Fangraphs Top Prospect Analysis

By Andrew Gold

Introduction

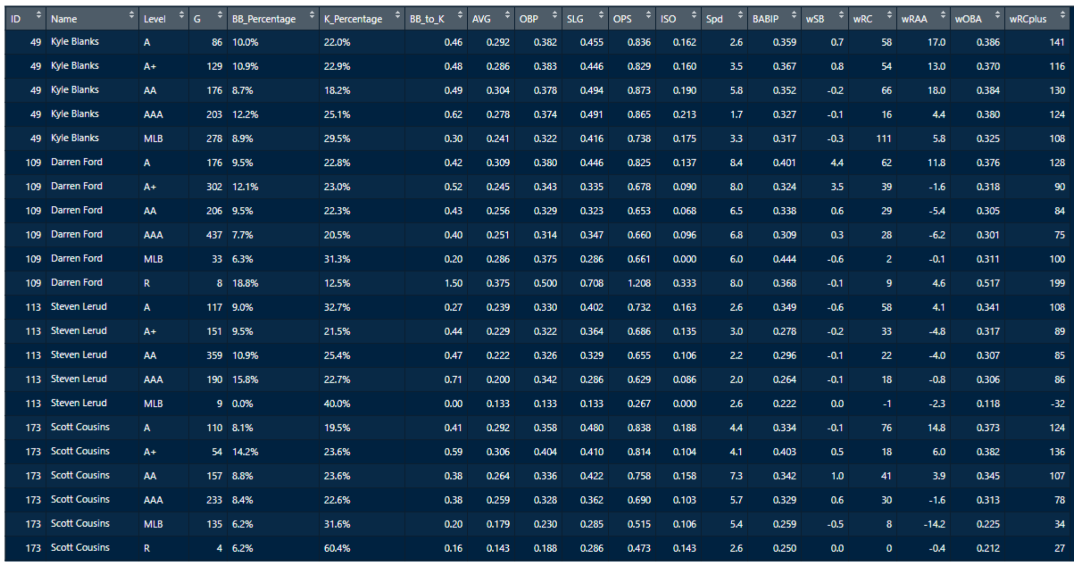
The goal of every Major League team is to constantly produce homegrown talent that will make an impact on the team's success at the big-league level. This is especially key for smaller market teams that cannot afford large free agent contracts, as these homegrown players are very affordable until their last couple of years of arbitration. The goal of this project was to gather historical data of current Major League players, along with their Minor League stats, and determine which metrics are best at predicting Major League success. By using R, we would then create a model and apply it to current MLB top prospects to predict their future production once they reach the Majors. Below, I detail the process of gathering and prepping the data, creation of the models, the interpretation of results, and how the model could be improved in the future.

Data Engineering

The first step in gathering the data was to plan out the project to determine the data source. I decided to gather the rosters for every MLB team dating back to 2010. I would then need to obtain both the MLB and MiLB stats for these players, I order to run a proper analysis. Once this analysis was complete, I would then apply it to the current Top Prospects. To start, I web scraped Baseball Reference to get the rosters for all 30 MLB teams between 2010 and 2021. I then used the "People" dataset in the "Lahman" database to get BR PlayerID's and names. This data frame was then merged with the rosters to combine the IDs with the players that played during my time frame. One goal I had from the beginning of the project was to use advanced metrics instead of the standard metrics. These advanced metrics are more accurate and adjust based on the run-scoring environment of each year, so you can accurately compare a player who is playing today to one who played years ago. The best place to get these metrics in my opinion is FanGraphs. Conveniently, FanGraphs also has the MiLB data, so I could web scrape all the data from one source. The key would be to get the FanGraphs player IDs, which I could then use to loop through each player link to acquire the data. By using the databank provided by the Chadwick Baseball Bureau, I was able to match the Baseball Reference IDs I had already scraped by the same IDs in the Chadwick database to get the Chadwick FanGraphs IDs. Now I had all the player names and FanGraphs IDs to scrape FanGraphs for the data. After exploring FanGraphs, I realized that they only started tracking MiLB stats dating back to 2006, so I had to cut my dataset slightly to only include players that debuted after 2006. This means that everyone in my dataset would have both Major and Minor League data.

