates for γ, the Weibull distribution's shape parameter. Miller specified two equations in his paper, one for OLS, minimizing a sum-of-squares, and another for MLE, maximizing the likelihood of the Weibull shape parameter fitting the distribution for that team's season runs-scored and runs-allowed data. These equations are shown in Figures 7 and 8 respectively. Details can be found in Appendix B.

Detroit Tigers 2004	Games	5	16	16	22	22	20	13	10	14	7	5	6	2	1	1	2
	Runs Scored	0	1	2	3	4	5	6	7	8	9	10	11	12	13	15	17
	Games	9	11	15	25	19	17	15	15	8	8	8	7	2	1	1	1
	Runs Allowed	0	1	2	3	4	5	6	7	8	9	10	11	12	13	16	26

Table 1. Runs-scored and runs-allowed data for the 2004 Detroit Tigers

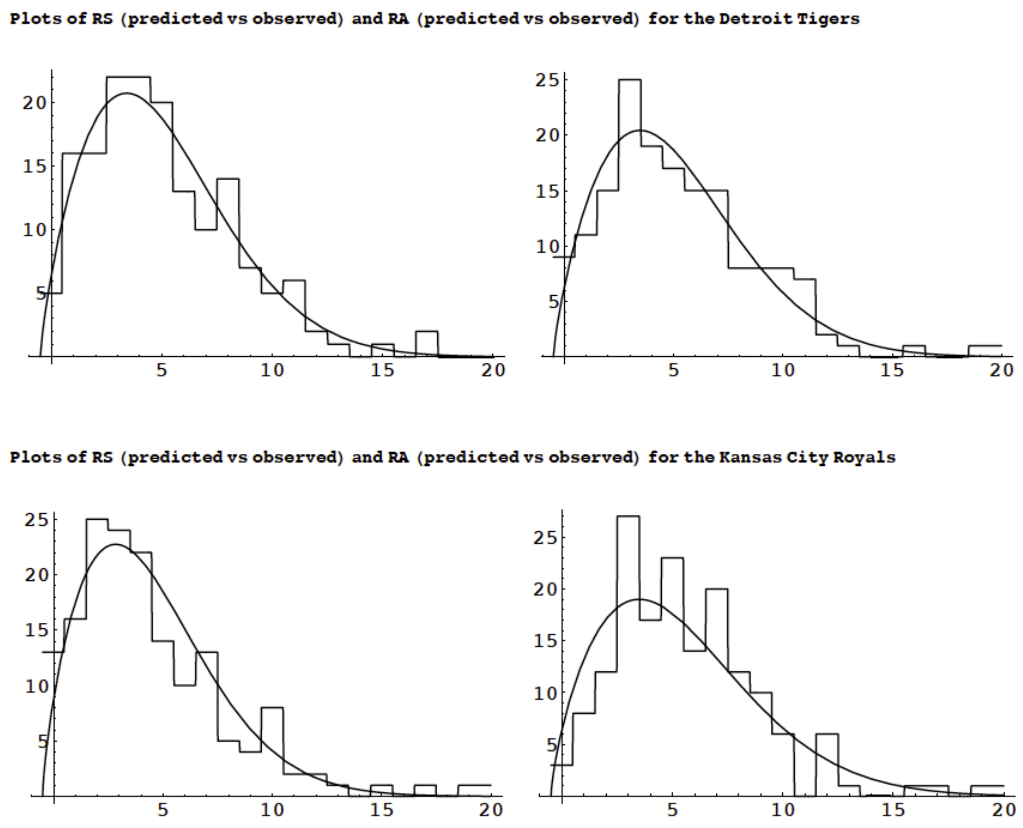


Figure 6. Weibull distribution fits from Miller (2005)

We took it as a challenge to finish what Miller started by producing similar Weibull fit visualizations for the 2004 National League teams. Armed with the equations above, we were able to implement a solution to estimate γ in both the R and Julia programming languages. After obtaining our estimates, we plugged them back into R plotting code to produce National League Weibull fit graphs such as those shown in Figure 9. It should be noted that our OLS estimates very closely match Miller's for the AL teams. Our MLE estimates however, agree closely for some teams and less so for others. The full list of PE exponent (γ) estimates can be found in Appendix C.

