Understanding Baseball in an Analytical Fashion Using Similarity Scores to Compare Team Performances and Player Performance

Kai Chang

Ay119, SP 2016

June 2, 2016

Abstract

I've always been into baseball – more for the stats than for the game itself. Don't get me wrong, the game is great, but what I love the game is what's behind it. Each action made can be contributed deep down to numbers, the statistics. For this AY class, I decided to focus on a number of my favorite aspects of baseball and some of the tasks I've always wanted to do, and attempt to complete them to an acceptable degree. This is rather a starting point into full analysis, and this will act as finding some preliminary results, whether it be successful or not. We focus on determining similarity scores within baseball teams across many seasons, visualizing the aspects or characteristics that either successful teams or non-successful teams share. This is particularly the grunt work, as we cover SQL/CSV queries, to implementing our own sabremetrics through effective programming practices, to isolating resulting data and interpreting through visual models. From here, we can expand further to apply our learning to individual major league players, and by finding similar players throughout history, we can then attempt to project performance of baseball players ten years down the road. We will compare the results to the PECOTA results.

Contents

Source Code Snippets

1 Introduction

"Baseball is a sport dominated by statistics."[8] Every part of the game is statistic based, and most scenarios are binary based. Although players, like variables in nature, and bound by Gaussian tendencies, baseball seems unpredictable. However, many sabremetrics will argue that the reason can be explained behind the number, and there is a lot of it. Baseball stats have been recorded since the late 19th century, where the sport first began, and the current day database contains numbers describing the exact pitch in a game. This large database and statistical tendencies in the sport make baseball the perfect dataset.

Baseball is a data-driven sport, more so than others. "What were looking for in any stat is that it can help us make better predictions. When it comes down to it, thats all we need; if a stat cant lead to actionable insights, its really of no use to us."[8]

1.1 Objectives

Thus, the goal of this project is to understand the current state of baseball, and potentially project stats in the future. The most important aspect is focusing on similarities between entities, either players or teams. So, we look into analyzing and quantifying similar team performances across time, beyond wins and losses. From gaining this insight, we can then apply this to quantifying similarity scores in players, and by gathering similar players, we can use their later years to project the statistics of our desired player years (up to 10) down the road. This can be compared to the current PECOTA numbers, and down the road, when the events play out, we can determine which one is more accurate.

It's important to start at the team level, because of the sample size of the team data relative to the individual player data. Once the analysis is successful on the team data, only then can we apply it to a much larger mapping of player data (with season frames).

What is meant by season frames is this: a team season performance are individual and only respective to that season – one cannot combine multiple seasons of the same team due to the constant change in rosters, so queries of similar performances are limited to 1 season or 1 datapoint (vector). An individual player, however, can query multiple seasons to better gauge similarities, meaning the search is much more difficult.

2 Understanding Statistics

It's important to understand the statistics we use and develop to determine a similarity score between two players. These statistics fall into three categories: traditional statistics,

or statistics that have been used universally for some time now, professional sabremetrics, or statistics that have been developed to give a more weighted understanding of performance, and individualized sabremetrics, or statistics personally developed that I felt current sabremetrics lacked.

2.1 Traditional Statistics

2.1.1 OPB

OBP or **On Base Percentage** is a statistic used to measure how often a batter gets on base. The formula is given by

$$OBP = \frac{H + BB + HBP}{AB + BB + HBP + SF} \tag{2.1.1}$$

where H is hits by batter, BB is walks by batter, HBP is times batter is hit by pitch, AB is total at bats, and SF is sacrifice flies made by batter. This is an independent statistic from average because it takes into accounts sacrifices made by batters to improve chances of team success. Higher OPB suggests more successful hitters.

2.1.2 SLG

SLG or **Slugging Percentage** is a statistic used to measure how much power a batter has. The formula, given by

$$SLG = \frac{1B + 2 \times 2B + 3 \times 3B + 4 \times HR}{AB} \tag{2.1.2}$$

where $1B$ is singles, $2B$ is doubles, $3B$ is triples, and HR represents home runs made. SLG is an elementary way of differentiating production amongst hitters, giving weight to those hits that provide a higher probability of scoring. Higher SLG suggests more successful hitters.

2.1.3 ERA

ERA or **Earned Run Average** is a statistic used to measure the earned runs a player would give up in a traditional nine-inning game. The formula is given by

$$ERA = 9 \times \frac{ER}{IP} \tag{2.1.3}$$

where ER stands for the number of earned runs (runs minus errors made by fielding), and IP represents the innings pitched. Henry Chadwick first invented this statistic to measure the effectiveness of pitchers. Lower ERA suggests more successful pitchers.

2.1.4 WHIP

WHIP or **Walks Plus Hits per Inning Pitched** is a measurement of the number of base runners a pitcher allows in an inning. The formula, given by

$$WHIP = \frac{BB + H}{IP} \tag{2.1.4}$$

helps give an additional aspect to a player besides giving up runs. It provides an understanding of the tendency for a player to allow batters to reach base. Lower WHIP suggests more successful pitchers.

2.2 Professional Sabremetrics

2.2.1 BABIP

BABIP or **Batting Average on Balls in Play** measures the tendency for a ball in play play to be recorded as a hit.

$$BABIP = \frac{H - HR}{AB - SO - HR + SF} \tag{2.2.1}$$

where SO represents the strikeouts of the player. BABIP is a step above average because it has a tendency to account for luck. We can notice that a player whose BABIP is much higher or lower that his average BABIP can be accounted for by good or bad luck. "If a player has a BABIP of .450 over his last 10 games, its a sign that hes gotten really lucky; perhaps it was the location of his hits or poor defense, but no one will maintain a .450 BABIP over the long run."[8]. Note the average BABIP is around 0.300.

2.2.2 wOBA

wOBA or **Weighted On-Base Average** is a measurement that values certain hits more than others. wOBA values distributes the weights of hits according to their probabilities to score for a team. From fangraphs[6]:

> Batting average assumes that they are. On-base percentage does too, but does one better by including other ways of reaching base such as walking or

being hit by a pitch. Slugging percentage weights hits, but not accurately (Is a double worth twice as much as a single? In short, no) and again ignores other ways of reaching base. On-base plus slugging (OPS) does attempt to combine the different aspects of hitting into one metric, but it assumes that one percentage point of SLG is the same as that of OBP. In reality, a handy estimate is that OBP is around twice as valuable than SLG (the exact ratio is x1.8). In short, OPS is asking the right question, but we can arrive at a more accurate number quite easily.

Weighted On-Base Average combines all the different aspects of hitting into one metric, weighting each of them in proportion to their actual run value. While batting average, on-base percentage, and slugging percentage fall short in accuracy and scope, wOBA measures and captures offensive value more accurately and comprehensively.

$$wOBA = \frac{0.75 \times BB + 0.9 \times HBP + 0.89 \times 1B + 1.24 \times 2B + 1.56 \times 3B + 1.95 \times HR}{PA} \tag{2.2.2}$$

where PA represents a batter's total plate appearance (different from at bats).

Note that wOBA varies across all sites, but for this case study, we approximate wOBA as such, based on our truncated data.

2.2.3 FIP

FIP or **Field Independent Pitching** measures a pitchers performance without defense. None of the statistics that are defensive dependent are involved.

$$FIP = \frac{13 \times HR + 3 \times (BB + HBP) - 2 \times SO}{IP} + FIP\_Constant \tag{2.2.3}$$

where in our case the $FIP\_Constant$ is zero. This looks past the ERA and gives more weight to those pitchers that can strike out more, which is typically more dominant pitchers. However, this may overlook pitchers that rely on fielding as a groundball pitcher.

2.2.4 PBABIP

PBABIP is given in the same context as a batter's BABIP, but now for how lucky or how effective some pitcher are. When a pitcher has a high BABIP, it suggests that an improvement of performance is in the future. However, with a low BABIP, the opposite can be applied. BABIP is lower for those that rely on fielders or groundball pitchers.

$$PBABIP = \frac{HA - HRA}{(IPouts + HA + HRA + BBA + SOA) - BBA - HRA + DP/4.00 - E/8.00} \tag{2.2.4}$$

where HA is hits allowed, HRA is home runs allowed, $IPouts$ is total outs pitched, BBA is walks allowed, SOA is strike outs allowed, DP is double plays, and E is errors. Note this is slightly different from other PBABIP as this includes DP and E.

2.3 Individualized Sabremetrics

2.3.1 effERA

The effERA or **Effective ERA** is a self-developed statistic that determines an earned run average score by awarding the pitcher for complete games and shutouts, while penalizing the pitcher for having relief pitchers pitch, and homers allowed. The effective ERA is a take on the ERA of a pitcher from the pressures and environments around him or her.

$$effERA = ERA - 0.018 \times SHO - 0.006 \times CG + 0.002 \times HRA + 0.0015 \times SV + (1.00 - FP)/2 \tag{2.3.1}$$

where SHO is shutouts pitched, CG is complete games pitched, SV represents saves and FP is fielding percentage.

Note that this is for team statistics, because this effERA would not work for closers (with high SVs) or pitchers (who don't usually incur saves). To modify this effERA for individual players, SV coefficients would need to be adjusted. Fielding Percentage would also need to be reweighed to include accurate statistic.

2.3.2 Complete Game Rate

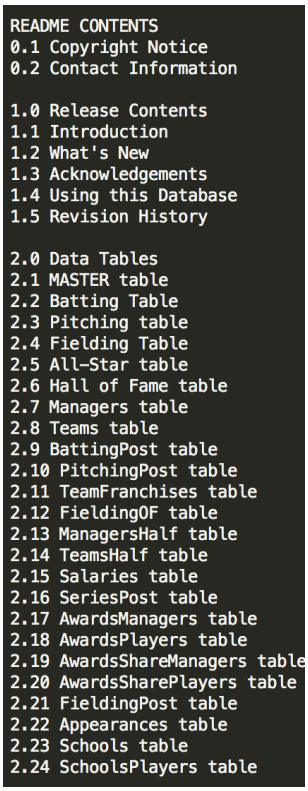
The complete game rate is a statistic looking into how dominant a team's starting rotation is. It is suggestive that better pitchers throw more complete games, either through their endurance, or through the box score. This can be both a pitcher and team statistic.

$$CGRate = \frac{CG}{IPouts/27} \tag{2.3.2}$$

3 Procedure

In order to tackle this project, we went through the following steps.

- First, we took the SQL and CVS files from the Lahman's Baseball Database, updated up to 2015. Within this database contains lots of tables for every possible datapoint.



- Familiarize self with R and SQL. In order to compare players at a fast rate, we will need to understand how queries run within R.

- Develop similarity algorithm or method for teams, taking in CVS datafiles.

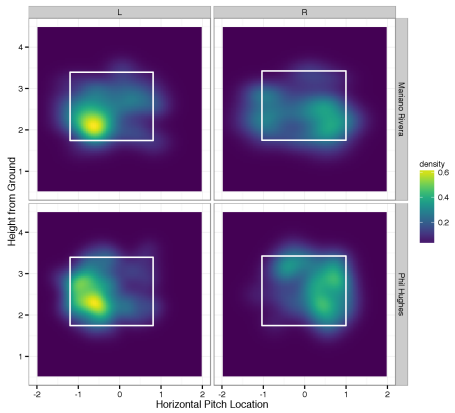
- Apply algorithm to a number of current teams, running the program to compare select team to historical teams. Then, we store the similarity scores for both batting and pitching.

- Visualize data with sabremetric statistics results from our refined query/computation CVS file.
- Interpreting any potential groupings or traits, apply PCA methods. We then see if the groupings developed share similar similarity scores as the ones computed. If so, then our algorithm and sabremetric tools were successful, and the same can be modified and applied to individual players.

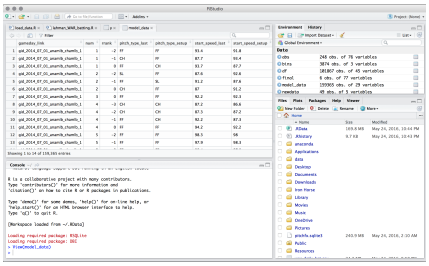
4 Database / Algorithm

4.1 R

We first needed to experiment with databases through R and SQL/CVS. This is reasonable as the player database has a lot more components, and will need a more powerful program. Thus, we followed and manipulated with the module *Interactive animations of PITCHf/x with animint and pitchRx* and *Introduction to pitchRx package*. From these modules, we learned how to pull the PITCHf/x data straight from MLB, which is a database of all pitches thrown up to the detail of spin rate, velocity, and movement. We also learned how to collect data from an online database, manage data in R, and visualize it in 2D and 3D. Although this did not directly apply to the analysis, it did give me more insight into my analysis.



This is very cool stuff, which I may consider later – heat maps, pitch frames, groupings. This can be used down the line to help with similarity calculations.



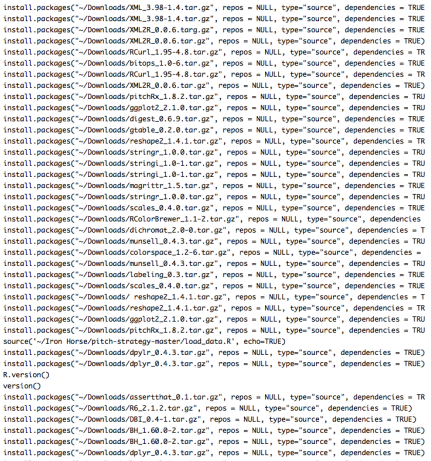


Figure 1: 2014 Results in various binnings

I also played around with RStudio a bit, getting help with a Github tutorial to extract data, learning R along the way. R is now set up for any baseball analytics task.