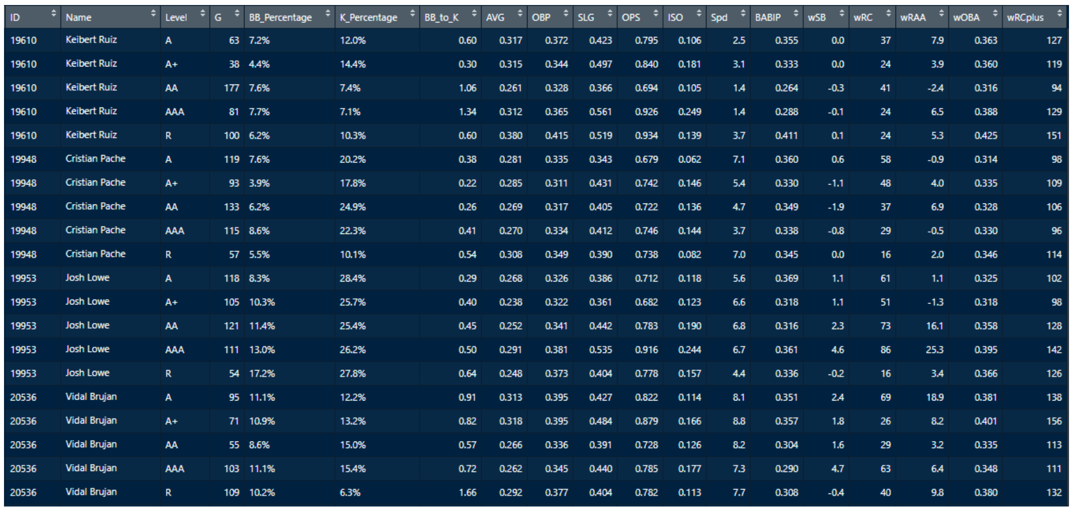
Now that I had all the IDs, I could scrape FanGraphs for their advanced metrics. The key to this was to find the correct "xpath" using Google Chrome developer tools. This would locate the “Advanced Metrics” table on FanGraphs and load it into a data frame in R. Throughout this process, there was a lot of data cleaning and manipulation to get the code into the format needed to run analysis, which you can find on my GitHub page. After scraping all the current Major Leaguers data, I did a similar process for the current Top 100 prospects. By using FanGraphs’ prospect rankings, I was able to download their Top 100 position player prospects, along with their ID’s, and do similar scraping and cleaning as described above. By the end of the data engineering process, I was left with two data frames, one with current Major Leaguers and their MLB and MiLB metrics, and one for the top prospects, with their MiLB metrics. Examples of the two data frames are below:

Current MLB



ID	Name	Level	G	BB_Percentage	K_Percentage	BB_to_K	AVG	OBP	SLG	OPS	ISO	Spd	BABIP	wSB	wRC	wRAA	wOBA	wRCplus
49	Kyle Blanks	A	86	10.0%	22.0%	0.46	0.292	0.382	0.455	0.836	0.162	2.6	0.359	0.7	58	17.0	0.386	141
49	Kyle Blanks	A+	129	10.9%	22.9%	0.48	0.286	0.383	0.446	0.829	0.160	3.5	0.367	0.8	54	13.0	0.370	116
49	Kyle Blanks	AA	176	8.7%	18.2%	0.49	0.304	0.378	0.494	0.873	0.190	5.8	0.352	-0.2	66	18.0	0.384	130
49	Kyle Blanks	AAA	203	12.2%	25.1%	0.62	0.278	0.374	0.491	0.865	0.213	1.7	0.327	-0.1	16	4.4	0.380	124
49	Kyle Blanks	MLB	278	8.9%	29.5%	0.30	0.241	0.322	0.416	0.738	0.175	3.3	0.317	-0.3	111	5.8	0.325	108
109	Darren Ford	A	176	9.5%	22.8%	0.42	0.309	0.380	0.446	0.825	0.137	8.4	0.401	4.4	62	11.8	0.376	128
109	Darren Ford	A+	302	12.1%	23.0%	0.52	0.245	0.343	0.335	0.678	0.090	8.0	0.324	3.5	39	-1.6	0.318	90
109	Darren Ford	AA	206	9.5%	22.3%	0.43	0.256	0.329	0.323	0.653	0.068	6.5	0.338	0.6	29	-5.4	0.305	84
109	Darren Ford	AAA	437	7.7%	20.5%	0.40	0.251	0.314	0.347	0.660	0.096	6.8	0.309	0.3	28	-6.2	0.301	75
109	Darren Ford	MLB	33	6.3%	31.3%	0.20	0.286	0.375	0.286	0.661	0.000	6.0	0.444	-0.6	2	-0.1	0.311	100
109	Darren Ford	R	8	18.8%	12.5%	1.50	0.375	0.500	0.708	1.208	0.333	8.0	0.368	-0.1	9	4.6	0.517	199
113	Steven Lerud	A	117	9.0%	32.7%	0.27	0.239	0.330	0.402	0.732	0.163	2.6	0.349	-0.6	58	4.1	0.341	108
113	Steven Lerud	A+	151	9.5%	21.5%	0.44	0.229	0.322	0.364	0.686	0.135	3.0	0.278	-0.2	33	-4.8	0.317	89
113	Steven Lerud	AA	359	10.9%	25.4%	0.47	0.222	0.326	0.329	0.655	0.106	2.2	0.296	-0.1	22	-4.0	0.307	85
113	Steven Lerud	AAA	190	15.8%	22.7%	0.71	0.200	0.342	0.286	0.629	0.086	2.0	0.264	-0.1	18	-0.8	0.306	86
113	Steven Lerud	MLB	9	0.0%	40.0%	0.00	0.133	0.133	0.133	0.267	0.000	2.6	0.222	0.0	-1	-2.3	0.118	-32
173	Scott Cousins	A	110	8.1%	19.5%	0.41	0.292	0.358	0.480	0.838	0.188	4.4	0.334	-0.1	76	14.8	0.373	124
173	Scott Cousins	A+	54	14.2%	23.6%	0.59	0.306	0.404	0.410	0.814	0.104	4.1	0.403	0.5	18	6.0	0.382	136
173	Scott Cousins	AA	157	8.8%	23.6%	0.38	0.264	0.336	0.422	0.758	0.158	7.3	0.342	1.0	41	3.9	0.345	107
173	Scott Cousins	AAA	233	8.4%	22.6%	0.38	0.259	0.328	0.362	0.690	0.103	5.7	0.329	0.6	30	-1.6	0.313	78
173	Scott Cousins	MLB	135	6.2%	31.6%	0.20	0.179	0.230	0.285	0.515	0.106	5.4	0.259	-0.5	8	-14.2	0.225	34
173	Scott Cousins	R	4	6.2%	60.4%	0.16	0.143	0.188	0.286	0.473	0.143	2.6	0.250	0.0	0	-0.4	0.212	27

