Introduction:

Recently, the sports world has been fixated with the idea of “Moneyball Revolutions”. The idea is based upon the book-turned-movie which looks into the Oakland A’s baseball team whose general manager (Billy Beane) decided to sign players who had less value according to traditional player-value measures. At the time Beane opted for this strategy, most baseball teams were signing players who could knock the ball out of the park more than their peers could. Beane, however, looked for players who could simply get on base more often than their peers. His methodology was based upon data analytics that ultimately took his team from a collection of questionable players on a tight budget to one that could win. This prospect is incredibly interesting, if elusive, to most professional sports teams around the world. Who would not want to secure the most effective players, and not only that, but at the best cost? Value players are on the rise in the sports world. General managers and coaches are always trying to bring cheap wins to team owners because, at its root, sports teams are businesses.

This parable may be attractive to those involved in front office positions for sports teams, however, the situation is not always analogous to Billy Beane’s. Every sport is not baseball and not every sport has an equivalent of base hits. But, as data collection capability grows, so does the quest for big data analysts who can interpret raw data into something meaningful that will give one team a competitive edge over all of the others. With this paper we attempt to provide general managers, team owners, coaches, and all those involved in player acquisitions an effective model that could be applied in the real world to make value decisions on players in (almost) any sport.

Benchmarks:

The basis for our calculations is something known as the Elo Rating System which was initially developed to create rankings for individual players in competitive chess circles. One difficulty in ranking teams in sports, or individuals in two-player games, is that the quality of a team is a rather subjective thing. We might suggest that the quality of a player or team comes from a combination of their statistics or, perhaps, they pass the “eye-test” of goodness. Both of these things are ultimately subjective and attempt to make a science out of sports scouting when boiled down to picking individual members of a team. Even when choosing players or teams based upon who can win, this is relative and, thus, subjective. But, ultimately, what matters in the world of sports is who wins. The Elo Rating System was developed with the relative nature of sports in mind with regard to chess players. On a simplistic level, the Elo Rating System designates an exchange of assigned points between two players and these points get passed around over time. When the system is initialized in a league, a set number of points is given to each player in the league. When one player loses to another, they lose a number of points to the other player commensurate with the difference in points each player possessed before the start of the match. For example, if player one was considered vastly superior to player two at the start of a match between the two players and player one lost, player one would lose a large number of their points to player two. Conversely, if player one were to win the match, the rating system would respond by awarding player one a small number of points out of player two’s pocket.

It is clear to see the value of such a system; it takes the relativity of the sporting world into account and focuses on wins and losses primarily. However, the system becomes more complex when applied to a team composed of many players. We might easily transpose the idea of the Elo Rating System to the world of basketball when asking the question, “Which team is the best?” However, general managers do not want the answer to this question; they want to answer the question, “Who is contributing the most to the team?” and, furthermore, “What is a player’s specific contribution given their cost?” The answer to the second question can be interpreted as a statement of economic value for a player which, as mentioned before, holds great value for sports team owners. This presents some problems for the Elo Rating System. Although the benchmark for this problem is the Elo Rating System developed for chess competitions, it is appropriate to speak of the unique considerations this team made in developing a solution to this problem.

Methodology:

Considerations:

The first step in answering the aforementioned question was to assign a score (or Elo Value) to each team. We made an assumption that a team is simply the aggregate of the players of which it is composed. This assumption ignores the intangible interactions between players which allow them to build off of each other’s success. We believe this assumption, as with the others we made, was necessary because it allowed us to simplify the calculation and focus on our core idea of extending the Elo Ranking System into the field of player-value metrics. However, we believe that by clearly stating these assumptions, improvements can and will be made to our methodology which will, in turn, generate more accuracy in estimating the value of a player.