4.2 Python

The full grunt of the work was written in Python. We imported our CSV files, and the program normalized each team's sabremetric as vectors to that season's league average. Then it calculated the similarities between the two by deriving at a score using the sabremetric statistics. From there, we saved the resulting data to a CSV to be visualized and further analyzed down the road. We compared each individual team from seasons 2012-2015 to each individual team's performance from season 1960-2015.

yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	WSWin	R	AB	H	2B	3B	HR	BB	SO	SB	CS	HBP	SF	RA	ER	ERA	CG	SHO
1871	NA	BS1	BNA		3	31		20	10			N		401	1372	426	70	37	3	60	19	73				303	109	3.55	22	1
1871	NA	CH1	CNA		2	28		19	9			N		302	1196	323	52	21	10	60	22	69				241	77	2.76	25	0
1871	NA	CL1	CFC		8	29		10	19			N		249	1186	328	35	40	7	26	25	18				341	116	4.11	23	0
1871	NA	FW1	KEK		7	19		7	12			N		137	746	178	19	8	2	33	9	16				243	97	5.17	19	1
1871	NA	NY2	NNA		5	33		16	17			N		302	1404	403	43	21	1	33	15	46				313	121	3.72	32	1
1871	NA	PH1	PNA		1	28		21	7			Y		376	1281	410	66	27	9	46	23	56				266	137	4.95	27	0
1871	NA	RC1	ROK		9	25		4	21			N		231	1036	274	44	25	3	38	30	53				287	108	4.3	23	1
1871	NA	TRO	TRO		6	29		13	15			N		351	1248	384	51	34	6	49	19	62				362	153	5.51	28	0
1871	NA	WS3	OLY		4	32		15	15			N		310	1353	375	54	26	6	48	13	48				303	137	4.37	32	0
1872	NA	BL1	BLC		2	58		35	19			N		617	2576	747	94	35	14	27	28	35	15			434	173	3.02	48	1
1872	NA	BR1	ECK		9	29		3	26			N		152	1075	235	26	6	0	14	29	8	4			413	165	5.73	28	0
1872	NA	BR2	BRA		6	37		9	28			N		237	1466	370	46	10	0	19	24	17	14			473	189	5.06	37	0
1872	NA	BS1	BNA		1	48		39	8			Y		521	2137	677	114	31	7	28	26	47	14			236	95	1.99	41	3
1872	NA	CL1	CFC		7	22		6	16			N		174	935	272	37	6	0	17	13	12	3			254	101	4.57	15	0
1872	NA	MID	MAN		8	24		5	19			N		220	1014	305	31	8	1	5	12	5	3			348	140	5.97	22	0
1872	NA	NY2	NNA		3	56		34	20			N		523	2423	668	86	13	4	55	52	56	21			362	145	2.55	54	3
1872	NA	PH1	PNA		4	47		30	14			N		539	2140	678	77	23	4	69	47	58	32			349	140	3.01	47	1
1872	NA	TRO	TRO		5	25		15	10			N		273	1124	329	56	11	5	7	14	9	7			191	77	3.08	17	2
1872	NA	WS3	OLY		10	9		2	7			N		54	363	93	12	3	0	4	4	0	3			140	56	6.38	9	0
1872	NA	WS4	NAT		11	11		0	11			N		80	460	105	3	2	0	1	3	0	0			190	76	6.91	11	0
1873	NA	BL1	BLC		3	57		34	22			N		644	2563	810	119	39	12	40	25	21	10			451	170	3.01	55	1
1873	NA	BL4	MAR		9	6		0	6			N		26	211	33	5	0	0	0	0	0	0			152	48	8	6	0
1873	NA	BR2	BRA		6	55		17	37			N		366	2210	588	60	27	6	53	43	18	9			549	221	3.98	52	1
1873	NA	BS1	BNA		1	60		43	16			Y		739	2755	930	137	46	13	62	24	39	27			460	154	2.59	48	1
1873	NA	ELI	RES		8	23		2	21			N		98	868	204	21	8	0	8	22	2	1			299	74	3.22	22	0
1873	NA	NY2	NNA		4	53		29	24			N		424	2214	622	67	41	5	42	22	15	5			385	139	2.62	48	2

Figure 2: RAW CSV files from Lahman's Baseball Database

comparedTeam	League	Wins	SimiliartyHitting	SimilarityPitching	BABIP	wOBA	ERA	FIP
2015 Philadelphia Phillies	NL	63.0	100	100	100.12537314986069	95.07158975706152	118.58567448182127	159.23751531823936
1960 Boston Red Sox	AL	68.37662337662337	132.04522302558405	60.879989363497813	102.9193264850769	102.15517728662043	121.02258566978192	111.43093678657014
1960 Chicago White Sox	AL	91.51948051948052	202.76918456789906	-45.064712873660142	104.15468903658368	104.28573917263878	94.30331350892098	109.99457567944798
1960 Chicago Cubs	NL	62.307692307692314	121.08313674171829	31.087327055119303	97.79030374965913	96.29884666084976	113.94983715661284	120.05547799474164
1960 Cincinnati Reds	NL	70.48051948051948	164.63821476574878	-42.173988424985282	98.71520556119533	98.94912741383536	104.78145945435664	93.30130127374814
1960 Cleveland Indians	AL	79.94805194805195	135.65904474758707	37.586174891336356	100.2597124226317	99.92422173377615	103.47169121117719	145.31233745069017
1960 Detroit Tigers	AL	74.68831168831169	183.22206782434608	-65.99076087649695	90.43266782898492	99.01370144423663	95.35112810346456	93.40960930027225
1960 Kansas City Athletics	AL	60.61935483870968	93.62179465150356	41.160861904743271	97.6227768242365	96.71398278694838	114.73569810252053	138.31257081356927
1960 Los Angeles Dodgers	NL	86.25974025974025	157.10895576795554	-58.013161696618454	101.27003042147898	99.70515230615723	89.06424053620316	84.70573863941982
1960 Milwaukee Braves	NL	92.57142857142857	188.65224105179618	-48.942760936715501	101.75338550010962	103.03629080775313	98.49457188709523	95.97611061690652
1960 New York Yankees	AL	101.38064516129033	209.73926755258265	-50.582958312662186	98.99204184694769	104.83888017672378	92.20768431983385	116.95233707318515
1960 Philadelphia Phillies	NL	62.06493506493507	67.68785460807155	-38.067705507315736	101.19883049383323	92.07912144244835	105.04341310299255	93.48760476518552
1960 Pittsburgh Pirates	NL	99.29032258064517	125.62422797698986	-132.17066014553893	108.21842796321175	103.98667058518947	91.42182337392619	52.59677058920704
1960 San Francisco Giants	NL	82.03846153846153	147.6163476619849	-98.593091601292628	100.95919839783636	98.91609100601585	90.11205513074671	66.28213586903013
1960 St. Louis Cardinals	NL	89.88387096774193	126.99919390464298	-53.122856662319947	99.08342749211596	99.96072516514924	95.35112810346456	81.72758380833072
1960 Washington Senators	AL	76.79220779220779	148.3426214184574	-29.366021127326633	96.35241898132627	99.4079037736419	98.75652553573114	101.95423441983667
1961 Baltimore Orioles	AL	94.41717791411043	120.64681982823159	-135.65206308344784	99.76040056401068	98.97454057730334	79.9884623268617	69.44318912415554
1961 Boston Red Sox	AL	75.53374233128834	122.22610902852449	43.012613456277904	100.66500296674039	98.8111441109582	106.56847931125364	127.37148598901258
1961 Chicago White Sox	AL	85.47239263803681	150.23997081939558	-40.78779485186908	98.83008546802171	100.3371101585453	100.85501771647778	95.65589587192882
1961 Chicago Cubs	NL	66.46153846153845	166.23719886106284	-3.9495061360862849	101.91843359680158	101.20730058669704	111.28829541128584	105.73659994050182
1961 Cincinnati Reds	NL	97.83116883116884	153.5453929246728	-71.385550083489989	103.19187038956288	101.30316678740547	93.89949925327242	92.30823044066882
1961 Cleveland Indians	AL	78.48447204968944	187.23629539736663	0.55286464029748572	100.6771317350208	100.16041625699297	103.09072007965096	125.43346135488325
1961 Detroit Tigers	AL	100.38036809815951	265.2584676459032	-97.316451879639374	102.38993610137453	105.55348442824423	88.18603765849659	96.17083579573104
1961 Kansas City Athletics	AL	61.0	136.0070642039573	50.845083177620012	97.51822477727892	94.52123482545628	117.7469911271194	117.97391487521622
1961 Los Angeles Angels	AL	70.0	198.35154600046834	54.545576366578771	97.36079996121802	100.79850132936711	107.06530205862543	127.07361366507541
1961 Los Angeles Dodgers	NL	93.62337662337663	187.7087240674159	-19.60488723162058	100.72298457957581	102.6085646002844	100.35819496910598	83.39607953689179
1961 Minnesota Twins	AL	70.43478260869566	118.94175168046209	-13.51837229343009	95.54052638247276	99.45341045173282	106.32006793756771	100.80524913521604

Figure 3: Similarity CSV created from program

Each of the values were normalized with respect to the season league average, and scaled appropriately such that 100 represented the league average. The similarity score is represented such that 100 defines identical performance (ie. the first team has a 100, 100 for batting, pitching because they are comparing to themselves). The interpretation for the normalize scores for the sabremetrics can be described as

- Wins: Higher = better
- SimilarityHitting: > 100 = better than target team
- SimilarityPitching: < 100 = better than target team
- BABIP, WOBA: > 100 = better than league average
- ERA, FIP, PBABIP, effERA: < 100 = better than league average

A preliminary examination of study showed that there are similarities between teams across time periods. A quick comparison between two teams from our first CSV file, 2012 Atlanta Braves, suggests that the 2012 Atlanta Braves and the 2014 Los Angeles Dodgers shared similar similarity scores (BS=107.38, PS=103.11). When we compare the two team's statistics online, we see at first glance that it doesn't seem very likely the two teams are similar.

Rk	Pos	Name	Age	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB
1	C	A.J. Ellis	33	93	347	283	21	54	9	0	3	25	0	0	53	57	.191	.323	.254	.577	68	72	15	4	3	4	5
2	1B	Adrian Gonzalez*	32	159	660	591	83	163	41	0	27	116	1	1	56	112	.276	.335	.482	.817	130	285	13	2	0	11	9
3	2B	Dee Gordon*	26	148	650	609	92	176	24	12	2	34	64	19	31	107	.289	.326	.378	.704	101	230	3	4	3	3	0
4	SS	Hanley Ramirez	30	128	512	449	64	127	35	0	13	71	14	5	56	84	.283	.369	.448	.817	132	201	10	6	0	1	2
5	3B	Juan Uribe	35	103	404	386	36	120	23	0	9	54	0	1	15	77	.311	.337	.440	.777	120	170	15	1	0	2	2
6	LF	Carl Crawford*	32	105	370	343	56	103	14	3	8	46	23	6	16	55	.300	.339	.429	.767	118	147	5	6	0	4	0
7	CF	Andre Ethier*	32	130	380	341	29	85	17	6	4	42	2	2	31	74	.249	.322	.370	.691	97	126	5	6	1	1	3
8	RF	Yasiel Puig	23	148	640	558	92	165	37	9	16	69	11	7	67	124	.296	.382	.480	.863	145	268	7	12	2	1	3

Figure 4: Dodgers batting

Rk	Pos	Name	Age	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB
1	C	Brian McCann*	28	121	487	439	44	101	14	0	20	67	3	0	44	76	.230	.300	.399	.698	87	175	15	1	0	3	7
2	1B	Freddie Freeman*	22	147	620	540	91	140	33	2	23	94	2	0	64	129	.259	.340	.456	.796	113	246	10	7	0	9	4
3	2B	Dan Uggla	32	154	630	523	86	115	29	0	19	78	4	3	94	168	.220	.348	.384	.732	98	201	8	10	0	3	5
4	SS	Paul Janish	29	55	186	167	18	31	6	1	0	9	1	0	17	30	.186	.269	.234	.502	38	39	1	2	0	0	0
5	3B	Chipper Jones#	40	112	448	387	58	111	23	0	14	62	1	0	57	51	.287	.377	.455	.832	124	176	15	1	0	3	6
6	LF	Martin Prado	28	156	690	617	81	186	42	6	10	70	17	4	58	69	.301	.359	.438	.796	114	270	19	2	4	9	2
7	CF	Michael Bourn*	29	155	703	624	96	171	26	10	9	57	42	13	70	155	.274	.348	.391	.739	99	244	2	3	2	4	1
8	RF	Jason Heyward*	22	158	651	587	93	158	30	6	27	82	21	8	58	152	.269	.335	.479	.814	117	281	4	2	0	3	1

Figure 5: Braves batting

Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
1	SP	Zack Greinke	30	17	8	.680	2.71	32	32	0	0	0	0	202.1	190	69	61	19	43	3	207	2	0	12	821	129	2.97	1.152	8.5	0.8	1.9	9.2	4.81
2	SP	Clayton Kershaw*	26	21	3	.875	1.77	27	27	0	6	2	0	198.1	139	42	39	9	31	0	239	2	2	7	749	197	1.81	0.857	6.3	0.4	1.4	10.8	7.71
3	SP	Dan Haren	33	13	11	.542	4.02	32	32	0	0	0	0	186.0	183	101	83	27	36	7	145	3	1	8	776	87	4.09	1.177	8.9	1.3	1.7	7.0	4.03
4	SP	Hyun-jin Ryu*	27	14	7	.667	3.38	26	26	0	0	0	0	152.0	152	60	57	8	29	2	139	3	0	2	631	103	2.62	1.191	9.0	0.5	1.7	8.2	4.79
5	SP	Josh Beckett	34	6	6	.500	2.88	20	20	0	1	1	0	115.2	96	41	37	17	39	2	107	5	0	1	475	121	4.33	1.167	7.5	1.3	3.0	8.3	2.74
Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
6	CL	Kenley Jansen	26	2	3	.400	2.76	68	0	57	0	0	44	65.1	55	20	20	5	19	2	101	0	0	2	268	127	1.91	1.133	7.6	0.7	2.6	13.9	5.32
7	RP	Jamey Wright	39	5	4	.556	4.35	61	1	22	0	0	1	70.1	72	35	34	3	27	4	54	4	0	5	308	80	3.47	1.408	9.2	0.4	3.5	6.9	2.00
8	RP	Brandon League	31	2	3	.400	2.57	63	0	12	0	0	0	63.0	65	23	18	0	27	5	38	4	0	4	273	136	3.40	1.460	9.3	0.0	3.9	5.4	1.41
9	RP	J.P. Howell*	31	3	3	.500	2.39	68	0	8	0	0	0	49.0	31	14	13	2	25	1	48	1	0	3	199	147	3.30	1.143	5.7	0.4	4.6	8.8	1.92
10	RP	Brian Wilson	32	2	4	.333	4.66	61	0	6	0	0	1	48.1	49	26	25	5	29	3	54	4	0	1	223	75	4.29	1.614	9.1	0.9	5.4	10.1	1.86
Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W

Figure 6: Dodgers pitching

Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
1	SP	Mike Minor*	24	11	10	.524	4.12	30	30	0	0	0	0	179.1	151	88	82	26	56	7	145	5	0	3	728	97	4.38	1.154	7.6	1.3	2.8	7.3	2.59
2	SP	Tim Hudson	36	16	7	.696	3.62	28	28	0	1	1	0	179.0	168	77	72	12	48	2	102	9	0	3	749	110	3.78	1.207	8.4	0.6	2.4	5.1	2.13
3	SP	Tommy Hanson	25	13	10	.565	4.48	31	31	0	0	0	0	174.2	183	95	87	27	71	5	161	5	0	6	761	89	4.57	1.454	9.4	1.4	3.7	8.3	2.27
4	SP	Randall Delgado	22	4	9	.308	4.37	18	17	0	0	0	0	92.2	89	48	45	8	42	4	76	4	1	5	401	92	4.07	1.414	8.6	0.8	4.1	7.4	1.81
5	SP	Brandon Beachy	25	5	5	.500	2.00	13	13	0	1	1	0	81.0	49	24	18	6	29	1	68	1	0	4	319	200	3.49	0.963	5.4	0.7	3.2	7.6	2.34
6	SP	Paul Maholm*	30	4	5	.444	3.54	11	11	0	1	1	0	68.2	63	29	27	8	19	1	59	1	0	1	283	113	3.76	1.194	8.3	1.0	2.5	7.7	3.11
Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
7	CL	Craig Kimbrel	24	3	1	.750	1.01	63	0	56	0	0	42	62.2	27	7	7	3	14	0	116	2	0	5	231	399	0.78	0.654	3.9	0.4	2.0	16.7	8.29
8	RP	Cristhian Martinez	30	5	4	.556	3.91	54	0	26	0	0	1	73.2	80	33	32	6	19	4	65	0	1	2	313	103	3.16	1.344	9.8	0.7	2.3	7.9	3.42
9	RP	Chad Durbin	34	4	1	.800	3.10	76	0	19	0	0	1	61.0	52	25	21	9	28	3	49	0	0	1	257	130	4.78	1.311	7.7	1.3	4.1	7.2	1.75
10	RP	Jonny Venters*	27	5	4	.556	3.22	66	0	12	0	0	0	58.2	61	23	21	6	28	3	69	5	0	9	262	125	3.76	1.517	9.4	0.9	4.3	10.6	2.46
11	RP	Eric O'Flaherty*	27	3	0	1.000	1.73	64	0	7	0	0	0	57.1	47	14	11	3	19	2	46	2	0	1	230	233	3.27	1.151	7.4	0.5	3.0	7.2	2.42

Figure 7: Braves pitching

However, upon closer examination, looking beyond traditional statistics and into where the teams stood during the season, it becomes more apparent the similarity between the two teams.