Ordinary Least Squares

$$\sum_{k=1}^{\#Bins}(RS_{obs}(k) - \#Games \cdot A(\alpha_{RS}, -0.5, \gamma, k))^2 + \sum_{k=1}^{\#Bins}(RA_{obs}(k) - \#Games \cdot A(\alpha_{RA}, -0.5, \gamma, k))^2$$

Figure 7. Ordinary least squares from Miller (2005). See Appendix B for details.

Maximum Likelihood Estimate

$$L(\alpha_{RS}, \alpha_{RA}, -0.5, \gamma) = \binom{\#Games}{RS_{obs}(1), ..., RS_{obs}(\#Bins)} \prod_{k=1}^{\#Bins} A(\alpha_{RS}, -0.5, \gamma, k)^{RS_{obs}(k)}$$
$$\cdot \binom{\#Games}{RA_{obs}(1), ..., RA_{obs}(\#Bins)} \prod_{k=1}^{\#Bins} A(\alpha_{RA}, -0.5, \gamma, k)^{RA_{obs}(k)}$$

Figure 8. Maximum likelihood estimation from Miller (2005). See Appendix B for details.

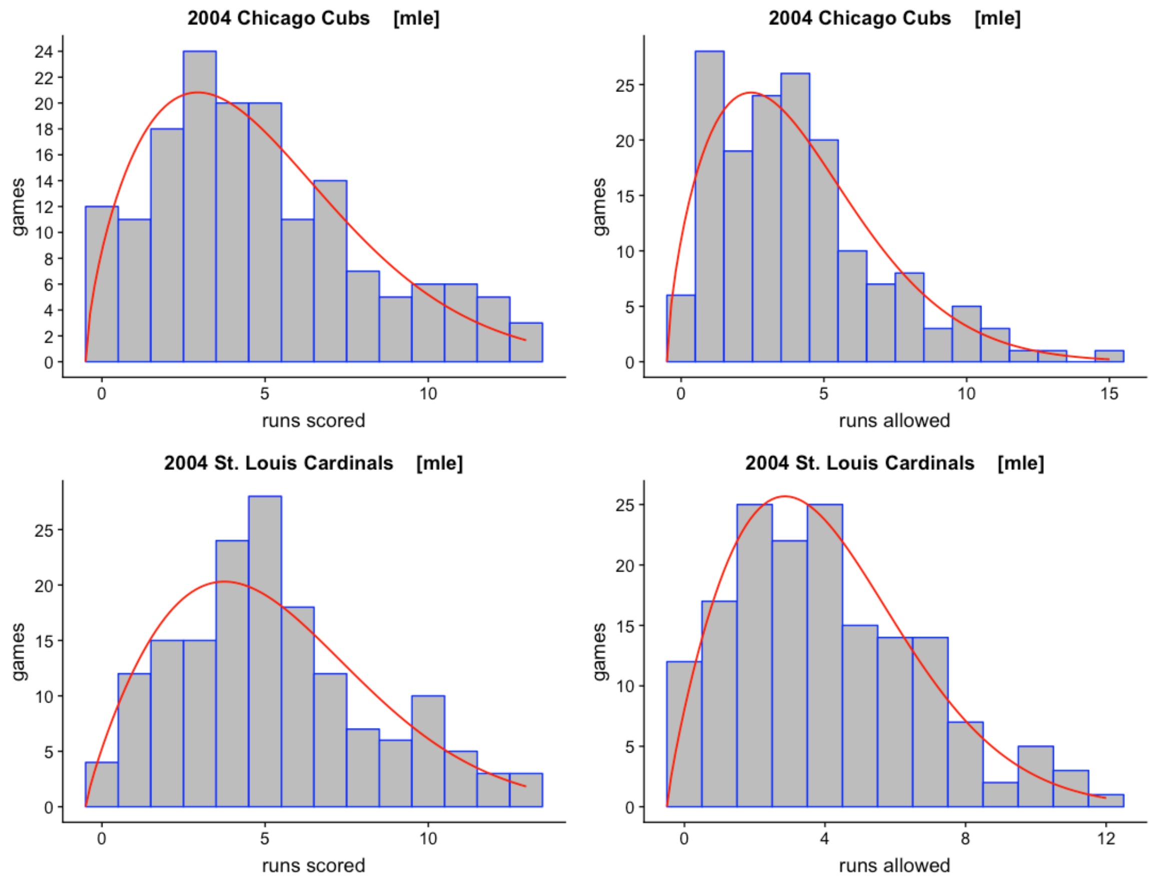


Figure 9. Weibull distribution fits for 2004, CHC and STL (NL teams)

This Weibull-fitting strategy provides the framework for a dynamic expectation model that could be updated daily. During the baseball season, updated runs data are available after the completion of each game. Professional analysts could calculate custom PE exponents for each team and use them for tailored PE-derived winning percentage estimates. In this way, each team could employ a custom strategy for predicting winning percentage and assess whether teams are performing above, at, or below expectation, in the context of competition from other teams, who each would have winning percentage expectations calculated with their own custom γ values.

Further Inquiries and Conclusion

Given more time, there are a few areas we would like to explore further in the context of this project. First, variable transformations, such as polynomial regression, might yield reductions in RMSE for the regression models. Even though from a purely baseball analytics point-of-view this exercise might be of dubious value (how would we interpret runs- or hits-squared, for example?), we'd like to satisfy our own curiosity and find out if the variable selection models might be improved further.

Second, we would like to explore interaction effects. In particular, the interplay between the components of hits (1B, 2B, 3B, and HR) and the other components that make up on-base percentage (BB, HBP).