We assumed that a team’s Elo value was merely the sum of its component player’s Elo value because, in order to compare individual players to one another, it was necessary to compare teams to one another then share the result of either losing or winning among each team member. This, in turn, presents another complication. When the Elo Rating System is applied to the game of chess, we can safely assume that each player bears the entire responsibility for either their loss or their victory; however, when applying the system to a team sport it is necessary to decide if players should share equally in a victory and, if not, then which player bears the largest responsibility for a team’s successes and failures.

One feature of NBA basketball is the number of players that sit on a bench in any given game. Some players never enter the game or, if they do, still get to ride on the coattails of the star players on the team. With this in mind, it seemed unnatural to award such players an equal share in the victory, or blame them equally for losing a game to which they only superficially contributed. Conversely, it seemed like a poor idea to punish a player who played an excellent game. Sometimes, we see consistently good players (like the most valuable player in the NBA) play on teams that fare poorly in the postseason or regular season. Thus, we introduced a bit of subjectivity into the calculation. We decided on different ways to split the share of a loss or a win given an individual player’s expected contribution to a team and their actual contributions.

First, in a loss, we assumed that each player was responsible, in some way, for the loss. We decided to limit the number of assumptions and the subjectivity of our calculations as much as we could in order to produce the best possible results. As stated before, we wanted to distribute blame for a loss as fairly as possible between each player on the team that lost. We arrived at the idea of an expected contribution from each player on the team. We computed this by deciding that we can reasonably expect a player to contribute some percentage to their team's overall performance. This means that if a certain player possesses 50% of their team's aggregated Elo value, then, if the team was to lose, that player would take 50% of the loss distribution. One of the downsides to this strategy for sharing a loss among players on a team is that even if the best player performs well they can still be penalized the most for a loss. However, the upside to this is that it means the share of a loss is apportioned out to each of the players according to what is expected of them. This is a significant improvement over deciding that each player bears equal responsibility for the performance of a team. In addition, this does not penalize players who sit on the bench during most games as much as it penalizes players who generally have a larger impact on each game played.

Then, in a win, we opted for an Elo Value sharing strategy which was dissimilar from the strategy for a loss. Although uniformity between our strategy for a loss and a win was instinctive, initially, we chose to focus more on an individual player's statistics in each game relative to that of his teammates. What this boiled down to was using the table below to give each player a set number of statistics-points. We then totalled the statistics-points of each player on the winning team and gave the players on the winning team a share of the Elo Value gleaned from the losing team that was commensurate with their relative performance. E.g. if player A on the winning team had 50% of the statistics-points of his team he would receive 50% of the Elo points his team received from the losing team. The reason this methodology was more effective in distributing Elo points a team gained from a win than distributing Elo points according to a player's expected value is that the players' actual performance became quantifiable. One problem that arose when adopting the strategy used for a loss was the inability for the system to detect rising talent. E.g. when a young player was drafted onto an established team he would be assigned his initial Elo value and would compose a small percentage of the team's total Elo Value. In effect, it was difficult to see his actual contribution to the team over time because his Elo value would remain on par with his teammates. The only way we would be able to see a change in his Elo value that tracked with his performance would be if he was traded frequently, so that there would be more teams with which he could be compared. To think of it with a real world example: intuitively, we know that Kevin Durant is a good player. We would see him start with a small share of his team's Elo value when he was first drafted, then watch his Elo value increase. It would only be when the Seattle Supersonics or Oklahoma City Thunder traded away their best players that we would see his Elo value exponentiate. Corollarily, a great player whose skills are fading, but is supported by younger, better players, would not see his Elo value diminish because, technically, he is still winning.

Dataset:
In order to calculate these Elo values correctly, we first needed to gather the required data. We wrote scripts to pull the data from http://www.basketball-reference.com. In all, we pulled statistics from every game played in the NBA since 1947 as well as every player that played in each game. This resulted in two main files that we used for computation: a players file and a games file. The players file has each player's performance as it relates to each game in which they were a contributor. The games file has the overall result of the games as well as the combined performance of each collective team. One difficulty of having this data in two separate files was finding a way to accurately combine them to decide: what team a player was on, what share of the team's statistics a player had and whether the player's team won or lost. Since we pulled this data from a website, we decided to use part of the URL to track the game ID, team ID and player ID because the URL for these items must be unique in some way for the website to function properly.