Figure 8: Comparing 2012 Dodgers and 2014 Braves source

This shows that we are on the right track. Now, we check our similarity scores using visualization. We graph our sabremetric values and investigate any groupings. From there, we then derive some form of similarity and compare these clusters with the the similarity scores obtained from our algorithm.

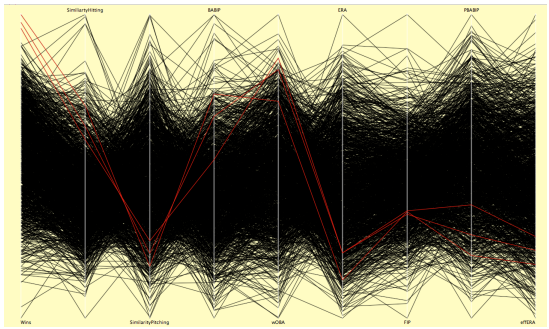
Note: team scores accounts for the average, so when we transition to player comparisons, it'll be slightly different. For teams, because the statistics of the entire team is what is important, having 2 aces (Dodgers) vs. 4 good starters (Braves) resulted in the same score. In the case of individual players, we go by season statistics. However, we can take a step further than teams (single data point) by isolating each individual year as a separate data point and querying across those years, rather than taking the average. This will give us a better result.

5 Visualization

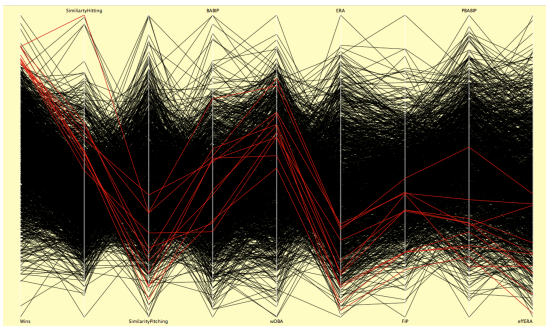
We then visualized our results (both sabremetrics and the similarities results). Specifically, we took our sabremetric results and applied similarities in groupings, and the compared them to our derived numerical similarity results.

5.1 Mondrian

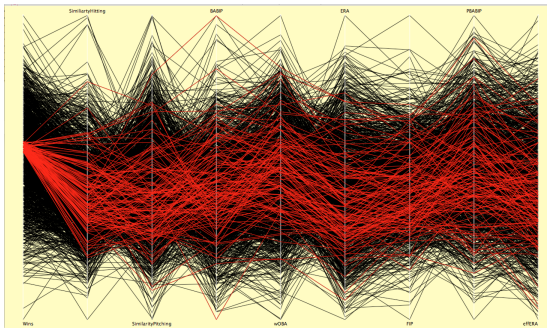
I applied my similarity results to Mondrian and looked at the associations or patterns.



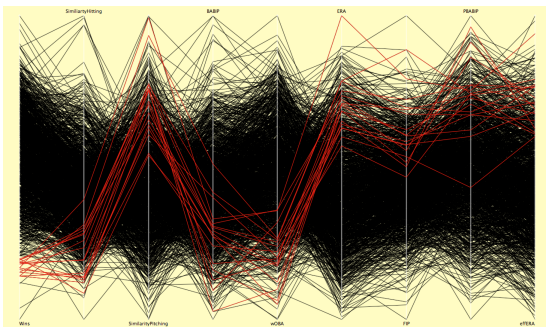
(a) Mondrian high wins



(b) Mondrian high wins



(c) Mondrian median wins



(d) Mondrian low wins

Figure 9: Similarity scores visualized on Mondrian

The map from left to right reads – Wins, similarity hitting scores, similarity pitching scores, BABIP, wOBA, ERA, FIP, PBABIP, and effERA. You will notice that like those with high wins in the season standings, they share very similar offensive statistics and pitching statistics. This is the same case for those with low wins in the season standings. It is interesting to note that the higher Wins teams tended to have a much better pitching score than batting score, suggesting the importance of pitching over batting. This was not apparent in the lower Wins teams – they had equally bad hitting and pitching. For those

in the middle of the pack, around the average, they're offensive and pitching statistics were varied.

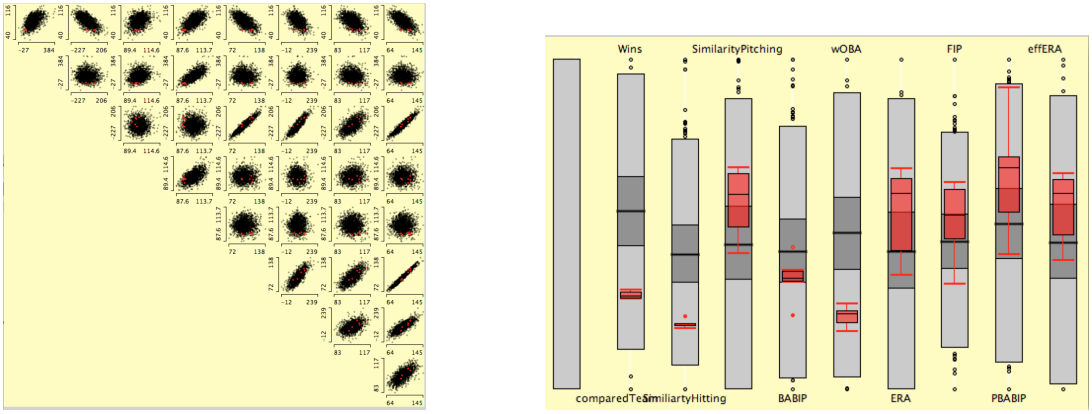
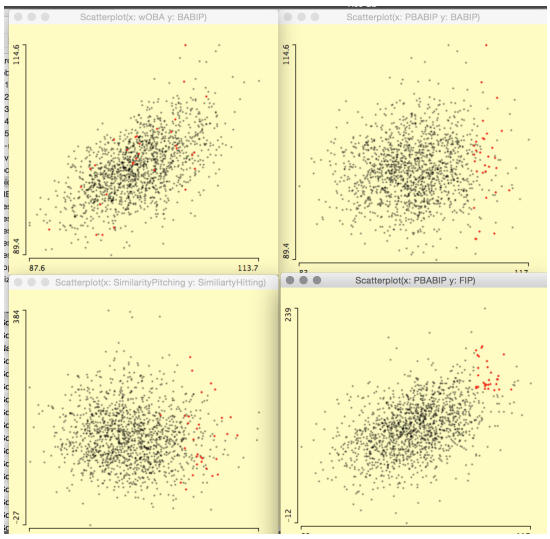
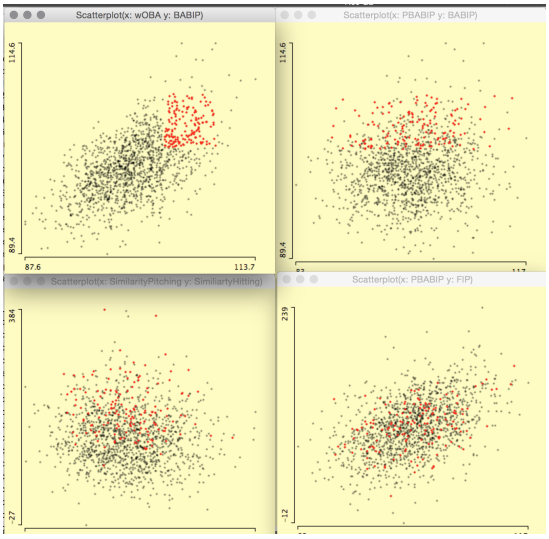


Figure 10: 2014 Results in various binnings. Note both the median or average wins had a very large distribution of the other categories

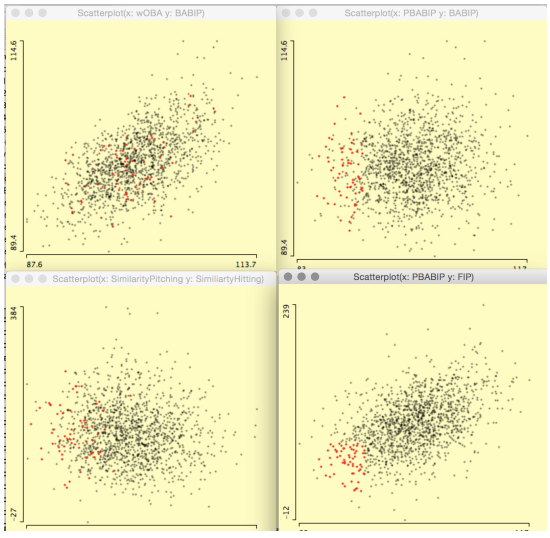
If we take a closer look and examine the offensive stats in relation to the pitching stats and similarity stats, you will notice a relatively week association, but nevertheless an association.



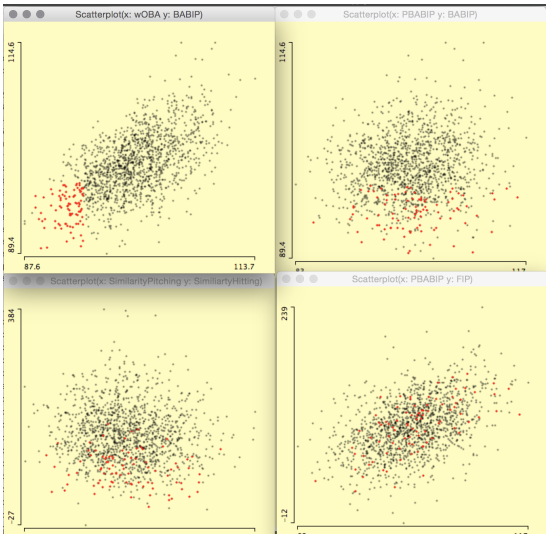
(a) bad pitching



(b) good batting



(c) good pitching



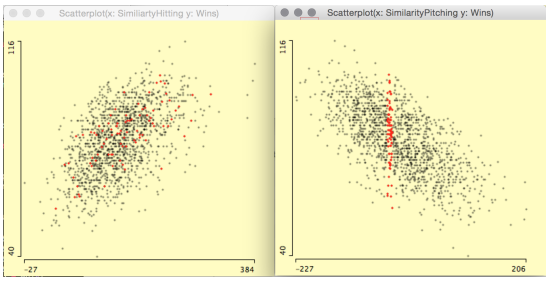
(d) poor batting

Figure 11: Similarity scores visualized on Mondrian

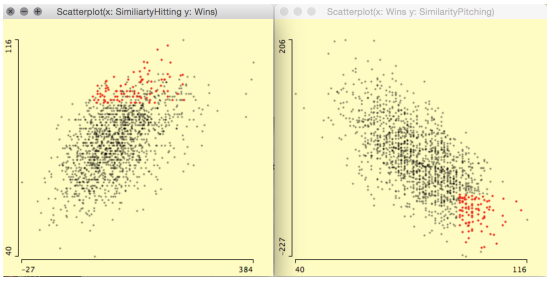
where in each subfigure in figure 11 has uniquely

- top left - BABIP vs. wOBA
- top right - BABIP vs. PBABIP
- bottom left - SimilarityHitting vs.SimilarityPitching
- bottom right - FIP vs. PBABIP

and remember that 100 is the norm, relative to the league average. For pitching, it is generally under 100 that performance is above league average, while for batting it is generally above 100 that performance is above league average. For similarity scores,



(a) Showing 1 to 1 independency between hitting and pitching scores

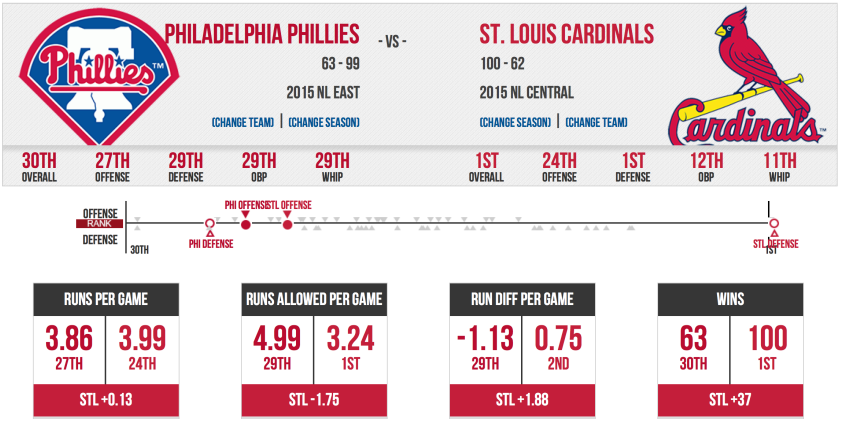


(b) Showing commonalities between scores and wins

Figure 12: Similarity score to team wins

100 represents perfectly identical teams. We can see from our interpretations that more win teams tend to have higher batting similarity scores (> 100) and lower pitching similarity score (< 100). We also see that having nearly identical similarity scores in pitching or batting does not necessarily translate to batting or pitching similarity scores, which is a positive sign that these variables are independent. However, one can see that those teams with a good win-loss record tended to have high batting similarity scores, and low pitching similarity scores. It is important to see that although this is generally the trend, the spread is quite large, suggesting that teams can have a high win-loss record with good scores on one end. An example would be the 2015 St. Louis Cardinals with a 100 win record and a -156 pitching similarity score, but having an almost average 108 batting similarity score.

Looking at their traditional statistics and comparing with the 2015 Philadelphia Phillies (our reference), we see that the Cardinals had league average batting with league leading pitching. A quick comparison across multiple sites would confirm our findings.