Lastly, we would like to dig into the number of "stranded runners" that a team leaves on base but fail to score. We are currently unsure about the precise calculations required, but there must be a way to compute the number of runners a team leaves on base during an entire season. This might require some tedious sifting through daily game data, though it is possible the data could be obtained in a more straight-forward way. It would be interesting to quantify how success throughout the season is affected for teams that manage to minimize the number of stranded runners.

In the current era of advanced analytics, anything a professional baseball team can do to gain an analytical edge over the competition could translate into success on the field as well as at the bank. Modeling and predicting a team's winning percentage is a quick, efficient, accessible, and indeed quite accurate way to get a handle on a team's aggregate performance throughout the course of a season.

Every team's analytics strategy is different, and there may in fact exist valid reasons to build models more complex than those described in this report. Ultimately however, a team's controllers are going to seek to answer the following fundamental question: how is our team doing? There are many ways to answer this, and many important decisions will depend on that answer. If a team is doing well, how much should they spend to push for the playoffs? If they are not doing so well, does it make sense to trade players and offload salary obligations for more economical options at the draft and in the minor leagues?

Without a baseline assessment of how well a team is doing on a day-to-day basis, team managers will have a difficult time making decisions to maximize benefit for the team. Modeling winning percentage should be part of the backbone of any team's analytics strategy, if only just for the ability to quickly assess, on a daily basis and with fresh data, how a team is performing relative to the competition and their own history. Run-differential is good enough to get a handle on how often a team might be expected to win, and the Weibull fit approach detailed here provides a customizable approach that can be tailored to the priorities of the organization.

References

2004 MLB Scoring and leads summaries. (n.d). Retrieved Dec 12, 2018, from *baseball-reference.com*

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). *Julia: A fresh approach to numerical computing.* SIAM Review, 59(1), 65–98.

Marchi, M., & Albert, J. (2014). Analyzing Baseball Data with R. Boca Raton, FL: CRC Press, Taylor & Francis Group.

Miller, S. J. (2005, September 29). *A Derivation of the Pythagorean Won-Loss Formula in Baseball.* Chance Magazine 20, 40-48.