Pre-Computation:
Due to the fact that our main data was in two different files, it was a simple realization that our first step needed to be to combine these two files to get an aggregate file containing all necessary data. The ID mappings we decided on when gathering the data helped us immensely in this step as we simply had to map player statistics to the correct game and combine everything on a single line of the output file. Once this file with all of the data was produced, the main phase of our processing (determining an accurate K-Factor for our analysis) could operate on this. The biggest challenge of calculating the Elo Value for NBA games, is that it must be done in chronological order. For example, we cannot calculate the outcome of a game on day two between team A and team C if we have not yet calculated the outcome of a game on day one between team A and team B since team A's Elo Value will be modified after the game on day one. Not only did we have to make sure to calculate a particular team's Elo in chronological order, but we also had to consider the possibility of a player switching teams. This tight dependency between all the games and all of the players meant we needed to perform the Elo calculation in a single mapreduce step. Further, due to the chronological requirement, we also had to ensure that the data would arrive at the reducer in chronological order to produce accurate results.

Even though our starting dataset was not large and a single Elo calculation needed to be performed in a single mapreduce step, using big data technology allowed for us to parallelize computations of Elo values with different K-Factors. Initially, we attempted a broad sweep of calculating the accuracy of K-Factors from one to one-hundred. This quickly failed in our Hadoop setup due to insufficient disk space as it generated approximately 10GB of data. Because we couldn't do a broad sweep, but still wanted to analyze all of these possible K-Factors, we, instead, decided to do a binary search type of analysis. Our first analysis compared all K-Factors from four to one-hundred with a step size of four, then we narrowed our search to K-Factors between two and fifty-six with a step size of two and our final determination was done with K-Factors between eight and thirty-six with a step size of one.

After we successfully computed the Elo values for all players and teams with different K-Factors, we decided we needed a couple more mapreduce steps to automatically analyze the results for us. We tested many different metrics for determining the best K-Factor, all of which operated on the output Elo value file so they were able to run in parallel. Ultimately, we decided to use a combination of two metrics: percent of games accurately predicted and root mean square error. The percent of games accurately predicted is a simple metric, if and only if the team with the higher Elo value to start the game won, it was predicted accurately. The root mean square error was a little more complicated as we had to determine what expected value to use as an error metric. After some thought, we determined that if a team wins their expected value should be 100% chance to win and the losing team's expected value should be 0% chance to win. Thus, the overall error for each game is the sum of the winning team and losing team's deviation from their expected value.