Rk	Pos	Name	Age	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB
1	C	Yadier Molina	32	136	530	488	34	132	23	2	4	61	3	1	32	59	.270	.310	.350	.660	80	171	16	0	1	9	3
2	1B	Mark Reynolds	31	140	432	382	35	88	21	2	13	48	2	3	44	121	.230	.315	.398	.713	93	152	10	4	0	2	2
3	2B	Kolten Wong*	24	150	613	557	71	146	28	4	11	61	15	8	36	95	.262	.321	.386	.707	93	215	10	15	0	5	2
4	SS	Jhonny Peralta	33	155	640	579	64	159	26	1	17	71	1	4	50	111	.275	.334	.411	.745	103	238	23	5	0	6	6
5	3B	Matt Carpenter*	29	154	665	574	101	156	44	3	28	84	4	3	81	151	.272	.365	.505	.871	135	290	5	6	0	4	5
6	LF	Matt Holliday	35	73	277	229	24	64	16	1	4	35	2	1	39	49	.279	.394	.410	.804	121	94	9	6	0	3	5
7	CF	Peter Bourjos	28	117	225	195	32	39	8	3	4	13	5	8	19	59	.200	.290	.333	.623	70	65	2	6	4	1	4
8	RF	Jason Heyward*	25	154	610	547	79	160	33	4	13	60	23	3	56	90	.293	.359	.439	.797	117	240	13	2	0	3	4

Figure 13: 2015 Cardinals Batting

Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
1	SP	John Lackey	36	13	10	.565	2.77	33	33	0	1	0	0	218.0	211	71	67	21	53	5	175	4	3	5	896	142	3.57	1.211	8.7	0.9	2.2	7.2	3.30
2	SP	Michael Wacha	23	17	7	.708	3.38	30	30	0	0	0	0	181.1	162	74	68	19	58	4	153	6	1	4	762	116	3.87	1.213	8.0	0.9	2.9	7.6	2.64
3	SP	Carlos Martinez	23	14	7	.667	3.01	31	29	1	0	0	0	179.2	168	65	60	13	63	5	184	8	1	8	755	130	3.21	1.286	8.4	0.7	3.2	9.2	2.92
4	SP	Lance Lynn	28	12	11	.522	3.03	31	31	0	0	0	0	175.1	172	66	59	13	68	5	167	5	0	2	751	129	3.44	1.369	8.8	0.7	3.5	8.6	2.46
5	SP	Jaime Garcia*	28	10	6	.625	2.43	20	20	0	0	0	0	129.2	106	37	35	6	30	0	97	3	0	4	510	161	3.00	1.049	7.4	0.4	2.1	6.7	3.23
Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
6	CL	Trevor Rosenthal	25	2	4	.333	2.10	68	0	57	0	0	48	68.2	62	16	16	3	25	3	83	1	0	7	287	188	2.42	1.267	8.1	0.4	3.3	10.9	3.32
7	RP	Kevin Siegrist*	25	7	1	.875	2.17	81	0	13	0	0	6	74.2	53	20	18	4	34	2	90	3	0	0	312	181	2.91	1.165	6.4	0.5	4.1	10.8	2.65
8	RP	Seth Maness	26	4	2	.667	4.26	76	0	13	0	0	3	63.1	77	35	30	7	13	4	46	1	0	2	270	92	3.78	1.421	10.9	1.0	1.8	6.5	3.54
9	RP	Carlos Villanueva	31	4	3	.571	2.95	35	0	20	0	0	2	61.0	50	21	20	6	21	2	55	2	0	1	250	133	3.74	1.164	7.4	0.9	3.1	8.1	2.62
10	RP	Randy Choate*	39	1	0	1.000	3.95	71	0	4	0	0	1	27.1	29	14	12	2	5	0	22	6	0	1	117	100	3.68	1.244	9.5	0.7	1.6	7.2	4.40

Figure 14: 2015 Cardinals Pitching

Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
1	SP	Aaron Harang	37	6	15	.286	4.86	29	29	0	0	0	0	172.1	189	100	93	26	51	3	108	6	0	4	748	82	4.83	1.393	9.9	1.4	2.7	5.6	2.12
2	SP	Cole Hamels*	31	6	7	.462	3.64	20	20	0	1	1	0	128.2	113	53	52	12	39	3	137	6	2	7	537	109	3.27	1.181	7.9	0.8	2.7	9.6	3.51
3	SP	Jerome Williams	33	4	12	.250	5.80	33	21	3	0	0	1	121.0	161	83	78	22	34	3	74	5	0	4	553	68	5.24	1.612	12.0	1.6	2.5	5.5	2.18
4	SP	Adam Morgan*	25	5	7	.417	4.48	15	15	0	0	0	0	84.1	88	45	42	14	17	0	49	4	1	2	352	89	4.88	1.245	9.4	1.5	1.8	5.2	2.88
5	SP	Aaron Nola	22	6	2	.750	3.59	13	13	0	0	0	0	77.2	74	31	31	11	19	1	68	2	0	0	318	111	4.04	1.197	8.6	1.3	2.2	7.9	3.58
6	SP	David Buchanan	26	2	9	.182	6.99	15	15	0	0	0	0	74.2	109	60	58	12	29	1	44	3	0	2	351	57	5.33	1.848	13.1	1.4	3.5	5.3	1.52
Rk	Pos	Name	Age	W	L	W-L%	ERA	G	GS	GF	CG	SHO	SV	IP	H	R	ER	HR	BB	IBB	SO	HBP	BK	WP	BF	ERA+	FIP	WHIP	H9	HR9	BB9	SO9	SO/W
7	CL	Jonathan Papelbon	34	2	1	.667	1.59	37	0	34	0	0	17	39.2	31	9	7	3	8	1	40	4	0	0	161	251	3.01	0.983	7.0	0.7	1.8	9.1	5.00
8	RP	Justin De Fratus	27	0	2	.000	5.51	61	0	16	0	0	0	80.0	92	52	49	9	32	2	68	5	0	7	362	72	4.28	1.550	10.4	1.0	3.6	7.7	2.13
9	RP	Jeanmar Gomez	27	2	3	.400	3.01	65	0	21	0	0	0	74.2	82	28	25	4	17	4	50	2	0	3	319	132	3.25	1.326	9.9	0.5	2.0	6.0	2.94
10	RP	Ken Giles	24	6	3	.667	1.80	69	0	28	0	0	15	70.0	59	23	14	2	25	2	87	1	0	1	298	221	2.13	1.200	7.6	0.3	3.2	11.2	3.48
11	RP	Luis Garcia	28	4	6	.400	3.51	72	0	14	0	0	2	66.2	72	28	26	4	37	8	63	0	1	6	304	113	3.69	1.635	9.7	0.5	5.0	8.5	1.70

Figure 15: 2015 Phillies Batting

Rk	Pos	Name	Age	G	PA	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG	OPS	OPS+	TB	GDP	HBP	SH	SF	IBB
1	C	Carlos Ruiz	36	86	320	284	23	60	13	1	2	22	1	1	28	43	.211	.290	.285	.575	59	81	13	4	3	1	2
2	1B	Ryan Howard*	35	129	503	467	53	107	29	1	23	77	0	0	27	138	.229	.277	.443	.720	94	207	11	5	0	3	2
3	2B	Cesar Hernandez#	25	127	452	405	57	110	20	4	1	35	19	5	40	86	.272	.339	.348	.687	90	141	6	2	4	1	1
4	SS	Freddy Galvis#	25	151	603	559	63	147	14	5	7	50	10	1	30	103	.263	.302	.343	.645	77	192	11	3	7	4	1
5	3B	Maikel Franco	22	80	335	304	45	85	22	1	14	50	1	0	26	52	.280	.343	.497	.840	127	151	8	4	0	1	2
6	LF	Cody Asche*	25	129	456	425	41	104	22	3	12	39	1	2	26	111	.245	.294	.395	.689	87	168	4	4	0	1	3
7	CF	Odubel Herrera*	23	147	537	495	64	147	30	3	8	41	16	8	28	129	.297	.344	.418	.762	108	207	6	8	5	1	0
8	RF	Jeff Francoeur	31	119	343	326	34	84	16	1	13	45	0	2	13	77	.258	.286	.433	.718	94	141	10	1	0	3	0

Figure 16: 2015 Phillies Pitching

5.1.1 Checking effectiveness of our effERA measurement

To check how effective our effERA sabremetric was, we looked at the scatterplot of effERA and ERA.

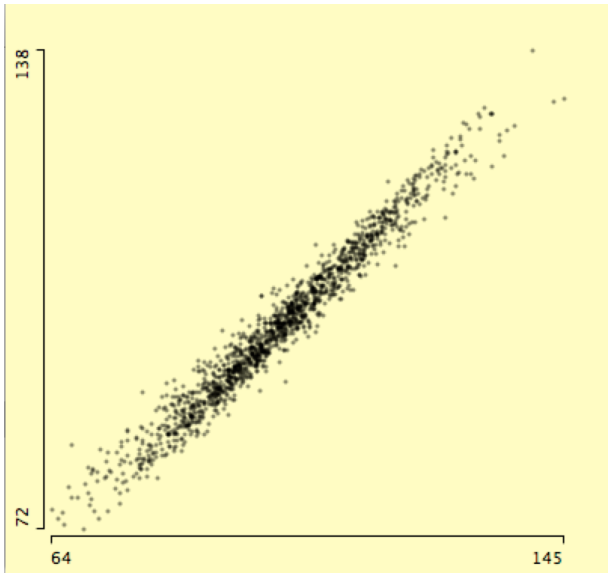


Figure 17: effERA vs. ERA

The scatterplot shows a strong positive linear correlation between ERA and effERA, which is good and should suggest a reasonable statistic. What is interesting is that when we look at the extrema, we can see that the ends are rather less strongly correlated, or "opening up". This is interesting as this can suggest that really good or really bad ERAs put up by pitchers (and the team) could be due to other factors (ie. relief pitchers giving up runs left stranded by starters or fielders with realistically bad fielding, the opposite case for the two, or just being rather unlucky). The incorporation of SVs, CG, SHO, and FLD helps identify these cases. It seems that the effERA is a rather sufficient tool in analyzing true team ERA, and can be applied to individual pitchers in the future. The coefficients are rather arbitrary, and will need some adjustments later down the road.

5.2 Paraview

From here, we then looked into another visualization program, Paraview. Although our dataset is not time-dependent, given that each dataset is by individual year, we examine our team set results using a selection of tools.

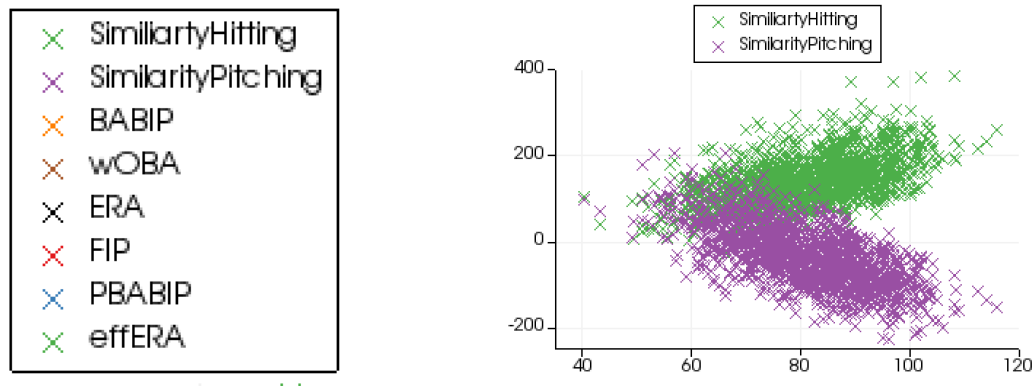
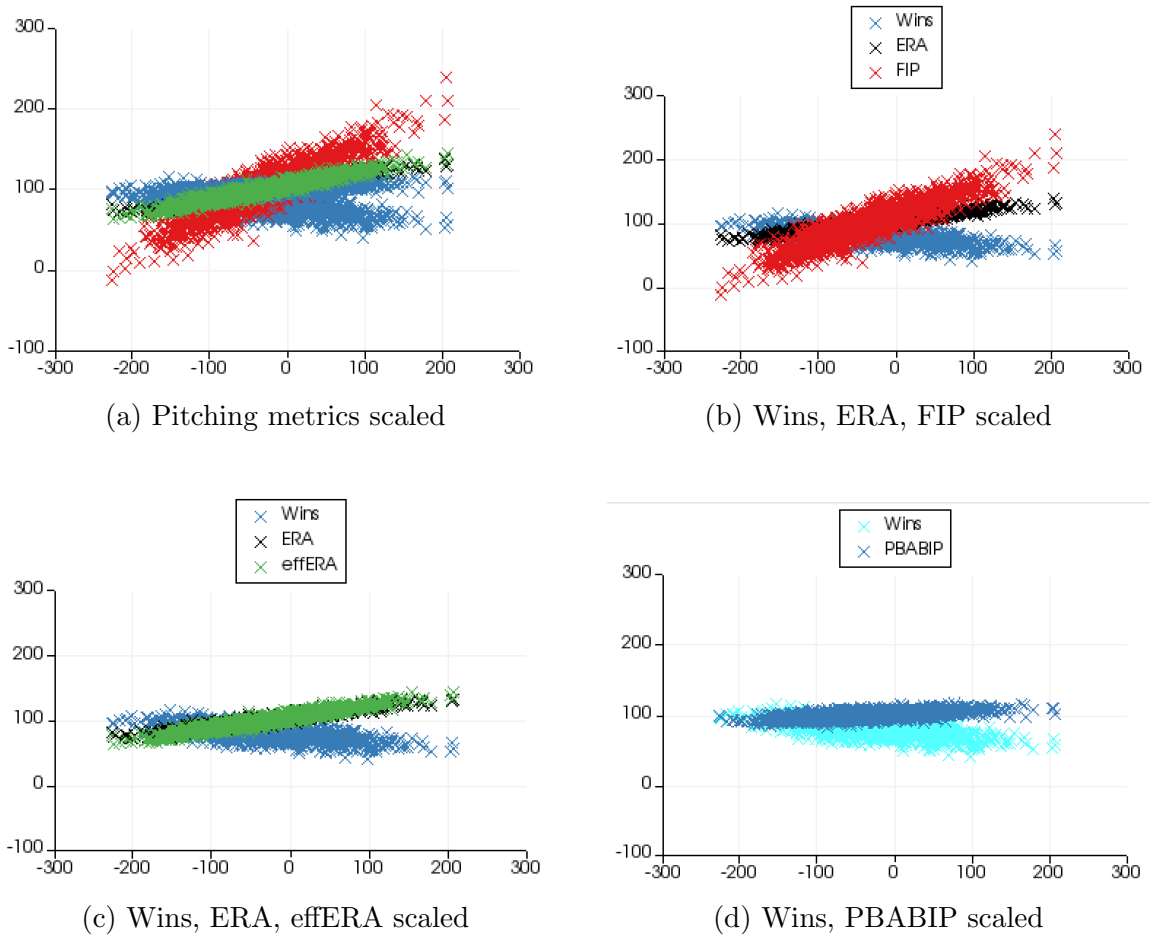


Figure 18: Similarity score to team wins

We see the very clear association and (because these are numerical or quantitative), very clear correlation between similarity hitting and similarity pitching and team wins. Note that the correlation may not be as strong because of the relative spread of the data points along the regression fit.



(a) Pitching metrics scaled

(b) Wins, ERA, FIP scaled

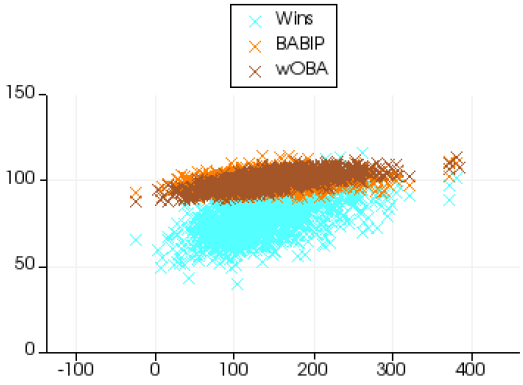
(c) Wins, ERA, effERA scaled

(d) Wins, PBABIP scaled

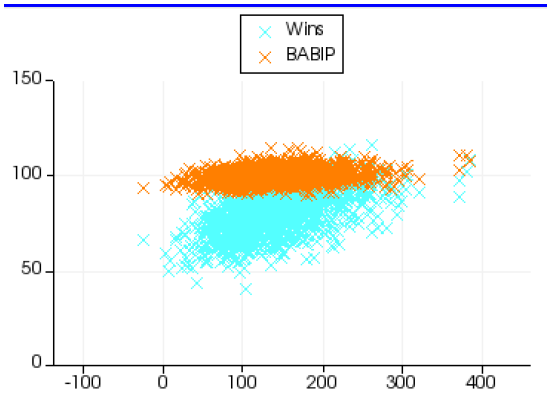
Figure 19: Similarity scores visualized on Paraview scaled to team pitching similarity scores

We see that the standard metrics with respect to the pitching similarity scores*, do not have a strong association, meaning they are not as good of an indicator to team performance. The sabremetrics however, have mixed results, with FIP being a great metric in associating team performance to pitching similarity scores. This is a much larger spread than the standard metrics, which suggests that we removed much of the noise. In addition, the effERA is nearly the same as the ERA in determining team performance near the middle, but differs from ERA at the extremas, potentially removing some noise in the ends due to bad luck or good luck or other environmental factors.