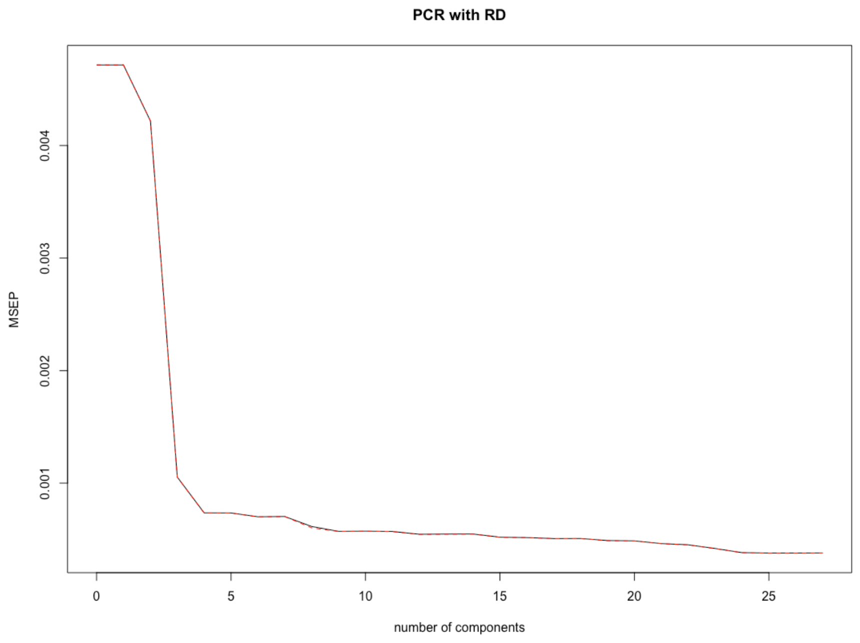
Pythagorean expectation. (n.d). In Wikipedia. Retrieved Dec 12, 2018, from https://en.wikipedia.org/wiki/Pythagorean_expectation

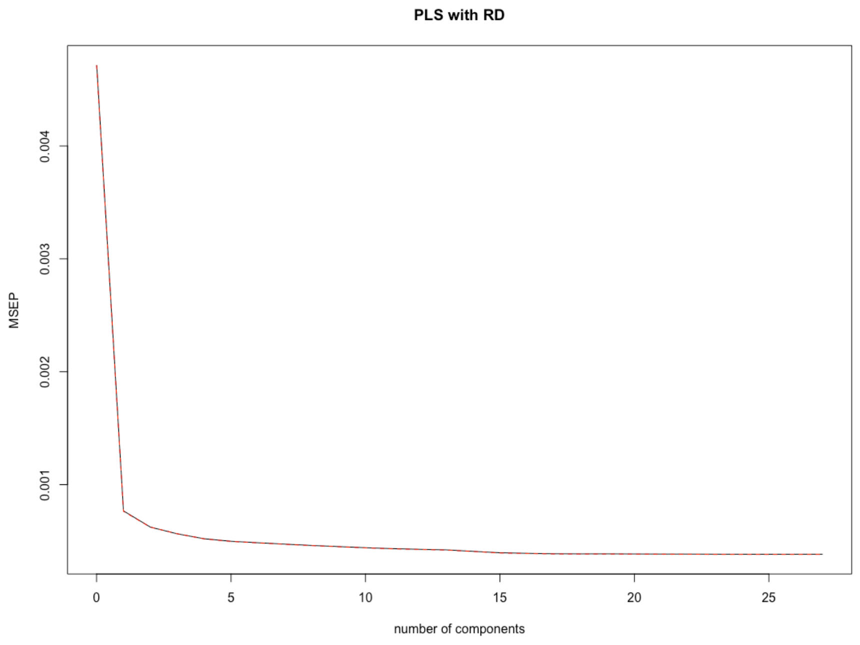
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Some of the analysis for this project was generated using SAS software. Copyright © 2018 SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA.

Appendix A

The validation plots for PCR and PLSR below show the number of components necessary to reduce and/or minimize MSEP, the mean squared error of prediction, with the R package *pls()*. At least 85% of the variance in the response is accounted for by just 7 components in the PCR case, and 2 components in the PLSR case. The corresponding points on the x-axes below (7 and 2) suggest a "first local minimum" for MSEP, while at the same time significantly reducing the dataset dimensionality.