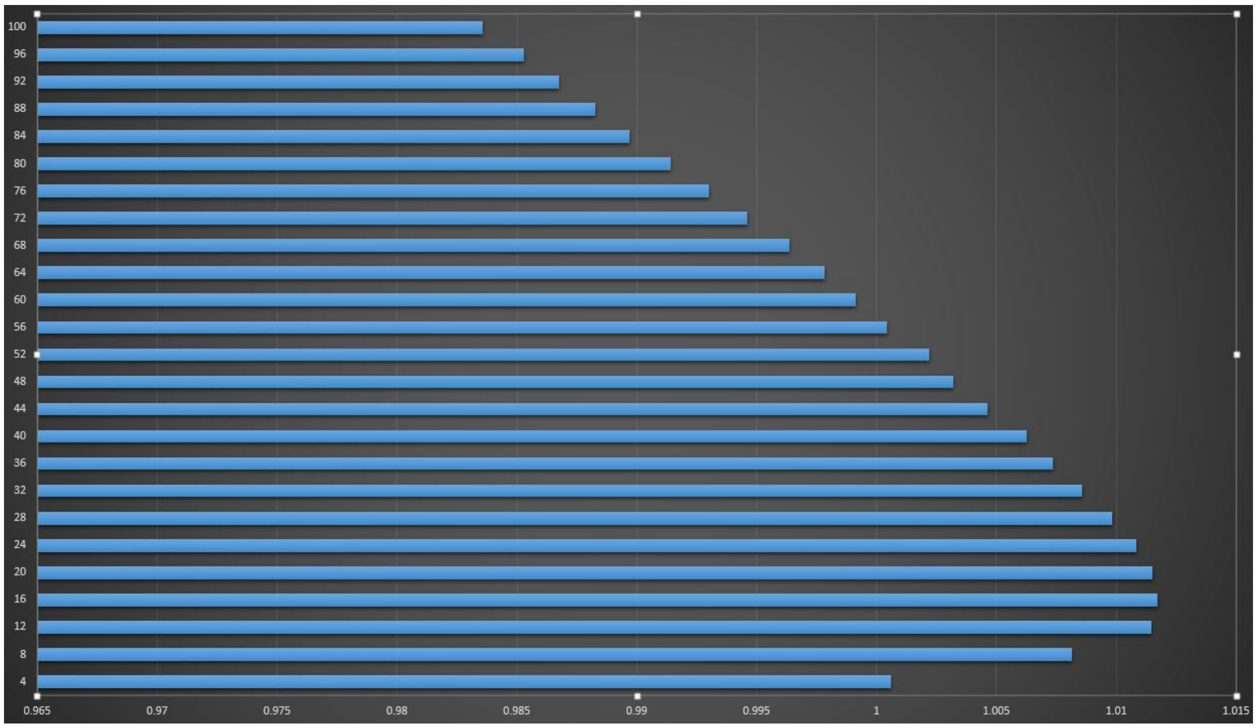
Ultimately, it was using these metrics that allowed us to determine which K-Factor was the best to use and narrow down our search for the best K-Factor on each iteration. What we determined from the data and the graphs below was a K-Factor of 27 produced the most accurate predictions, a K-Factor of 9 had the least amount of error and a K-Factor of 17 produced the best combination of accuracy and error. Examining these numbers we also realized, in our last iteration of testing K-Factors from 8 to 36, every value produced very similar results. While a K-Factor of 27 produced the most accurate results at 64.35% the least accurate K-Factor, 8, produced an accuracy of 63.64% or only a 0.71% accuracy difference. The same conclusion was drawn from the error values, 9 had the least amount of error at 0.949 and a K-Factor of 36 had the most amount of error at 0.96, or a difference of only 0.011.

K-Factors from 4 - 100 with a step size of 4

K-Factor	Error	Avg Error / Error	Accuracy %	Accuracy % / Avg Accuracy %	Average of averages
4	0.953338	1.017256	62.91597	0.983986	1.000621
8	0.949912	1.020925	63.64307	0.995358	1.008142
12	0.950182	1.020635	64.08593	1.002284	1.011459
16	0.9514	1.019328	64.20057	1.004077	1.011703
20	0.953003	1.017613	64.28381	1.005379	1.011496
24	0.954808	1.01569	64.31993	1.005944	1.010817
28	0.956732	1.013647	64.3215	1.005968	1.009808
32	0.958732	1.011533	64.29951	1.005624	1.008579

36	0.96078	1.009376	64.2791	1.005305	1.007341
40	0.96286	1.007196	64.28224	1.005354	1.006275
44	0.964956	1.005008	64.21314	1.004274	1.004641
48	0.967062	1.002819	64.1676	1.003561	1.00319
52	0.969171	1.000637	64.18016	1.003758	1.002198
56	0.971277	0.998467	64.09064	1.002358	1.000413
60	0.973378	0.996312	64.06395	1.00194	0.999126
64	0.975472	0.994174	64.03725	1.001523	0.997848
68	0.977556	0.992055	63.98072	1.000639	0.996347
72	0.979629	0.989955	63.88963	0.999214	0.994584
76	0.981692	0.987875	63.82053	0.998133	0.993004
80	0.983742	0.985816	63.75143	0.997053	0.991434
84	0.98578	0.983778	63.66192	0.995653	0.989715
88	0.987806	0.98176	63.60538	0.994769	0.988264
92	0.98982	0.979763	63.541	0.993762	0.986762
96	0.99182	0.977787	63.47661	0.992755	0.985271
100	0.993809	0.97583	63.38552	0.99133	0.98358

