
Dominance Hierarchy Ranking of a Synergistic Feature Set: Major League Baseball

Xing Yang Lan, Arya Rajpal, Sruthi Rayasam, David Wong, Shileng Zhang.
Statistics Department
University of California, Davis

Abstract

We explore and implement the concepts of ranking algorithms on the 2021 MLB team data and the surprising world series win by the Atlanta Braves. Our rankings, based on our variation of the Google PageRank algorithm, create relatively more accurate predictions than those based off the rankings of professional predictions such as an Elo ranking and a complex black-box machine learning algorithm. These were restricted due to their failure to account for the mid-season addition of prominent players and the inherent limitations of pairwise based ranking systems.

1 Introduction

Major league baseball is notable for statistical analysis on account of the sheer volume of games played, which allows statisticians to play around with a lot of data. One of the primary motivations, beyond those personally or professionally invested in these games, is betting. The 2021 MLB games are notable for the surprise champion, completely upsetting the expected statistical predictions.

1.1 MLB's Season and Bracket System

Baseball in the MLB is structured in a specific manner that follow an elimination-style selection method. There are 30 teams divided into two leagues: American and National. Within each of these two leagues, there are three subsets of teams separated by geography called divisions. The divisions are divided as East, Central, and West, with one version for both leagues, e.g., AL (American League) West and NL West.

The regular season is a series of 162 games that each team plays from April to October in order to determine the 10 qualifiers who will move onto the post-season, with one winner from each division and four "wild card" teams, the two runners-up from each league. It is important to note that fewer games are played between teams of different leagues, "inter-league" games as most are inter-division. Teams play 19 games against each of their division's 4 opponents (76 games inter-division), and then 6 games against 4 different teams and 7 games against each of the other 6 teams from other divisions in their own league (66 games). The remaining 20 games are against every team in a cycling rotation of one division from the other league. This means that every team will play against each other at least every three years.

1.2 MLB's 2021 Season

The 2021 season followed the abnormal 2020 season, due to Covid-19 complications. The inter-league plays in this year correspond to the geographical division rather than the established cycle.

American League [edit]

V·T·E	AL East	W	L	Pct.	GB	Home	Road
	(1) Tampa Bay Rays	100	62	0.617	—	52–29	48–33
	(4) Boston Red Sox	92	70	0.568	8	49–32	43–38
	(5) New York Yankees	92	70	0.568	8	46–35	46–35
	Toronto Blue Jays	91	71	0.562	9	47–33	44–38
	Baltimore Orioles	52	110	0.321	48	27–54	25–56

V·T·E	AL Central	W	L	Pct.	GB	Home	Road
	(3) Chicago White Sox	93	69	0.574	—	53–28	40–41
	Cleveland Indians	80	82	0.494	13	40–41	40–41
	Detroit Tigers	77	85	0.475	16	42–39	35–46
	Kansas City Royals	74	88	0.457	19	39–42	35–46
	Minnesota Twins	73	89	0.451	20	38–43	35–46

V·T·E	AL West	W	L	Pct.	GB	Home	Road
	(2) Houston Astros	95	67	0.586	—	51–30	44–37
	Seattle Mariners	90	72	0.556	5	46–35	44–37
	Oakland Athletics	86	76	0.531	9	43–38	43–38
	Los Angeles Angels	77	85	0.475	18	40–42	37–43
	Texas Rangers	60	102	0.370	35	36–45	24–57

National League [edit]

V·T·E	NL East	W	L	Pct.	GB	Home	Road
	(3) Atlanta Braves	88	73	0.547	—	42–38	46–35
	Philadelphia Phillies	82	80	0.506	6½	47–34	35–46
	New York Mets	77	85	0.475	11½	47–34	30–51
	Miami Marlins	67	95	0.414	21½	42–39	25–56
	Washington Nationals	65	97	0.401	23½	35–46	30–51

V·T·E	NL Central	W	L	Pct.	GB	Home	Road
	(2) Milwaukee Brewers	95	67	0.586	—	45–36	50–31
	(5) St. Louis Cardinals	90	72	0.556	5	45–36	45–36
	Cincinnati Reds	83	79	0.512	12	44–37	39–42
	Chicago Cubs	71	91	0.438	24	39–42	32–49
	Pittsburgh Pirates	61	101	0.377	34	37–44	24–57