Appendix B

For both equations listed in Figures 7 and 8, the function *A()* is the area under the Weibull distribution for bin *k*, in terms of the runs-scored and runs-allowed values from each bin in Table 1. Area *A()* was computed using an appropriate Weibull cumulative distribution function (cdf) provided by package *dweibull3()* for R or package *Distributions.jl* for Julia, calculating the integral value *F(b) – F(a)* for each bin. In this way, the continuous Weibull distribution was used to approximate or "smooth-out" the discrete runs-scored and runs-allowed histograms. Miller chose this strategy to simplify the computations and obtain closed-form solutions.

It should be noted that for MLE it is equivalent to maximize the logarithm of the likelihood, and our own code implementations in R and Julia reflect this adjustment. We also ignore the multinomial coefficients since these are independent of the distribution parameters. The theoretical background for this approach can be found in Miller's original paper.

The alpha (α_{RS} or α_{RA}) values are unique for runs-scored and runs-allowed for each team, and are calculated using the gamma function (Γ) available in R and Julia:

$$\alpha_{RS} = \frac{RS - \beta}{\Gamma(1+\gamma^{-1})}$$
$$\alpha_{RA} = \frac{RA - \beta}{\Gamma(1+\gamma^{-1})}$$

Where *RS* = the mean of runs-scored, *RA* = the mean of runs-allowed, and β = the Weibull shift parameter (-0.5 in the equations), which shifts the distribution plot to the left so that the number of runs scored or runs allowed are found in the *center* of each bin, rather than at the edges.

Our coding approach was straight-forward: our OLS and MLE functions looped through a reasonable range of minimum and maximum values (see Miller's estimates in Appendix C) until convergence was achieved on the minimized sum-of-squares or the maximized log-likelihood. We should point out that our first code implementation in R was quite slow, running in just under four minutes to estimate PE exponents (γ) via OLS and MLE for thirty 2004 MLB teams. However, we were able to speed this up significantly: our equivalent Julia implementation produces the very same results in less than one second of execution time.