Graph of above data (K-Factors on Y-axis, average of averages on X-axis)

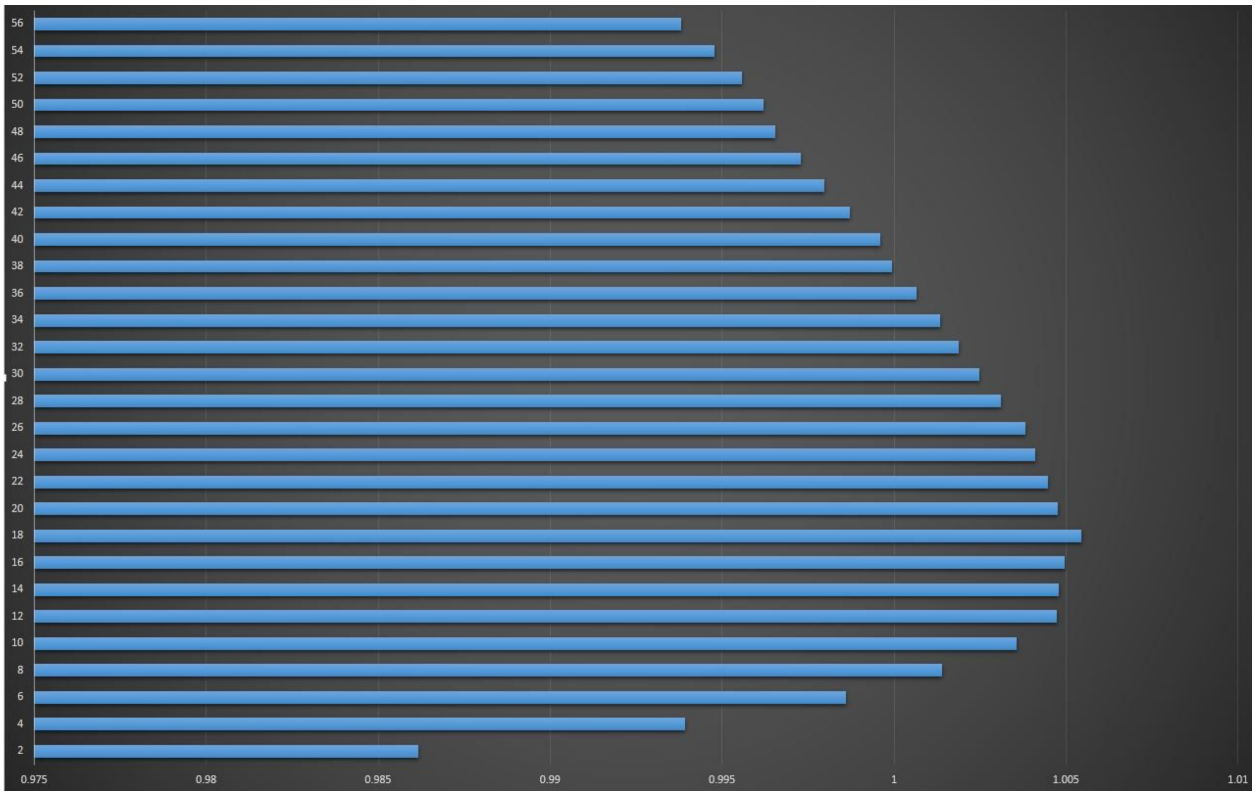


K-Factors from 2 - 56 with step size of 2

K-Factor	Error	Avg Error / Error	Accuracy %	Accuracy % / Avg Accuracy %	Average of averages
2	0.961635	0.997056	62.47782	0.97526	0.986158
4	0.953338	1.005734	62.91597	0.9821	0.993917
6	0.950721	1.008502	63.33998	0.988718	0.99861
8	0.949912	1.009361	63.64307	0.99345	1.001405
10	0.949857	1.009419	63.91476	0.99769	1.003555
12	0.950182	1.009074	64.08593	1.000362	1.004718
14	0.950723	1.0085	64.13305	1.001098	1.004799
16	0.951399	1.007783	64.20057	1.002152	1.004967
18	0.952168	1.006969	64.31522	1.003941	1.005455
20	0.953003	1.006087	64.28381	1.003451	1.004769

22	0.953887	1.005155	64.30422	1.00377	1.004462
24	0.954808	1.004185	64.31993	1.004015	1.0041
26	0.955758	1.003187	64.34977	1.004481	1.003834
28	0.956732	1.002166	64.32307	1.004064	1.003115
30	0.957724	1.001127	64.30893	1.003843	1.002485
32	0.958732	1.000075	64.29951	1.003696	1.001886
34	0.959751	0.999013	64.29794	1.003672	1.001342
36	0.96078	0.997943	64.2791	1.003378	1.00066
38	0.961817	0.996867	64.25711	1.003034	0.999951
40	0.962859	0.995788	64.28224	1.003427	0.999607
42	0.963906	0.994707	64.23669	1.002716	0.998711
44	0.964957	0.993624	64.21314	1.002348	0.997986
46	0.966009	0.992542	64.19429	1.002054	0.997298
48	0.967062	0.99146	64.1676	1.001637	0.996549
50	0.968117	0.990381	64.19429	1.002054	0.996217
52	0.969171	0.989303	64.18016	1.001833	0.995568
54	0.970225	0.988229	64.14718	1.001318	0.994774
56	0.971277	0.987158	64.09064	1.000436	0.993797

Graph of above data (K-Factors on Y-axis, average of averages on X-axis)