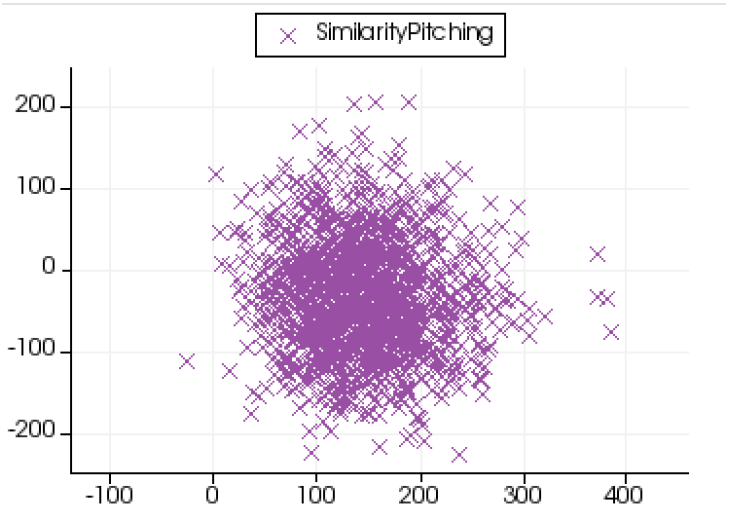
** to the 2015 Philadelphia Phillies*



(a) Wins, BABIP, wOBA



(b) Wins, BABIP

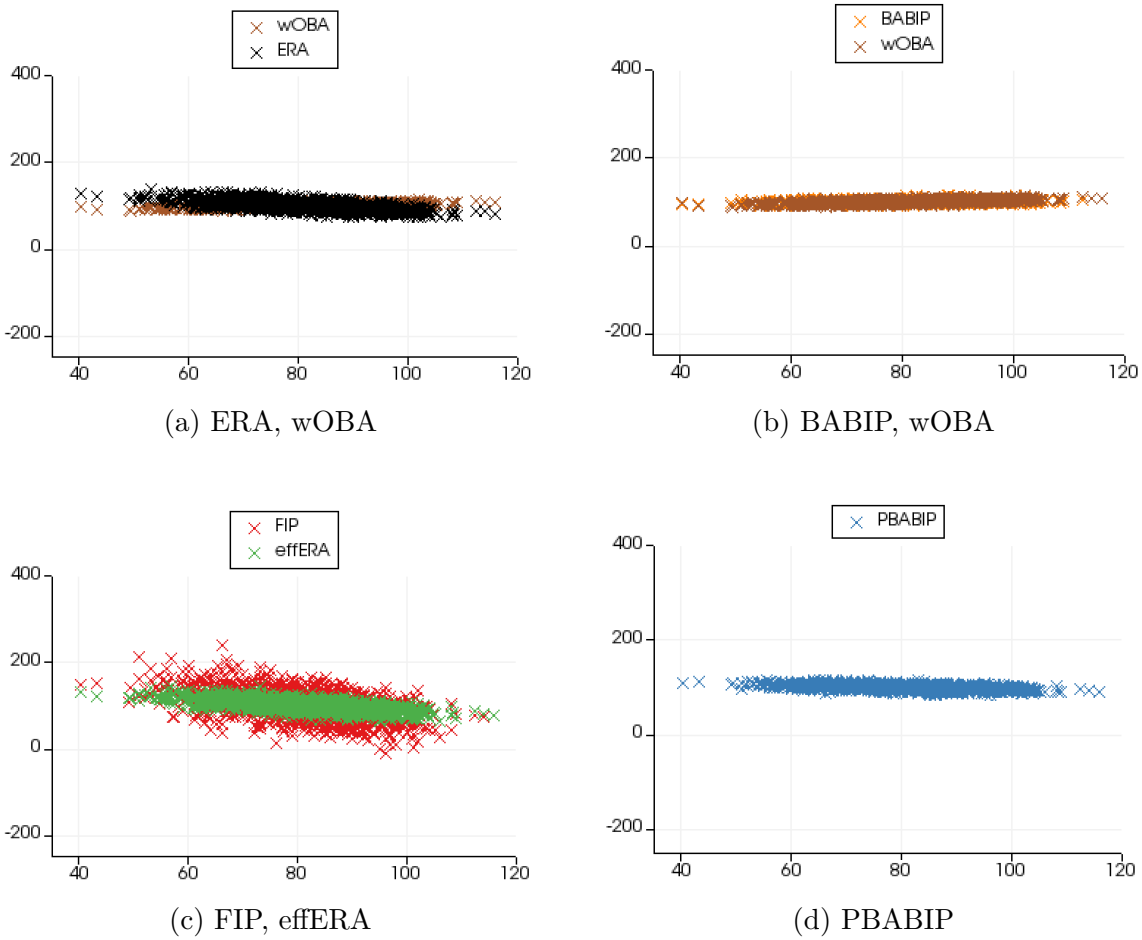


(c) Similarity Pitching Scores

Figure 20: Similarity scores visualized on Paraview scaled to team batting similarity scores

We see now when we shift to similarity batting, the Wins is the only one with a deep association but at the cost of a wider spread. The BABIP and wOBA have a much shallower asssociation, but the spread is much tighter. When we look at the similarity pitching teams to the similarity batting, we see a good sign – the association between the two is very weak. This confirms our scores calculated are independent.

Now, we centralize them to wins, such that the x-axis is team wins. Consider the results,



(a) ERA, wOBA

(b) BABIP, wOBA

(c) FIP, effERA

(d) PBABIP

Figure 21: Similarity scores visualized on Paraview scaled to team wins in a 162 game season

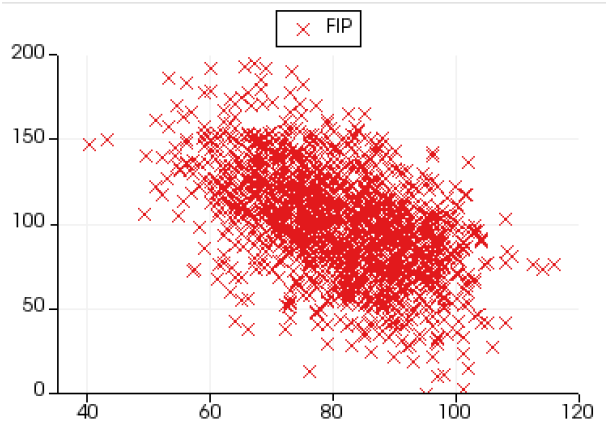


Figure 22: FIP vs. 162 expected team wins

The FIP has the steepest association, but along with it's strength also has the largest

spread.

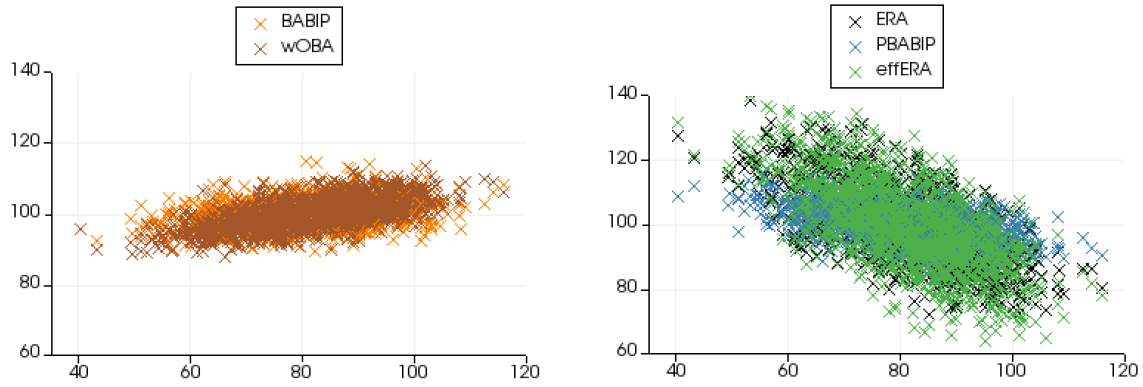


Figure 23: More similarity scores visualized on Paraview scaled to team wins in a 162 game season

We attempted to do a 4D vector representation in Paraview,

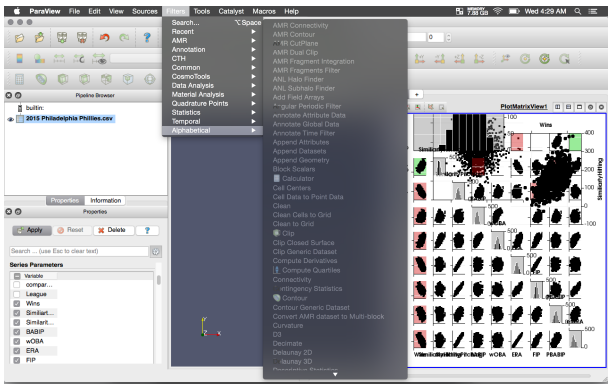
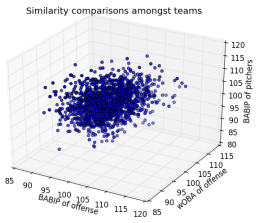


Figure 24: FIP vs. 162 expected team wins

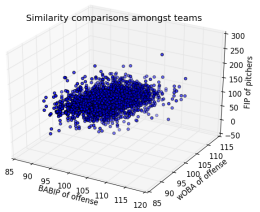
but could not after many attempts. Thus, we had to physically implement and draw from scratch in Python.

5.3 3D Plots

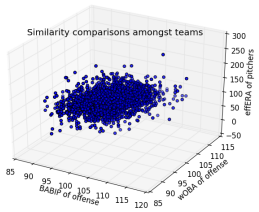
We then decided to visualize the calculated sabremetrics in three-dimensions to see if there were any clustering or similarities. First, we looked for pitching + batting team clusters.



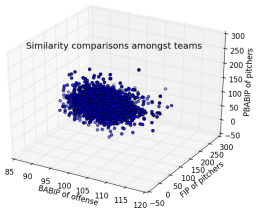
(a) BABIP, wOBA, PBABIP



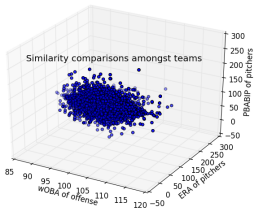
(b) BABIP, wOBA, FIP



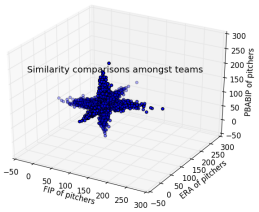
(c) BABIP, wOBA, effERA



(d) BABIP, FIP, PBABIP



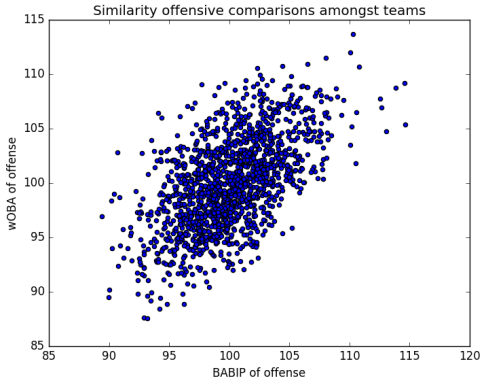
(e) wOBA, ERA, PBABIP



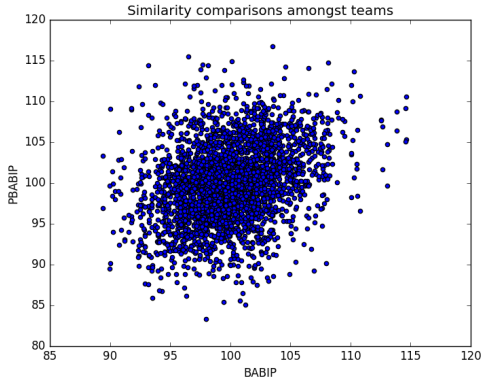
(f) FIP, ERA, PBABIP

Figure 25: Three Dimensional Visualization of Sabremetric statistics

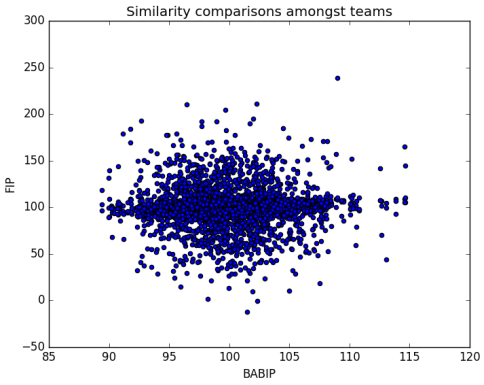
Looking at these plots, we can see that the data seems to be densely distributed, making any particular group hard to pull out. You will see that figure a has the highest chance of any particular clustering, while the rest seem to follow a normal distribution found in nature. Looking at purely the pitching statistics, we see that they are rather invariant of each other, all along the 100, 100, 100 axes. However, if you did make clustering groups, you'll find 6 neat K groups, but that does not give too much insight. We look into 2D visualizations next.



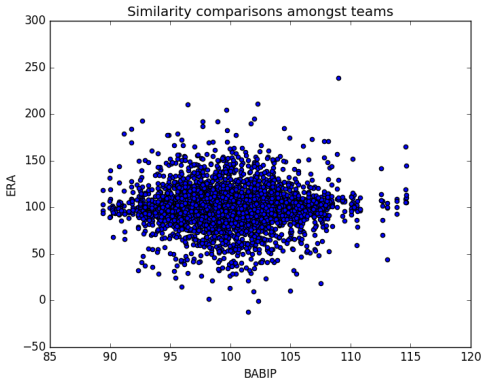
(a) wOBA, BABIP



(b) PBABIP, BABIP



(c) FIP, BABIP

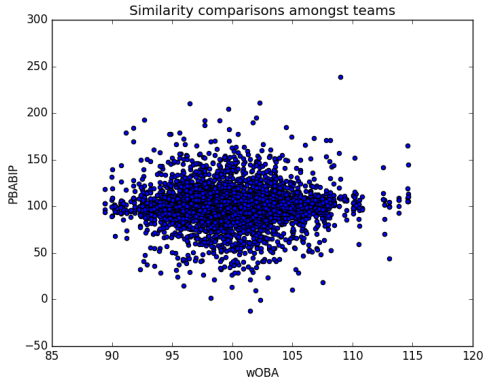


(d) ERA, BABIP

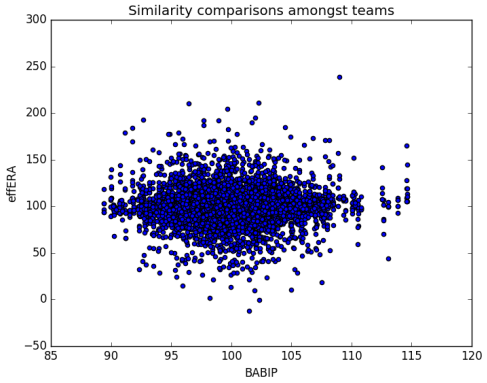
Figure 26: Two dimensional visualization of sabremetrics

We find that in the offensive sabremetric comparisons, there seems to be some form of clustering due to the relatively large distribution along high wOBA and high BABIP, as well as low wOBA and low BABIP. This also applies to all the pitching stats and BABIP, with the one exception being PBABIP vs. BABIP. In this scenario, it seems that the clustering less significant, as it is more radially spread out.