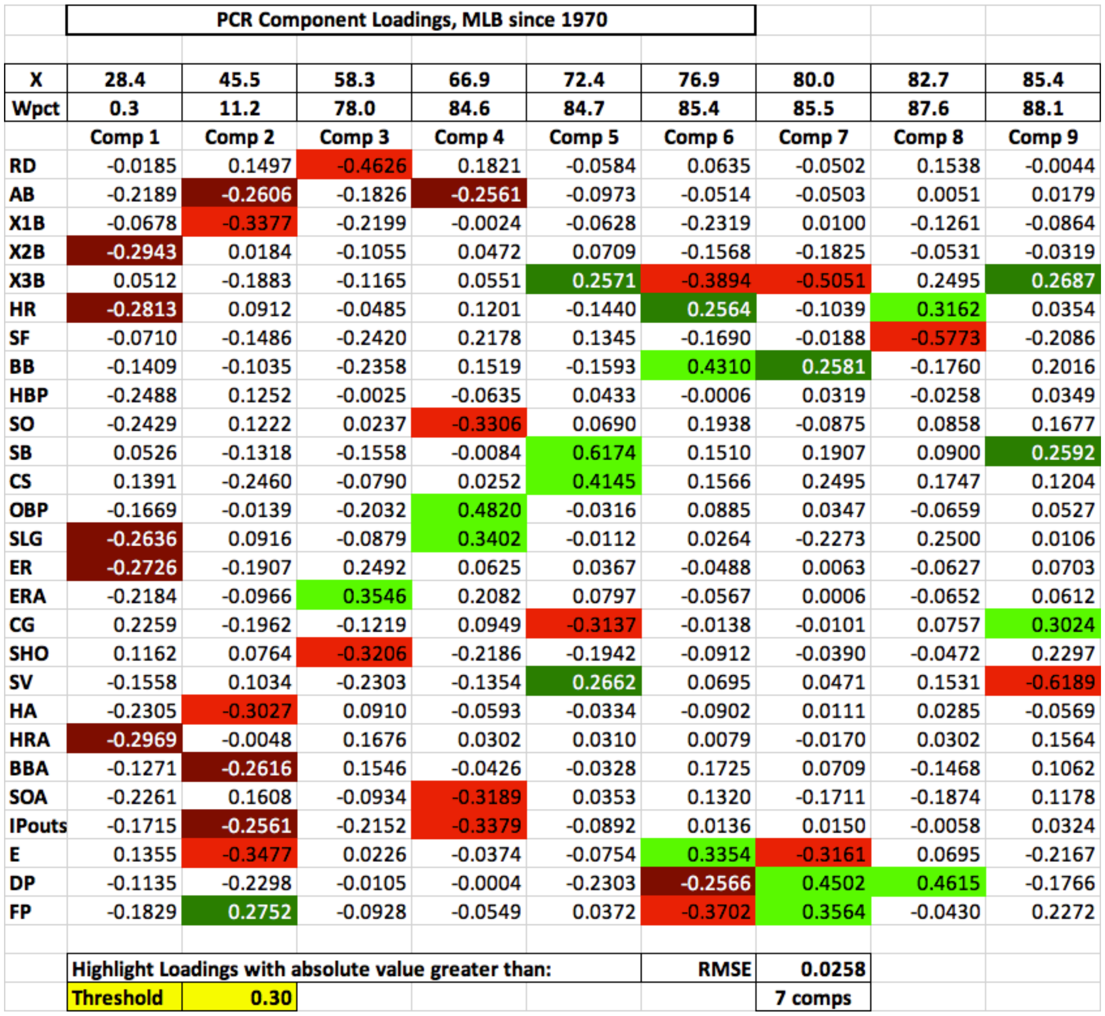
Appendix C

2004 MLB Pythagorean Exponent (γ) Estimates (Weibull fits)

Team	League	Miller OLS	Project OLS	Miller MLE	Project MLE
Boston Red Sox	AL	1.80	1.795	1.82	1.818
New York Yankees	AL	1.77	1.753	1.78	1.713
Baltimore Orioles	AL	1.63	1.635	1.66	1.641
Tampa Bay Devil Rays	AL	1.82	1.813	1.83	1.763
Toronto Blue Jays	AL	2.01	1.953	1.97	1.671
Minnesota Twins	AL	1.80	1.776	1.79	1.773
Chicago White Sox	AL	1.71	1.703	1.73	1.712
Cleveland Indians	AL	1.81	1.791	1.79	1.681
Detroit Tigers	AL	1.76	1.747	1.78	1.691
Kansas City Royals	AL	1.80	1.763	1.76	1.625
Los Angeles Angels	AL	1.68	1.682	1.71	1.660
Oakland Athletics	AL	1.79	1.759	1.76	1.747
Texas Rangers	AL	1.88	1.866	1.90	1.795
Seattle Mariners	AL	1.76	1.762	1.78	1.730
Atlanta Braves	NL		1.700		1.694
Philadelphia Phillies	NL		1.916		1.844
Florida Marlins	NL		1.648		1.705
New York Mets	NL		1.585		1.639
Montreal Expos	NL		1.537		1.509
St. Louis Cardinals	NL		1.825		1.841
Houston Astros	NL		1.745		1.761
Chicago Cubs	NL		1.666		1.656
Cincinnati Reds	NL		1.837		1.788
Pittsburgh Pirates	NL		1.610		1.622
Milwaukee Brewers	NL		1.578		1.629
Los Angeles Dodgers	NL		1.798		1.815
San Francisco Giants	NL		1.925		1.873
San Diego Padres	NL		1.796		1.800
Colorado Rockies	NL		1.842		1.876
Arizona Diamondbacks	NL		1.754		1.768

Appendix D

The PCR loading plot below shows a similar phenomenon to that of the PLSR plot in the main text, the significant contribution of feature RD. A significant jump in variance explained for the response, Wpct, can be observed at the top of the chart, quickly increasing from 11.2% to 78.0% between components two and three. The largest contributor to this jump is the RD variable, run-differential. Significant contributions from ERA (earned-run average) and SHO (shutouts) are also shown.