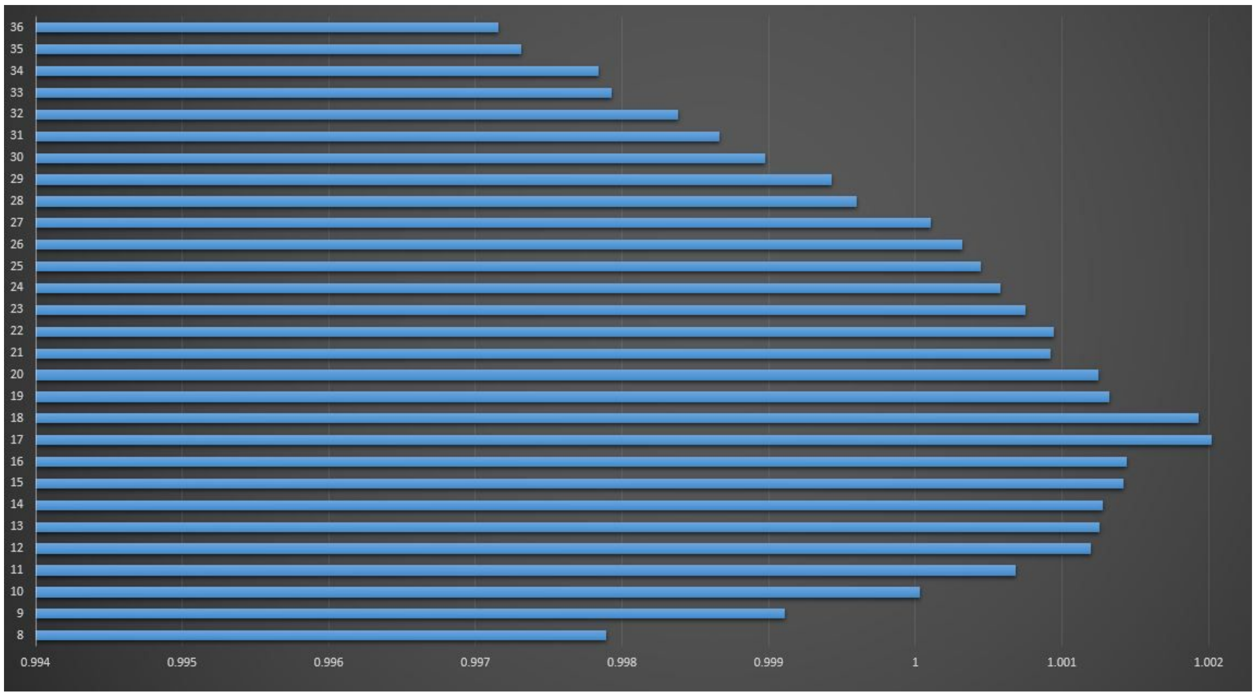


K-Factors from 8 - 36 with a step size of 1

K-Factor	Error	Avg Error / Error	Accuracy %	Accuracy % / Avg Accuracy %	Average of averages
8	0.949912	1.004642	63.64307	0.991135	0.997889
9	0.949821	1.004738	63.79383	0.993483	0.99911
10	0.949857	1.0047	63.91476	0.995366	1.000033
11	0.949985	1.004564	64.00741	0.996809	1.000687
12	0.950182	1.004357	64.08593	0.998032	1.001194
13	0.950431	1.004093	64.11106	0.998423	1.001258
14	0.950723	1.003785	64.13305	0.998765	1.001275
15	0.951047	1.003442	64.17388	0.999401	1.001422

16	0.951399	1.003071	64.20057	0.999817	1.001444
17	0.951774	1.002676	64.29951	1.001358	1.002017
18	0.952168	1.002261	64.31522	1.001602	1.001932
19	0.952579	1.001829	64.26496	1.00082	1.001324
20	0.953003	1.001383	64.28381	1.001113	1.001248
21	0.95344	1.000925	64.27124	1.000918	1.000921
22	0.953887	1.000456	64.30422	1.001431	1.000943
23	0.954343	0.999977	64.3105	1.001529	1.000753
24	0.954808	0.999491	64.31993	1.001676	1.000583
25	0.95528	0.998997	64.33406	1.001896	1.000446
26	0.955758	0.998497	64.34977	1.00214	1.000318
27	0.956242	0.997991	64.35448	1.002214	1.000102
28	0.956732	0.99748	64.32307	1.001725	0.999602
29	0.957226	0.996965	64.33406	1.001896	0.999431
30	0.957724	0.996447	64.30893	1.001505	0.998976
31	0.958227	0.995924	64.30265	1.001407	0.998666
32	0.958732	0.9954	64.29951	1.001358	0.998379
33	0.95924	0.994872	64.27596	1.000991	0.997931
34	0.959751	0.994342	64.29794	1.001333	0.997838
35	0.960265	0.993811	64.26496	1.00082	0.997315
36	0.960781	0.993277	64.2791	1.00104	0.997158

Graph of above data (K-Factors on Y-axis, average of averages on X-axis)



After using the data above to determine the optimal K-Factor is 17, we made a final run using this K-Factor to produce recommendations for General Manager's to use. To produce these recommendations, we go through our same Elo calculation process that we did to determine the best K-Factor. Once the Elo numbers for every player and every game are generated we run two more mapreduce steps to make the final recommendations. The first step simply outputs every player's most recent Elo value as this is the value they will have heading into next year's season. The final mapreduce job takes this file and a salary file and outputs final recommendations. We sort the recommendations based on who has the highest Elo value per salary dollar. This will give the general manager the best "bang-for-his-buck". Using these recommendations a general manager should be able to create a "moneyball-like" team that puts the highest valued players on his team.

One thing that was not taken into account in our recommendations, that a general manager would have to consider, is whether the player is available in the offseason or not. We decided to include players that are under contract because it could still assist a GM in deciding whether a player is worth trading or not.