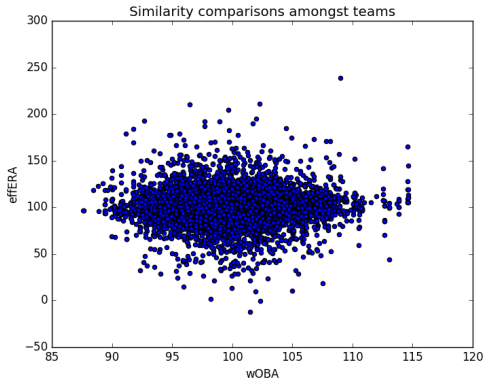
Looking at wOBA now



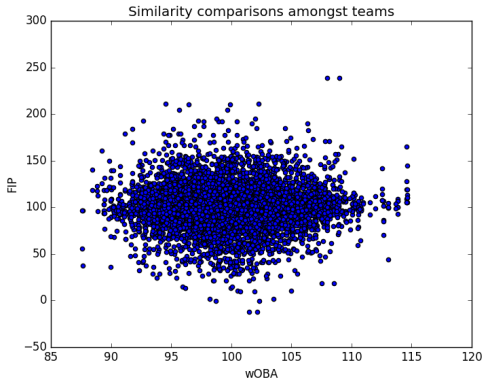
(a) PBABIP, wOBA



(b) effERA, BABIP



(c) effERA, wOBA



(d) FIP, wOBA

Figure 27: Two dimensional visualization of sabremetrics (Cont.)

we can see that it has similar relations to that of pitching sabremetrics with BABIP, which confirms that these stats are respective of their binary categories (pitching or hitting). It is interesting to note that FIP has a larger spread than the other pitching categories, and that the spread increases as we go from ERA (traditional) to the most extreme of sabremetrics (FIP). This likely shows that ERA suppresses information within the dataset. Furthermore, we note that many of the datapoints do not change – I think this is an error, and I will need to rerun again.

6 4D Plots

To further look into data, now comparing to wins, we consider a 4D dataset. We originally wanted to look at a 3D dataset, analyzing similarity wins to similarity pitching to wins

in a 162 game season, but instead, we are looking at an additional variable, BABIP. We chose BABIP because of it's moderate association relative to the rest of the dataset.

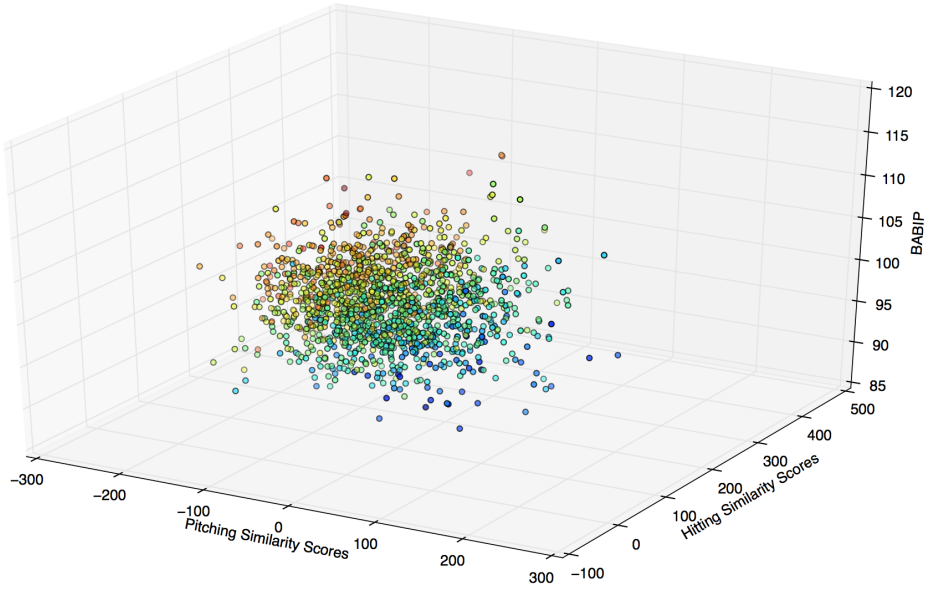


Figure 28: 4D visual representation of key baseball metrics

This plot is a preliminary plot showing the relative degree of information of from pitching similarity, batting similarity, BABIP, and color scaled by wins. You can see that the colors are grouped near each other, in sections. This is the first instance of potential success in k-clustering. So, we further examine using a 3D mapping with color for geography.

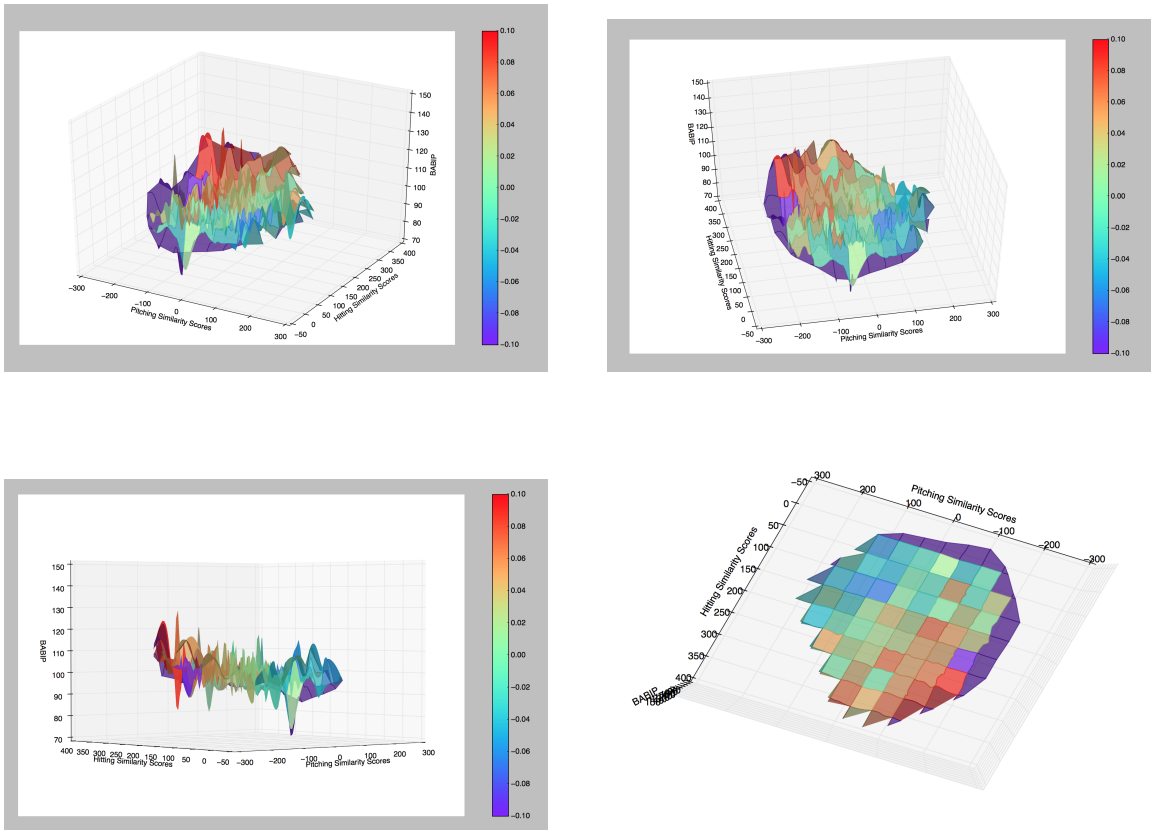


Figure 29: 4 dimensional visualization of sabremetrics$^+$

with axes being the same as above,

- X = pitching similarity scores
- Y = hitting similarity scores
- Z = normalized BABIP
- C = projected 162 season wins

and we something quite interesting here. We can see some significant clustering in win values along the X-Y plane and this is not significantly dependent on BABIP (our Z-axis). The coloring, representing wins, is group relative to one another, the most wins in particularly one corner, while the least wins wrapping around another corner. However, there is some significant clustering along the z-axis, too. This is a comparative sign that BABIP can be used for a clustering algorithm. It's interesting to observe the purple group or least wins group was isolated around the corners, and underneath the most wins groups. The teams with average wins varied around the center and surface, while the

most and least wins encompassed the corners.

Note the scale is off on the color in terms of value. I attempted to fix it for a couple hours and had no avail. The coloring and grouping is correct, but the display and values is not. The red or top part of the color scheme represents the most wins, while the bottom part of the color scheme or purple represented the least projected wins in a 162 game season.

+See full graphic picture below. The visualization is clear and the interpretation is stunning.

7 K-means Clustering

Because I saw some form of partition within our data, I decided to see if there were an classifications of the dataset by doing a K-means Clustering, specifically focusing on the offensive statistics. The k-means algorithm takes a dataset X of N points as input, and makes K clusters to create. The output is a set of K cluster centroids and each cluster containing datapoints from our dataset X.

Mathematically, this means

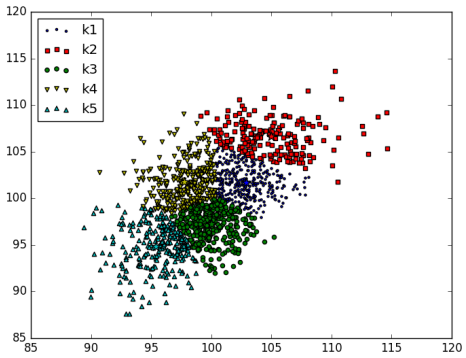
$$Min \sum_{k=1}^{K} \sum_{x_n \epsilon C_k} \|x_n - \mu_k\|^2 \text{ with respect to } C_k, \mu_k \tag{7.0.1}$$

Specifically, I used Lloyd's algorithm [4][5] which can be expressed as so:

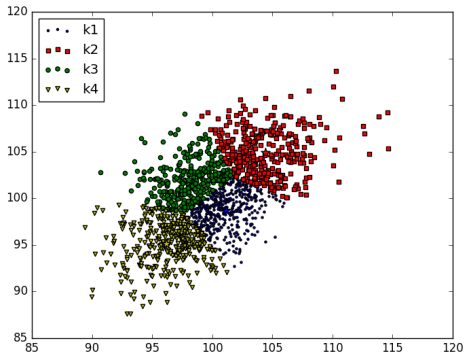
$$C_k = \{\mathrm{x}_n : ||\mathrm{x}_n - \mu_k|| \leq \text{ all } ||\mathrm{x}_n - \mu_l||\}$$

$$\mu_k = \frac{1}{C_k} \sum_{\mathrm{x}_n \in C_k} \mathrm{x}_n$$

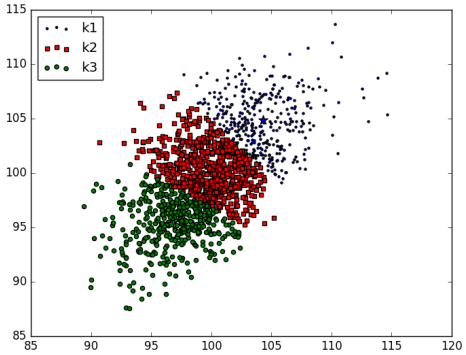
and by partitioning from $K = 3$ to $K = 6$, we get the following



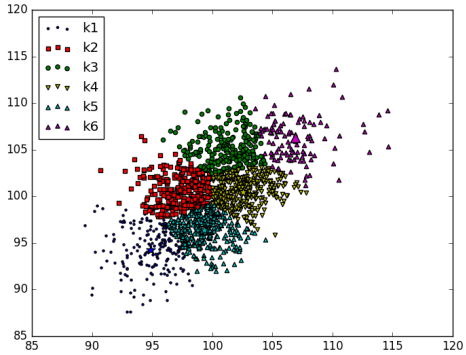
(a) K = 5



(b) K = 4



(c) K = 3



(d) K = 6

Figure 31: K Clustering for wOBA vs. BABIP

We can see that at $K = 3$ and $K = 4$, the K values were not significant enough to consider grouping these datasets. However, at $K > 5$, this is where we start to see teams group together. We are careful not to go over $K = 6$ because having large K can make sets seem unique to group, but rather muddies the relationships as groupings become more individualized.

When we compared a select number of these datapoints to their similarity scores, we found that they seemed to group together (within a respectable range) in the offensive similarity. Applying the same study to some pitching sabremetrics showed similar results.

This suggests that our similarity algorithm was decent at finding similar teams, although the range was large enough to warrant a future calibrating of coefficients in our similarity calculator.

And comparing the $k = 5$ clustering results using Paraview

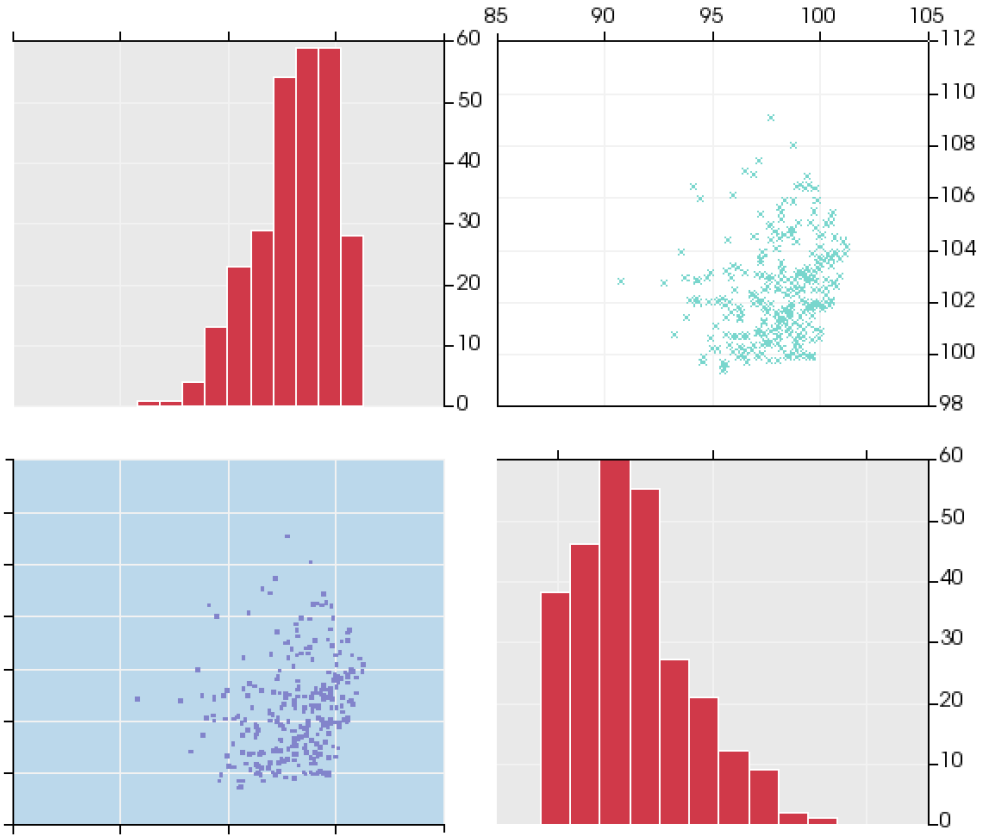
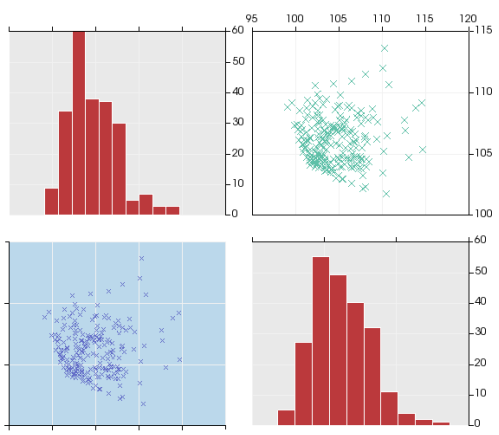
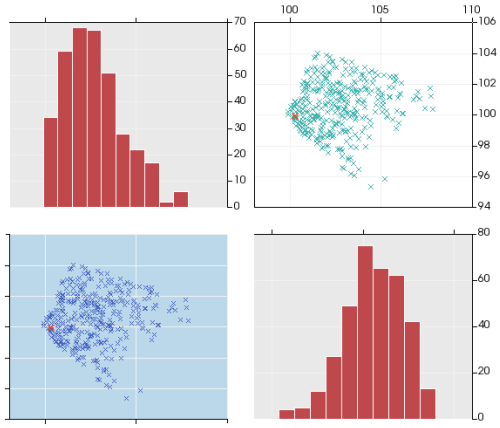


Figure 32: Cluster 1, yellow

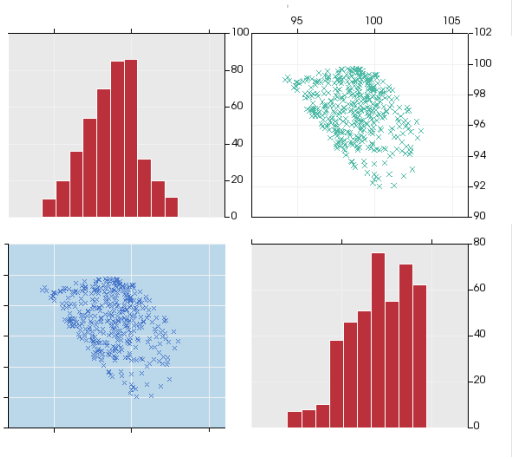
We can see that there's a definite cluster in both our histogram databin and our scatter plot. The histogram in the top left corner represents the x-axis, while the histogram on the bottom right represent the y-axis. The scatterplot on the top right represents a density scatterplot, similar to that of the scatterplot describing our K-means clustering, while the scatterplot on the bottom left represents a weighted density plot with the increments being distributed to account for density of regions. The bins in the histogram suggest a rather tight distribution. It is not quite exactly normal due to it's relative location to the other clusters (it's skewed), but is close to being normally distributed because of the nature of the dataset (team performance).