PCR Component Loadings, MLB since 1970

X	**28.4**	**45.5**	**58.3**	**66.9**	**72.4**	**76.9**	**80.0**	**82.7**	**85.4**
Wpct	**0.3**	**11.2**	**78.0**	**84.6**	**84.7**	**85.4**	**85.5**	**87.6**	**88.1**
	Comp 1	**Comp 2**	**Comp 3**	**Comp 4**	**Comp 5**	**Comp 6**	**Comp 7**	**Comp 8**	**Comp 9**
RD	-0.0185	0.1497	-0.4626	0.1821	-0.0584	0.0635	-0.0502	0.1538	-0.0044
AB	-0.2189	-0.2606	-0.1826	-0.2561	-0.0973	-0.0514	-0.0503	0.0051	0.0179
X1B	-0.0678	-0.3377	-0.2199	-0.0024	-0.0628	-0.2319	0.0100	-0.1261	-0.0864
X2B	-0.2943	0.0184	-0.1055	0.0472	0.0709	-0.1568	-0.1825	-0.0531	-0.0319
X3B	0.0512	-0.1883	-0.1165	0.0551	0.2571	-0.3894	-0.5051	0.2495	0.2687
HR	-0.2813	0.0912	-0.0485	0.1201	-0.1440	0.2564	-0.1039	0.3162	0.0354
SF	-0.0710	-0.1486	-0.2420	0.2178	0.1345	-0.1690	-0.0188	-0.5773	-0.2086
BB	-0.1409	-0.1035	-0.2358	0.1519	-0.1593	0.4310	0.2581	-0.1760	0.2016
HBP	-0.2488	0.1252	-0.0025	-0.0635	0.0433	-0.0006	0.0319	-0.0258	0.0349
SO	-0.2429	0.1222	0.0237	-0.3306	0.0690	0.1938	-0.0875	0.0858	0.1677
SB	0.0526	-0.1318	-0.1558	-0.0084	0.6174	0.1510	0.1907	0.0900	0.2592
CS	0.1391	-0.2460	-0.0790	0.0252	0.4145	0.1566	0.2495	0.1747	0.1204
OBP	-0.1669	-0.0139	-0.2032	0.4820	-0.0316	0.0885	0.0347	-0.0659	0.0527
SLG	-0.2636	0.0916	-0.0879	0.3402	-0.0112	0.0264	-0.2273	0.2500	0.0106
ER	-0.2726	-0.1907	0.2492	0.0625	0.0367	-0.0488	0.0063	-0.0627	0.0703
ERA	-0.2184	-0.0966	0.3546	0.2082	0.0797	-0.0567	0.0006	-0.0652	0.0612
CG	0.2259	-0.1962	-0.1219	0.0949	-0.3137	-0.0138	-0.0101	0.0757	0.3024
SHO	0.1162	0.0764	-0.3206	-0.2186	-0.1942	-0.0912	-0.0390	-0.0472	0.2297
SV	-0.1558	0.1034	-0.2303	-0.1354	0.2662	0.0695	0.0471	0.1531	-0.6189
HA	-0.2305	-0.3027	0.0910	-0.0593	-0.0334	-0.0902	0.0111	0.0285	-0.0569
HRA	-0.2969	-0.0048	0.1676	0.0302	0.0310	0.0079	-0.0170	0.0302	0.1564
BBA	-0.1271	-0.2616	0.1546	-0.0426	-0.0328	0.1725	0.0709	-0.1468	0.1062
SOA	-0.2261	0.1608	-0.0934	-0.3189	0.0353	0.1320	-0.1711	-0.1874	0.1178
IPouts	-0.1715	-0.2561	-0.2152	-0.3379	-0.0892	0.0136	0.0150	-0.0058	0.0324
E	0.1355	-0.3477	0.0226	-0.0374	-0.0754	0.3354	-0.3161	0.0695	-0.2167
DP	-0.1135	-0.2298	-0.0105	-0.0004	-0.2303	-0.2566	0.4502	0.4615	-0.1766
FP	-0.1829	0.2752	-0.0928	-0.0549	0.0372	-0.3702	0.3564	-0.0430	0.2272
	Highlight Loadings with absolute value greater than:					**RMSE**	**0.0258**		
	Threshold	**0.30**					**7 comps**		