V·T·E	NL West	W	L	Pct.	GB	Home	Road
	(1) San Francisco Giants	107	55	0.660	—	54–27	53–28
	(4) Los Angeles Dodgers	106	56	0.654	1	58–23	48–33
	San Diego Padres	79	83	0.488	28	45–36	34–47
	Colorado Rockies	74	87	0.460	32½	48–33	26–54
	Arizona Diamondbacks	52	110	0.321	55	32–49	20–61

Figure 1: MLB Leagues (Wikipedia) [8]

1.3 MLB 2021 Post-Season Bracket

The post-season games involve the top teams in best of 3's, best of 5's, and best of 7's games. For 2021, there were 35 postseason games, a relatively low number.

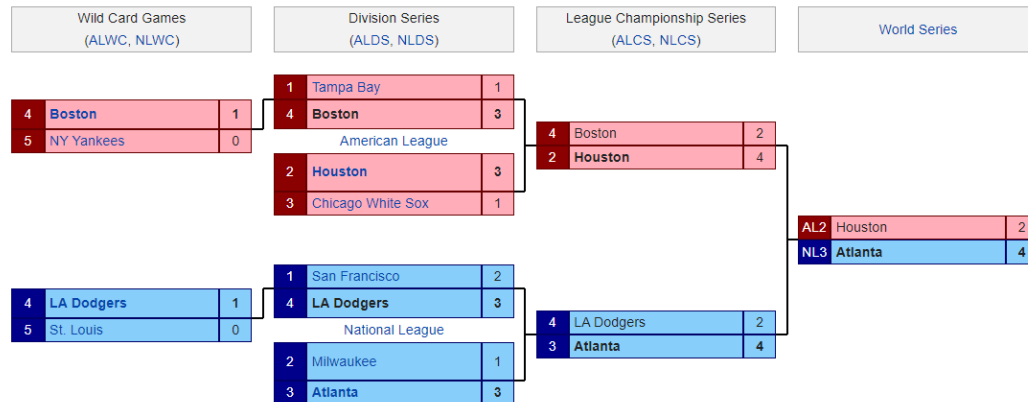


Figure 2: MLB Postseason Leagues (Wikipedia) [8]

1.4 Synergistic Feature Data

The determining factors for more complex ranking systems and win predictions rely on the synergistic data of the team's stats. A more comprehensive ranking system evaluates the interactions between variables like innings pitched and number of strikeouts. Modern and more computationally expensive methods of ranking and other deterministic sets can account for a more holistic approach to performing prediction or classification. With this in mind, one must ask how necessary is it to use all of the features available to one's analysis? If and when a simpler model should be implemented that has a similar or adequate accuracy is a question relevant to any sort of similar statistical endeavor.

1.5 Definitions of Ranking

There are many different methods to ranking baseball teams and other networks. Some ranking systems are more comprehensive and account for every piece of data available. Some of those variables include numbers of players divided into pitchers, batters, number of hits, runs, bases stolen, and different odds. While such complex models or ranking algorithms can be susceptible to overfitting, a lot of research, feature selection, parameter tuning, and amount of data results in an acceptable list of rankings.

One of the more popular and simpler methods is Elo ranking. This system assigns every team or player a score that determines the expected win ratio between a match. The winner of every match gains a number of points to their Elo score proportionate to the difference between the two player's respective scores and based off a K weight. The Elo ranking assumes that a team's performance is a random variable that follows a normal distribution, with the mean skill level being the true ranking that self corrects by gaining or losing points. This ranking also simplifies the different standard deviation between different teams to allow for highly simplistic calculations.

For example, one formula to calculate Elo ranking that can be easily modified is as follows [1]:

$$R_n = R_o + K * (W - W_e)$$

- R_n is the new rank
- R_o is the old rank
- K is the weight or K-factor
- W is the result of the game, encoded as 1 for win, 0.5 for a draw, and 0 for loss
- $W_e = \frac{1}{10^{-dr/400} + 1}$ is the expected result
- dr is the difference in rankings plus a constant, e.g 100

1.6 About Betting, Moneyline and Implied probability

Team ranking and match up prediction support a big industry world wide - sports betting. The betting involves moneyline betting and score betting in MLB. The difference is that moneyline betting is only related to the match up outcome(win or lose) and score betting is related to the score differences. We are talking about the moneyline betting here since our ranking is more related to the win-lose relationship.