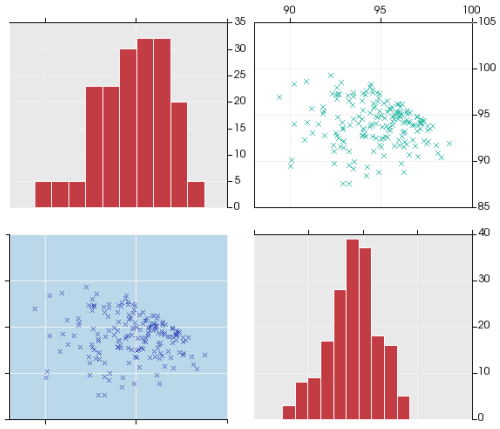
(a) cluster 2, red



(b) cluster 3, blue



(c) cluster 4, green



(d) cluster 5, cyan

Figure 33: Analyzing each individual k-cluster

We can see that like the first cluster, each one of these clusters, when isolated, seemed to be appropriately distributed or clumped, based on the nature of the histograms and the density of the scatterplots. The information or data becomes much clearer when isolated from the other k_i clusters. Most of the k_i clusters look appropriate.

8 Future work

From this finding, we can now move on to further analysis. Our current algorithm for similarity looks good, but the coefficients may need to be rerun on known sets to tweak to obtain a certain accuracy. We may also want to include ballpark effects – every ballpark

has a factor that affects batting and pitching performances. Once this is done, we can then modify these files to work on individual player data. This can first be written in python, but eventually may need to be moved into R for more visualization. We predict that player similarities should be more obvious as even basic offensive categories creates groups of people (ie. sluggers, fast runners, 5 tool, etc), and the k-mean clustering algorithm would produce more obtuse results. Different clusters of players should have have similar stats (remember, we use normalized remember, not RAW). *The ballpark effect will have a more prominent effect and use for individual players.*

Once the similarity can be run on players, we can then then run test sets found from queries made on the database, and apply machine learning until we get the correct coefficients. Then, we apply this to a non-test set to yield results for any players throughout any duration of time. These similar player stats will be combined for the seasons after the similar years, and we will then use this statistic to project production and performance down the road for the target player. *Note we cannot apply this extrapolation to teams, because teams vary to a degree that extrapolating provides no additional important information.*

Note, similarity scores for teams could also be used down the road. For the projections of wins or successes in the new season, we can consider taking the new roster and statistics from the previous season and run the program the same way.

Github R snippets

```r
# track the number of pitches per game (for each pitcher-type combination)
n_pitches <- pitches %>%
  group_by(pitch_type, pitcher_name, date) %>%
  summarise(count = n()) %>%
  mutate(group = paste(pitcher_name, pitch_type, sep = ": ")) %>%
  mutate(dated = as.Date(date, format = "%Y_%m_%d")) %>%
  data.frame %>% mutate(max_n = max(count))

# time series plot with clickSelects
series <- ggplot() +
  geom_line(aes(x = dated, y = count, colour = pitch_type,
                linetype = pitcher_name, group = group), data = n_pitches)
   +
  stat_identity(aes(x = dated, y = max_n, clickSelects = date, alpha = 0.2)
   ,
                geom = "bar", data = n_pitches) + scale_alpha(guide = 'none
   ') +
  xlab("") + ylab("Number of pitches") + theme_animint(width = 800, height
   = 200)

plist2 <- list(strike = strike,
               strikeDate = strike_date,
               series = series,
               selector.types = list(date = "multiple"))

structure(plist2, class = "animint")
```

SrC Snippets 1: Github R snippets

Rstudio snippets

```r
library(splines)
library(arm)
library(pitchRx)
library(dplyr)
library(reshape2)

#Download data.
start = "2014-07-01"
end = "2014-10-01"

db <- src_sqlite("pitchfx.sqlite3", create=TRUE)
```

```r
#Uncomment to download data only needs to happen once.
pitches <- scrape(start=start, end=end, connect=db$con)

#Select only the setup and strikeout pitches.
abs <- tbl(db, sql("SELECT * FROM atbat WHERE  date > '2014-03-01'"))
pitches_ <- tbl(db, sql("SELECT * FROM pitch"))

#Join on atbat info.
abs <- inner_join(abs, pitches_, by=c("gameday_link", "num"))

#Need to join pitches, with setup pitches.
z <- collect(abs) #Need to use rank, not implemented in sqlite.

y <- rbind_list(mutate(z, lbl='last'),
                mutate(z, lbl='setup')) %>%
     arrange(lbl, gameday_link, num, id) %>%
     group_by(lbl, gameday_link, num) %>%
     mutate(rk = rank(id),
            rrank=rank(id) - max(rk),
            rrank = ifelse(lbl=='setup', rrank+1, rrank)
            ) %>%
    filter(lbl %in% c('setup', 'last') & rk > 1 & rrank < 1) %>%
    select(gameday_link, num, rrank, pitch_type, lbl, start_speed,
           count, event, des, px, pz, stand, p_throws, pitcher, count) %>%
    melt(id.vars=c('gameday_link', 'num', 'rrank', 'lbl')) %>%
    dcast(gameday_link + num + rrank  ~ variable + lbl) %>%
    filter(!is.na(event_setup))

#Independent of other variables, what pitches "set up" others best for
    swings and misses?
strikeouts <- y%>%
    mutate(swinging_k = des_last %in% c('Swinging Strike', 'Swinging Strike
    (Blocked)'),
           called_k = des_last=='Called Strike',
           speed_change = as.numeric(start_speed_last) - as.numeric(start_
    speed_setup),
           x_change = ifelse(stand_last=='L', -1, 1) * (as.numeric(px_last)
    - as.numeric(px_setup)),
           y_change = as.numeric(pz_last) - as.numeric(pz_setup))
```

SrC Snippets 2: Rstudio R snippets

Similarity of Teams

```
import csv
import pandas as pd
from math import isnan
import os
from sys import exit
import argparse
import numpy as np

dataFile = "/Users/kaichang/Documents/classes/ay119/final_project/lahman-csv/Teams.csv"
years, numSeasons, teamNames, league, gamesPlayed, teamWins, R, AB, H, doubles, triples, HR, BB, SO, SB, CS, HBP, SF, ERA, IPouts, BBA, SOA, HRA, HA, CG, SHO, SV, FP, E, DP, BPF, PPF, ER, avg, sums = readDatabase(dataFile)
header = ["comparedTeam", "League", "Wins", "SimiliartyHitting", "SimilarityPitching","BABIP","wOBA","ERA","FIP","PBABIP","effERA"]
dir = "results/"
if not os.path.exists(dir): # create results directory if it doesn't exist already
    os.makedirs(dir)
createOutputFiles(years, teamNames, numSeasons)

# compare teams, calculate scores, and write the scores to a file
# starts at 2012
for j in range (2686, numSeasons):
        year1, team1, lg1, projwins1, BABIPPlus1, wOBAPlus1, OBPPlus1, SLGPlus1, KPPlus1, BBPPlus1, RPPlus1, HRPPlus1, SBCSRatePlus1, ERAPlus1, WHIPPlus1, HP9Plus1, BBP9Plus1, KP9Plus1, FIPPlus1, PBABIPPlus1, effERAPlus1, CGratePlus1 = getTeamInfo(years, teamNames, league, gamesPlayed, teamWins, R, AB, H, doubles, triples, HR, BB, SO, SB, CS, HBP, SF, ERA, IPouts, BBA, SOA, HRA, HA, CG, SHO, SV, FP, E, DP, BPF, PPF, ER, avg, j)
        id1 = str(year1) + ' ' + team1
        if args.verbose:
            print "Comparison team: %s" %id1
        fileToOpen = os.path.join(dir, id1) + '.csv'
        try:
            # Open Team J's results file for writing
            f = open(fileToOpen, 'a')
            results = csv.writer(f)
            row = [id1, lg1, projwins1, 100, 100, BABIPPlus1, wOBAPlus1, ERAPlus1, FIPPlus1, PBABIPPlus1, effERAPlus1]
```

```
            results.writerow(row)
            #runs and checks similarities dating back to 1960
            for k in range (1344, numSeasons):
                year2, team2, lg2, projwins2, BABIPPlus2, wOBAPlus2,
    OBPPlus2, SLGPlus2, KPPlus2, BBPPlus2, RPPlus2, HRPPlus2, SBCSRatePlus2,
     ERAPlus2, WHIPPlus2, HP9Plus2, BBP9Plus2, KP9Plus2, FIPPlus2,
    PBABIPPlus2, effERAPlus2, CGratePlus2 = getTeamInfo(years, teamNames,
    league, gamesPlayed, teamWins, R, AB, H, doubles, triples, HR, BB, SO,
    SB, CS, HBP, SF, ERA, IPouts, BBA, SOA, HRA, HA, CG, SHO, SV, FP, E, DP,
     BPF, PPF, ER, avg, k)
                id2 = str(year2) + ' ' + team2
                if (id1 != id2): # prevent comparing a team to itself
                    row = [] # start a blank row for a new comparison
                    row.append(id2) #add the comparison team as the first
    column
                    row.append(lg2)
                    row.append(projwins2)
                    simScoreHit = compareTeamsHitting(wOBAPlus1, wOBAPlus2,
     BABIPPlus1, BABIPPlus2, OBPPlus1, OBPPlus2, SLGPlus1, SLGPlus2, RPPlus1
    , RPPlus2, KPPlus1, KPPlus2, BBPPlus1, BBPPlus2, HRPPlus1, HRPPlus2,
    SBCSRatePlus1, SBCSRatePlus2)
                    row.append(simScoreHit)
                    simScorePitch = compareTeamsPitching(ERAPlus1, ERAPlus2
    , FIPPlus1, FIPPlus2, WHIPPlus1, WHIPPlus2, KP9Plus1, KP9Plus2,
    BBP9Plus1, BBP9Plus2, HP9Plus1, HP9Plus2, PBABIPPlus1, PBABIPPlus2,
    effERAPlus1, effERAPlus2)
                    row.extend((simScorePitch,BABIPPlus2,wOBAPlus2,ERAPlus2
    ,FIPPlus2,PBABIPPlus2,effERAPlus2))
                    results.writerow(row)
        except Exception as e:
            print "Error opening %s: %s" % (fileToOpen, e)
        f.close() #we are done with team J's CSV file.
```

SrC Snippets 3: Similarity Algorithm

3D Visualization

```
import pandas as pd  # (*) pandas for dataframe manipulation
import csv
import matplotlib.pyplot as plt # module for plotting
import sklearn
import scipy
import numpy as np
```

```
7
8  from mpl_toolkits.mplot3d import Axes3D
9
10
11
12
13 dataFile = "/Users/kaichang/Documents/classes/ay119/final_project/baseball-
       team-similarity-master/results/2015 Philadelphia Phillies.csv"
14 df = pd.read_csv(dataFile, sep=',')
15
16 #df = df.drop(df.index[0])  # drop first row (US totals)
17 #df = df[df['murder'] < 11] # drop out-of-range rows
18
19 fig = plt.figure()          # (!) set new mpl figure object
20 ax = fig.add_subplot(111, projection='3d')  # add axis
21
22 ax.set_xlabel('BABIP of offense')
23 ax.set_ylabel('wOBA of offense')
24 ax.set_zlabel('BABIP of pitchers')
25 plt.title('Similarity comparisons amongst teams')
26
27 scatter = ax.scatter(
28     df['BABIP'],
29     df['wOBA'],
30     df['PBABIP'],
31     # linewidths=2,
32     # edgecolor='w',
33     # alpha=0.6
34 )
35
36 #plt.show()
37 plt.savefig('Sim1.png')
```

SrC Snippets 4: 3D Visualization

4D Visualization

```
1 import matplotlib
2 import matplotlib.pyplot as plt
3 import pandas as pd  # (*) pandas for dataframe manipulation
4 import csv
5 import matplotlib.pyplot as plt # module for plotting
6 import sklearn
```

```
import scipy
import numpy as np
from matplotlib import cm
import matplotlib.colors as colors
from mpl_toolkits.mplot3d import Axes3D
import pylab
from scipy.interpolate import griddata


dataFile = "/Users/kaichang/Documents/classes/ay119/final_project/baseball-team-similarity-master/results/2015 Philadelphia Phillies.csv"
df = pd.read_csv(dataFile, sep=',')

#df = df.drop(df.index[0])  # drop first row (US totals)
#df = df[df['murder'] < 11] # drop out-of-range rows

fig = matplotlib.pyplot.gcf()
ax1 = fig.add_subplot(111, projection='3d')  # add axis
#ax1.set_zlabel('BABIP of offense')
#ax1.set_ylabel('wOBA of offense')
#ax1.set_xlabel('BABIP of pitchers')
ax1.set_xlabel('Pitching Similarity Scores')
ax1.set_ylabel('Hitting Similarity Scores')
ax1.set_zlabel('BABIP')
#plt.title('Similarity comparisons amongst teams')

Z = df['BABIP']
#Y = df['wOBA']
#X = df['PBABIP']
X = df['SimilarityPitching']
Y = df['SimiliartyHitting']
#Z = df['Wins']
C = df['Wins']

xi = np.linspace(X.min(),X.max(),100)
yi = np.linspace(Y.min(),Y.max(),100)
#zi = np.linspace(Z.min(),Z.max(),100)
#ci = np.linspace(C.min(),C.max(),100)
ci = griddata((X, Y), C, (xi[None,:], yi[:,None]), method='cubic')
zi = griddata((X, Y), Z, (xi[None,:], yi[:,None]), method='cubic')