We believe the main improvement on our recommendation system that could be made is to include a player's position in the recommendations. It is not uncommon for different positions to have completely different values within the NBA. For example, say that the center position in the NBA makes the least amount of money. Using our recommendation system, it could be likely that the highest recommended players are all centers. While this information would still be useful to a GM, he or she wouldn't want to have a team of only centers so they would have to sift through the information in order to optimally design their complete team. If we could also include positions and do a recommendation based on position, this could assist GMs with decisions even more. The main reason we decided not to include this, is because this information is difficult to obtain. Many players in the NBA are listed as having at least two positions. With more time, we believe would could have worked with this, but it was not in the scope of this project.

Another flaw we have seen with our recommendation system, which can be seen in the table below, is it tends to recommend players with a very low salary. That is because there is only about a 2000 Elo point spread between the best and the worst player in the NBA. However, the salary spread between these players is enormous (i.e., $30,000,000 vs $50,000). Even though this is the case, we still believe this information would be useful to GMs as they know what their salary cap is, which means they wouldn't be filling their team with 10 $50,000 players. They would choose a mix of those high-priced players with a high recommendation and low priced players with a high recommendation. This obviously could be a large improvement on our project though if we figured out how to group our recommendations. Unfortunately, this can be the downside of relying on pure numbers and may prove to be a difficult task. We did not foresee this issue so it was not in the scope of our project. Overcoming this downfall is not very difficult though, as GMs can simply sort by the overall Elo value to see the best players, and then decide whether their cost/value ratio is high enough.

Below is the top 20 players based on our recommendation system:

Name	Team	Elo / Salary ($M)	Elo Value	Salary ($M)
P.J. Hairston	CHO	99615.98	1148.971	0.011534
Ray McCallum	SAC	80115.2	957.2965	0.011949
Dahntay Jones	MEM	50065.19	913.9401	0.018255
Elijah Millsap	UTA	49374.43	1139.019	0.023069
Marcus Georges-Hunt	MIA	47393.01	1184.825	0.025
Tony Wroten	MEM	44980.75	1124.519	0.025
Quinn Cook	DAL	37480.07	1198.2	0.031969
Yogi Ferrell	BRK	36857.03	1178.282	0.031969
Troy Williams	MEM	36554.83	1168.621	0.031969
D.J. Stephens	MIL	34005.43	1190.19	0.035

Jordan Crawford	ATL	24774.93	1531.611	0.061821
D.J. Kennedy	CLE	23723.26	1186.163	0.05
Jarrod Uthoff	DAL	23631.54	1181.577	0.05
Axel Toupane	DEN	22872.5	1176.767	0.051449
Cliff Alexander	POR	22763.18	1171.143	0.051449
Reggie Williams	GSW	22184.34	1109.217	0.05
Phil Pressey	BOS	22006.78	770.2373	0.035
Dionte Christmas	PHO	21861.68	1093.084	0.05
Anthony Brown	LAL	20631.2	1061.455	0.051449
Justin Harper	ORL	20056.05	1156.673	0.057672

Conclusion:

Because the performance of a player can only be gleaned from the performance of his team in the NBA, we determined that to best demonstrate the goodness of our model we should look to how well a team's aggregated Elo value would work. Ultimately, our model produced decent results with a K-Factor of 27 determining the results of games with a higher accuracy than any other K-Factor; while a K-Factor of 9 had the least error in predicting the winner of a game. However, the model was only able to choose the winners of games with around a 64% level of accuracy. Ultimately, we would prefer a model with a higher degree of accuracy in choosing the best players; however, we believe the model for choosing good players presents a good start to any further investigations.

Individual Contributions:

Spencer Lofing worked on some of the planning in constructing the MapReduce design and wrote the bulk of the paper. Trevor Von Seggern created the visualizations and converted the raw Elo values to a cost to Elo value ratio. Nate Olson created the MapReduce portion of the project and contributed to the paper.