Fangraphs Top Prospects



ID	Name	Level	G	BB_Percentage	K_Percentage	BB_to_K	AVG	OBP	SLG	OPS	ISO	Spd	BABIP	wSB	wRC	wRAA	wOBA	wRCplus
19610	Keibert Ruiz	A	63	7.2%	12.0%	0.60	0.317	0.372	0.423	0.795	0.106	2.5	0.355	0.0	37	7.9	0.363	127
19610	Keibert Ruiz	A+	38	4.4%	14.4%	0.30	0.315	0.344	0.497	0.840	0.181	3.1	0.333	0.0	24	3.9	0.360	119
19610	Keibert Ruiz	AA	177	7.6%	7.4%	1.06	0.261	0.328	0.366	0.694	0.105	1.4	0.264	-0.3	41	-2.4	0.316	94
19610	Keibert Ruiz	AAA	81	7.7%	7.1%	1.34	0.312	0.365	0.561	0.926	0.249	1.4	0.288	-0.1	24	6.5	0.388	129
19610	Keibert Ruiz	R	100	6.2%	10.3%	0.60	0.380	0.415	0.519	0.934	0.139	3.7	0.411	0.1	24	5.3	0.425	151
19948	Cristian Pache	A	119	7.6%	20.2%	0.38	0.281	0.335	0.343	0.679	0.062	7.1	0.360	0.6	58	-0.9	0.314	98
19948	Cristian Pache	A+	93	3.9%	17.8%	0.22	0.285	0.311	0.431	0.742	0.146	5.4	0.330	-1.1	48	4.0	0.335	109
19948	Cristian Pache	AA	133	6.2%	24.9%	0.26	0.269	0.317	0.405	0.722	0.136	4.7	0.349	-1.9	37	6.9	0.328	106
19948	Cristian Pache	AAA	115	8.6%	22.3%	0.41	0.270	0.334	0.412	0.746	0.144	3.7	0.338	-0.8	29	-0.5	0.330	96
19948	Cristian Pache	R	57	5.5%	10.1%	0.54	0.308	0.349	0.390	0.738	0.082	7.0	0.345	0.0	16	2.0	0.346	114
19953	Josh Lowe	A	118	8.3%	28.4%	0.29	0.268	0.326	0.386	0.712	0.118	5.6	0.369	1.1	61	1.1	0.325	102
19953	Josh Lowe	A+	105	10.3%	25.7%	0.40	0.238	0.322	0.361	0.682	0.123	6.6	0.318	1.1	51	-1.3	0.318	98
19953	Josh Lowe	AA	121	11.4%	25.4%	0.45	0.252	0.341	0.442	0.783	0.190	6.8	0.316	2.3	73	16.1	0.358	128
19953	Josh Lowe	AAA	111	13.0%	26.2%	0.50	0.291	0.381	0.535	0.916	0.244	6.7	0.361	4.6	86	25.3	0.395	142
19953	Josh Lowe	R	54	17.2%	27.8%	0.64	0.248	0.373	0.404	0.778	0.157	4.4	0.336	-0.2	16	3.4	0.366	126
20536	Vidal Brujan	A	95	11.1%	12.2%	0.91	0.313	0.395	0.427	0.822	0.114	8.1	0.351	2.4	69	18.9	0.381	138
20536	Vidal Brujan	A+	71	10.9%	13.2%	0.82	0.318	0.395	0.484	0.879	0.166	8.8	0.357	1.8	26	8.2	0.401	156
20536	Vidal Brujan	AA	55	8.6%	15.0%	0.57	0.266	0.336	0.391	0.728	0.126	8.2	0.304	1.6	29	3.2	0.335	113
20536	Vidal Brujan	AAA	103	11.1%	15.4%	0.72	0.262	0.345	0.440	0.785	0.177	7.3	0.290	4.7	63	6.4	0.348	111
20536	Vidal Brujan	R	109	10.2%	6.3%	1.66	0.292	0.377	0.404	0.782	0.113	7.7	0.308	-0.4	40	9.8	0.380	132

Data Preparation

After scraping the data and getting it into a format similar to how you would view it on FanGraphs, I needed to arrange the data in the Current MLB player dataset so that their MLB and MiLB data were side by side in columns, with one row per player, instead of multiple rows per player, by level, like you would see on the back of a baseball card. By having the data set up this way, I could run a proper regression analysis. The issue, though, was that I did not want to neglect the value that the MiLB level had on player production. The higher the level, the more difficult, and I did not want to lose that when combining all of a player's MiLB data into one metric. The reason for this is because a player who performs at Triple-A is much more likely to have an impact at the MLB level than one currently Single-A. To get around this, I applied weights based on the level of play in which the numbers were produced. The weights are as follows:

R: 1

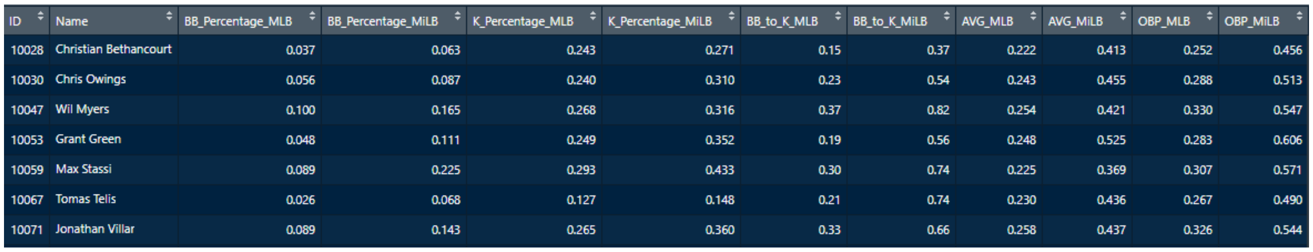
A = 1.25

A+ = 1.5

AA = 1.75

AAA = 2

By using these weights, the numbers a player produces at Triple-A are twice as important as in Rookie ball, since the competition is much better. Now that each metric has been weighed based on level, I could group by the player's ID and name and get an average for each statistic, weighed by the level, while in the minors. Now, the columns in the dataset are setup up so that their MLB and MiLB metrics are side by side. Below is an example for a few players:



ID	Name	BB_Percentage_MLB	BB_Percentage_MiLB	K_Percentage_MLB	K_Percentage_MiLB	BB_to_K_MLB	BB_to_K_MiLB	AVG_MLB	AVG_MiLB	OBP_MLB	OBP_MiLB
10028	Christian Bethancourt	0.037	0.063	0.243	0.271	0.15	0.37	0.222	0.413	0.252	0.456
10030	Chris Owings	0.056	0.087	0.240	0.310	0.23	0.54	0.243	0.455	0.288	0.513
10047	Wil Myers	0.100	0.165	0.268	0.316	0.37	0.82	0.254	0.421	0.330	0.547
10053	Grant Green	0.048	0.111	0.249	0.352	0.19	0.56	0.248	0.525	0.283	0.606
10059	Max Stassi	0.089	0.225	0.293	0.433	0.30	0.74	0.225	0.369	0.307	0.571
10067	Tomas Telis	0.026	0.068	0.127	0.148	0.21	0.74	0.230	0.436	0.267	0.490
10071	Jonathan Villar	0.089	0.143	0.265	0.360	0.33	0.66	0.258	0.437	0.326	0.544

Model Creation and Analysis

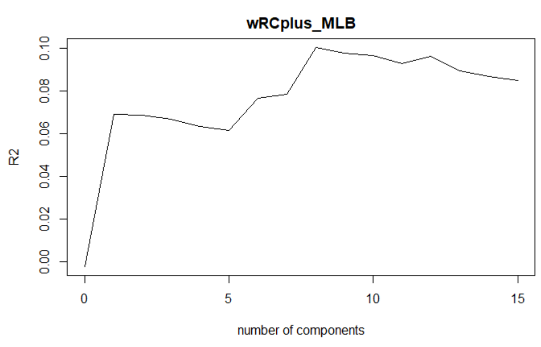
With a stat line full of valuable metrics, I decided to use wRC+ as my y-variable. This is because wRC+ is generally viewed as the most encompassing advanced metric that takes almost everything a batter does into account. With it being adjusted to a league average of 100, it is also easy to understand and compare with other players. The model I created would be taking wRC+ at the MLB level, or wRCplus_MLB in my code, and using all the MiLB metrics to predict the wRC+ at the Major League level. To start, I created scatterplots with regression lines using wRC+ and each MiLB metric. Immediately there were some metrics, like wSB and wRC, that would not have a profound impact on the model due to the regression line being relatively flat.

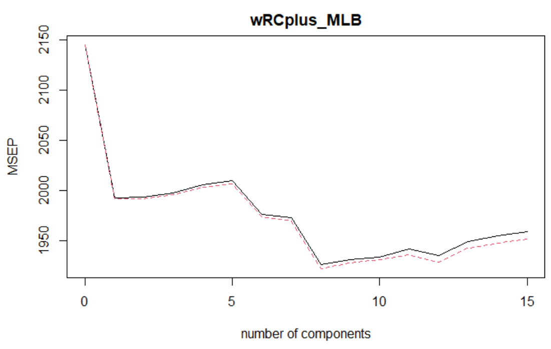
To being my model creation, I split the Current MLB player data into train and test subsets using an 80/20 split. This would allow me to use Machine Learning techniques and compare RMSE scores of all models at the end. After this split, I created a model using all the MiLB metrics to predict wRC+, but my model showed coefficient estimates as negative when they should be positive. For example, the initial regression equation showed a negative correlation with a metric like OBP, meaning that for every increase in OBP, wRC+ would decrease. In reality, this is not true. A player that increases their OBP will see an increase in wRC+, as getting on base is a positive outcome for an at-bat. This is when I realized that the correlation between all the metrics were having a heavy impact on the results. When there is so much correlation between variables, the coefficient estimates get thrown off. This is not surprising because all advanced metrics are based off of or build off of the others. To limit this, I created a correlation plot to select variables that were less correlated with each other to put in a model.