```

```
norm = colors.Normalize(vmin=C.min(),vmax=C.max())
xig, yig = np.meshgrid(xi, yi)
surf = ax1.plot_surface(xig, yig, zi,facecolors=cm.rainbow(norm(ci)), alpha
    =0.7)

m = cm.ScalarMappable(cmap=cm.rainbow)
m.set_array(np.unique(ci))
col = plt.colorbar(m)
plt.show()

#ax1.scatter(X,Y,Z,c=C)
#plt.show()
```

SrC Snippets 5: 4D Visualization (3d color mapping)

K-means Clustering

```
import pandas as pd  # (*) pandas for dataframe manipulation
import csv
import matplotlib.pyplot as plt # module for plotting
import sklearn
import scipy
import numpy as np
import random

from mpl_toolkits.mplot3d import Axes3D

def cluster_points(X, mu):
    clusters  = {}
    for x in X:
        bestmukey = min([(i[0], np.linalg.norm(x-mu[i[0]])) \
                    for i in enumerate(mu)], key=lambda t:t[1])[0]
        try:
            clusters[bestmukey].append(x)
        except KeyError:
            clusters[bestmukey] = [x]
    return clusters

def reevaluate_centers(mu, clusters):
    newmu = []
    keys = sorted(clusters.keys())
    for k in keys:
        newmu.append(np.mean(clusters[k], axis = 0))
```

```
    return newmu

def has_converged(mu, oldmu):
    return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu]))

def find_centers(X, K):
    # Initialize to K random centers
    oldmu = random.sample(X, K)
    mu = random.sample(X, K)
    while not has_converged(mu, oldmu):
        oldmu = mu
        # Assign all points in X to clusters
        clusters = cluster_points(X, mu)
        # Reevaluate centers
        mu = reevaluate_centers(oldmu, clusters)
    return(mu, clusters)
```

SrC Snippets 6: K-means Clustering

9 Full Graphics Vectors

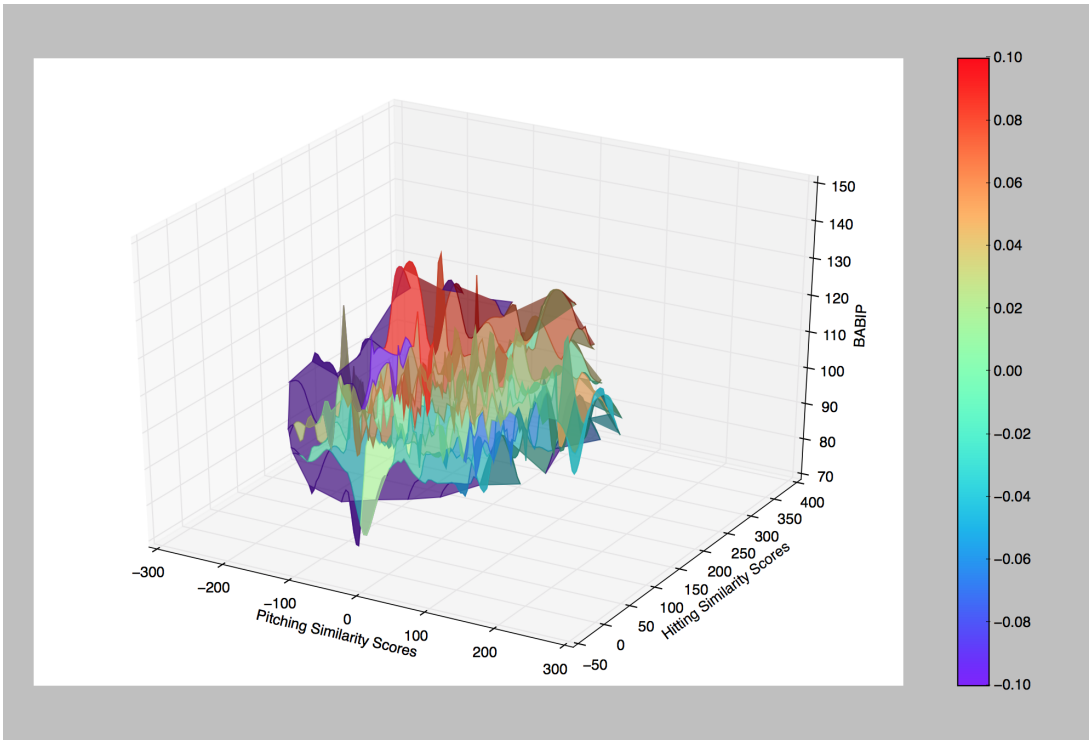


Figure 34: 4D Representation

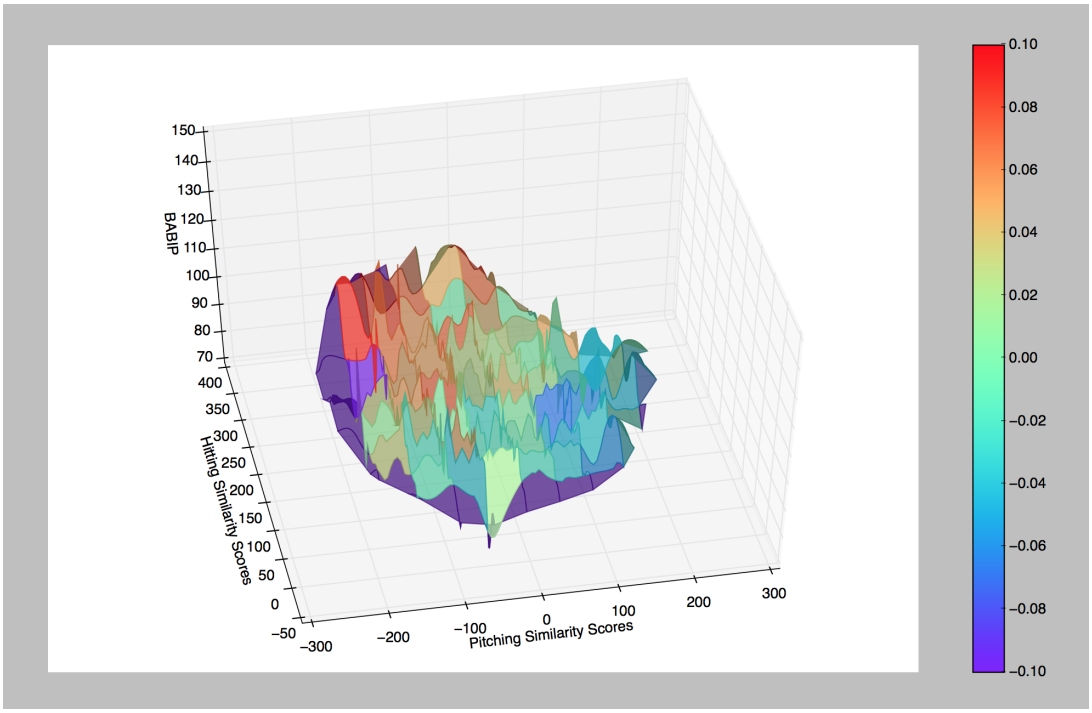


Figure 35: 4D Representation cont.

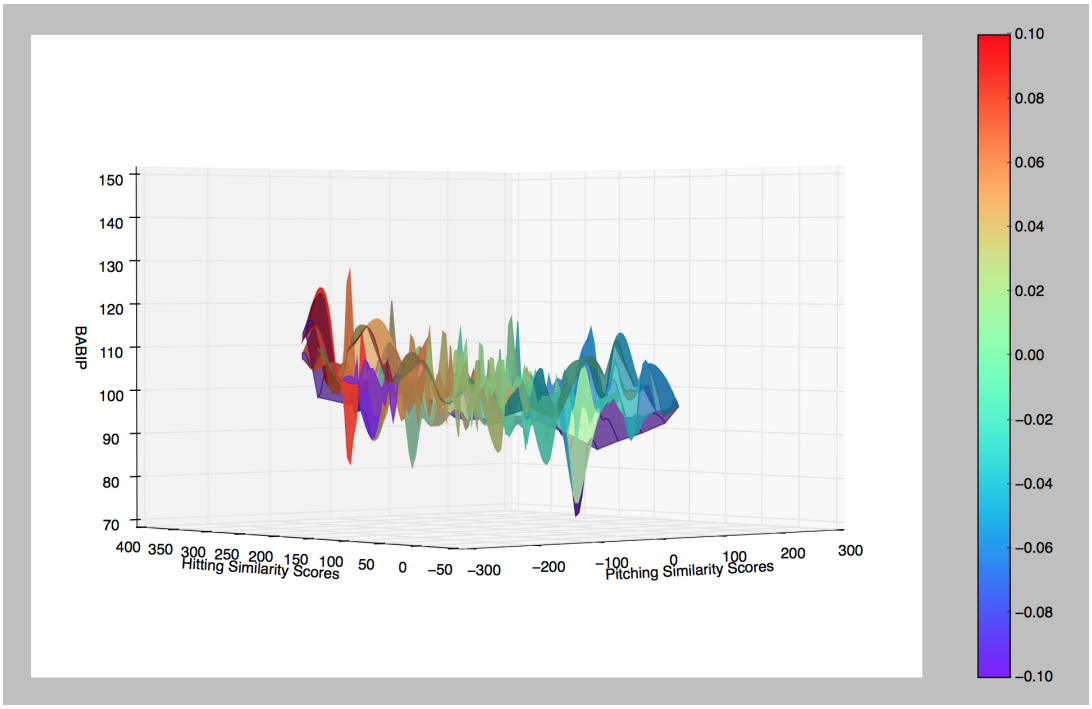


Figure 36: 4D Representation cont.

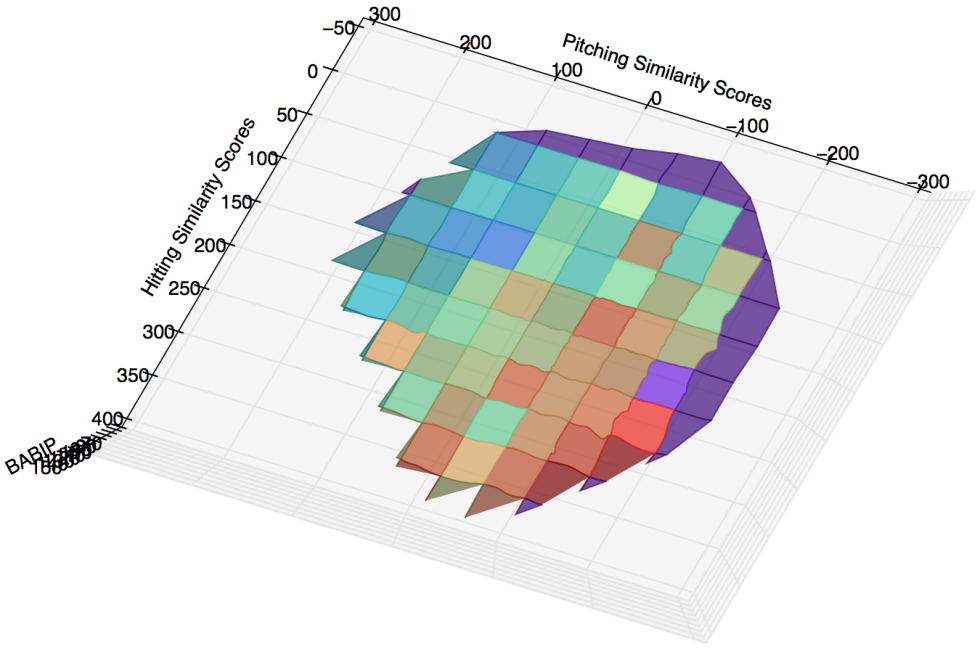


Figure 37: 4D Representation cont.

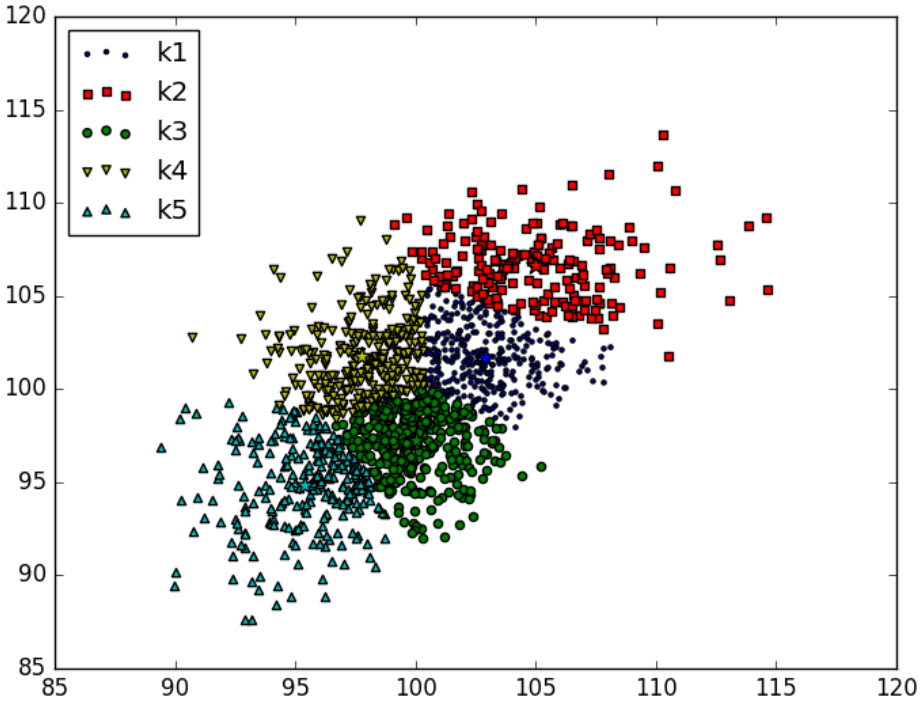


Figure 38: K=5 Clustering

References

[1] Gartheeban, Ganeshapillai, and John Guttag. "A Data-driven Method for In-game Decision Making in MLB." Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '13 (2013): n. pag. Web.

[2] "Saberizing a Mac #6: Calculating FIP, FIP-, ERA-." Beyond the Box Score. N.p., 24 Jan. 2013. Web. May 2016.

[3] "FIP — FanGraphs Sabermetrics Library." FIP — FanGraphs Sabermetrics Library. N.p., n.d. Web. May 2016.

[4] "Clustering With K-Means in Python." The Data Science Lab. N.p., 12 Dec. 2013. Web. May 2016.

[5] "K-means Clustering." K-means Clustering Scikit-learn 0.17.1 Documentation. N.p., n.d. Web. May 2016.

[6] "WOBA — FanGraphs Sabermetrics Library." WOBA — FanGraphs Sabermetrics Library. N.p., n.d. Web. Apr. 2016.

[7] "Park Adjustments — Baseball-Reference.com." Baseball-Reference.com. N.p., n.d. Web. 2016.

[8] Bales, John. "Most Important Stats: Batting." Most Important Stats: Batting. N.p., n.d. Web. May 2016.

[9] "BABIP — FanGraphs Sabermetrics Library." BABIP — FanGraphs Sabermetrics Library. N.p., n.d. Web. May 2016.

[10] "Principal Component Analysis Explained Visually." Explained Visually. N.p., n.d. Web. May 2016.

[11] "Matplotlib Correct Colors/colorbar for Plot with Multiple Surfaces Each of a Different Color." Python. N.p., n.d. Web. June 2016.