The common odds used in MLB is American Odds. The odds of two match up teams will be given upon the betting page. Here is an example of ARI Diamondbacks vs. SIN Reds. It is ARI Diamondbacks (+100) and CIN Reds (-120). This mean for every 100 dollars someone bets on ARI Diamondbacks, the person gets 200 dollars back (100 + 100) if ARI Diamondbacks wins. If someone bets 100 dollars on CIN Reds, the person gets 188.33 dollars back ((100 + 100) / 120).



TOMORROW	RUN LINE		TOTAL		MONEYLINE
3:40PM					
 ARI Diamondbacks Madison Bumgarner	+1.5	-170	O 9.5	-115	+100
 CIN Reds Hunter Greene	-1.5	+150	U 9.5	-105	-120

Figure 3: Odds between ARI Diamondbacks and SIN Reds [2]

There is an implied probability we can tell from the odds given. Let's look at the ARI Diamondbacks (+100) and CIN Reds (-120) example again. The probability of ARI Diamondbacks wins is 50% ((100 + 100) / 100), the chance that CIN Reds is 54.54% (120 / (120 + 100)). Of notice is that the odds are floating, which means they vary as new bets go into the pool. One might wonder why the sum of probability is 104.54%, which is more than 100%. This is the reason why the betting company always wins. No matter which team wins the game, the betting company wins the 4.54%.

2 Exploratory Data Analysis

A total of 2466 baseball games were webscraped from [baseball-reference](#) using Python. This included the scores for both teams and the respective team names for the games, for which the season and postseason bracket was included. Newer games are seen going down the webpage and our dataframes.

2.1 Scatterplot

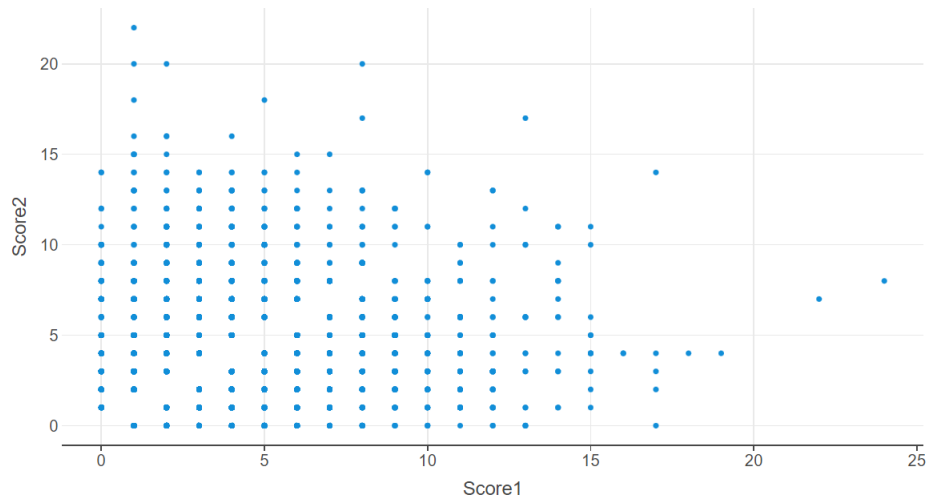


Figure 4: Scores of for two teams in a game

It can immediately be seen that there are no ties in the data, as a natural consequence of the play rules in the MLB. More notably, the clustering of the points in the bottom-left corner indicates that the scores for two teams playing against each other are have a negative linear relationship; when one team is doing very well, it tends to come at the cost of their opponent.

2.2 Barplot

This is a simple bar plot displaying the win-lose ratio of teams of the season.