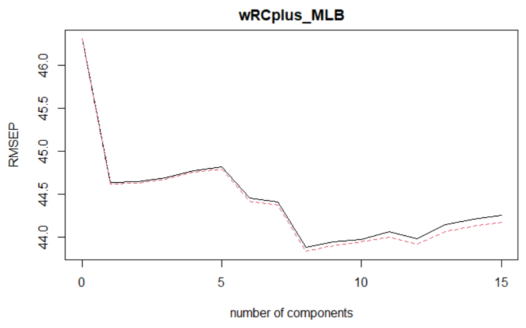
For the first model displayed in my code, “m1”, I used the correlation plot to select which metrics to input. To predict wRC+ at the big-league level, I used MiLB wOBA, wRAA, and wRC. When looking at the output for this linear regression model, we see that the wRC_MiLB variable has a negative coefficient. The second model I use, “m2”, removes wRC_MiLB due to the negative coefficient, since an increase in wRC would not result in a decrease in wRC+. This negative coefficient is due to the correlation between variables, as explained above. In model 2, I use the same process as model 1, but with just MiLB wOBA and wRAA. While the Adjusted R^2 is slightly lower with model 2 (.0989 vs. .1188) we no longer have negative coefficients.

To try other machine learning techniques, I applied a ridge regression model to the dataset. The correlation amongst the variables, though, makes many of the coefficient estimates negative.

Finally, I applied a principal components model to the dataset. Principal components regression is meant to be used when the variables within a dataset are highly correlated. The key to this type of analysis is to determine the number of components to use when predicting.







As we can tell by the plots above, 8 components are ideal for this model as it attains the highest R^2, and lowest MSE and RMSE.

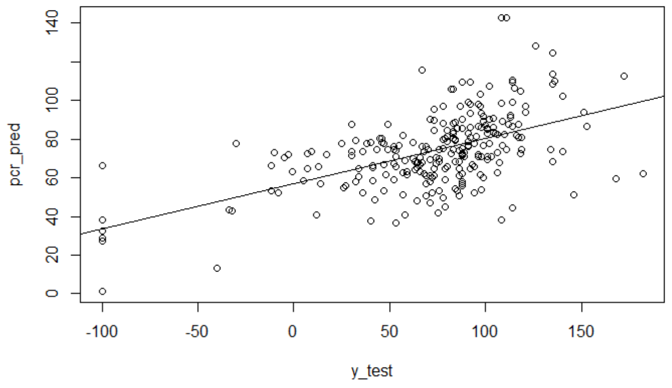
To compare the models, I calculated the RMSE for m1, m2, and the principal components model. The RMSE results were:

m1 = 39.13

m2 = 39.76

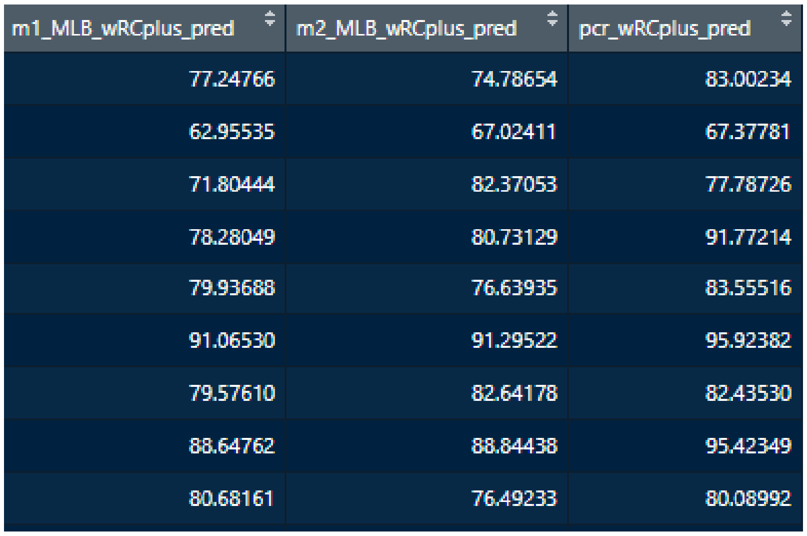
pcr = 38.34

As we can see, the principal components model has the lowest RMSE which means it is our preferred model. Also, we know principal components analysis accounts for the correlation between variables which is an added benefit. Below, we can see the plot comparing the actual wRC+ (y_test) and what the principal components model predicted (pcr_pred).



The principal component model accurately predicts that as the actual wRC+ scores increase, the principal components predictions also increase. While there is a lot of variance, and some major outliers, this is a great start in building a functional model to predict which metrics are best at projecting current minor leaguers' future production at the major league level.

Finally, we apply these models to the current minor leaguers' dataset. We can apply these models to the current minor leaguers to predict their MLB wRC+.



m1_MLB_wRCplus_pred	m2_MLB_wRCplus_pred	pcr_wRCplus_pred
77.24766	74.78654	83.00234
62.95535	67.02411	67.37781
71.80444	82.37053	77.78726
78.28049	80.73129	91.77214
79.93688	76.63935	83.55516
91.06530	91.29522	95.92382
79.57610	82.64178	82.43530
88.64762	88.84438	95.42349
80.68161	76.49233	80.08992

Our final MiLB dataset now has all three model predictions as pictured above. We see that each model is slightly different, but usually the results are close.

While the first assumption would be to see this data and think that most of these players will be below average Major Leaguers, I believe it would be best to use this as a comparative tool amongst the group, not necessarily as a projection of their actual major league wRC+. The mean of the principal components' prediction is 82.86, so it would be better to view these predictions as players with above an 82.6 pcr wRC+ are projected to have above average production compared to the rest of the players in this group. Also, we should note that the adjusted R^2 for the first two models are very low. In reality, predicting how current Minor Leaguers will produce in the future at the Major League is nearly impossible. We constantly see that the teams that can produce big leaguers through their farm system have sustained success, and this is one of, if not the greatest competitive advantages organizations have over each other. While a sufficient model could be produced when a lot of resources are available, it still comes down to the actual development of each player and how well the player development team can push a player through the Minor Leagues and prepare them for the Majors.

Future Improvements

With more time and resources, this model and dataset could be improved tremendously. I believe the best way to improve this model is to include batted ball data from sources like TrackMan, Rapsodo, and Blast Motion. We know that a ball hit harder has a higher likelihood of being a hit, thus leading to more production, so if we can find which batted ball metrics have the most impact, we can compare that of MiLB players to those of MLB players to get better predictions. These metrics could include Barrel%, max exit velocity, average exit velocity, bat speed, swing decisions etc. I am sure there is a correlation between bat speed and Barrel% that leads to a higher likelihood of success at the Major League level. While teams are already targeting players based on their batted ball metrics, it would be interesting to investigate which of these metrics matter most, and if they are better indicators than the advanced metrics. A team with a lot of resources could also consider the grades scouts give on players using the 20/80 scouting scale. This would blend the “old school” and “new school” ways of thinking and also incorporate everyone within Baseball Ops opinions. Finally, when it comes to the models I have created with the data I scraped, it would be helpful to shrink the current Major Leaguers dataset based on games played. As we can see in the predictions vs. actual values plot above, there are some players with very low and negative wRC+ values. This is because of a very small sample size at the MLB level. A bad debut of 15 at-bats, or even a poor 10 games, could result in a very low wRC+ when those are the only chances you have at the MLB level. If we can find the correct number of games, or sample size, to suffice a normalized wRC+, the model results may be slightly improved, and this would eliminate the outliers displayed on the plot.