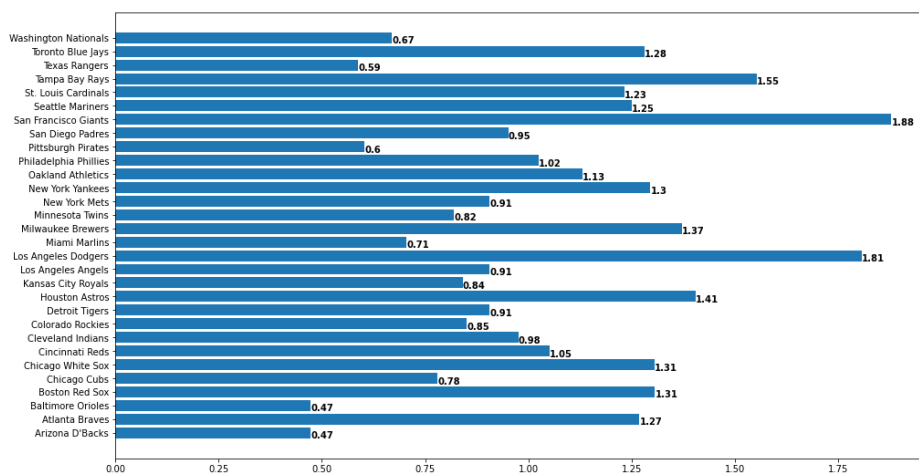


Figure 5: Odds between ARI Diamondbacks and SIN Reds

2.3 Heatmap

For the heatmap, we divided the Win Loss Ratio into three categories, Low, Medium, and High. We then plotted the heatmap with the number of wins and the average score per win. We were hoping to find a relationship between the number of wins a team has, and their average score per win and win-loss ratio. We were not able to find a relationship between the three variables. We assumed the higher the number of wins, the higher the average score and win-loss ratio, as the number of wins would indicate a higher capability of scoring more. However, the graph proves the opposite. The average scores are scattered across the map and across the levels of the win-loss ratio as well.

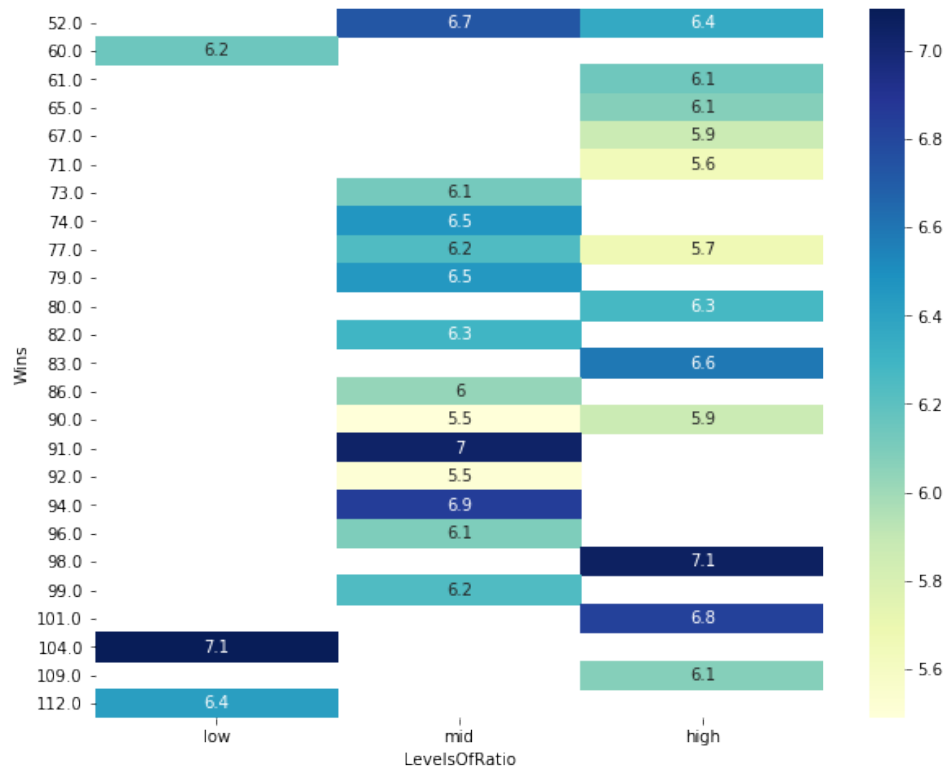


Figure 6: Heatmap with the number of wins and the average score per win

2.4 Heatmap

This heatmap displays the relationship between the level of the win-loss ratio, the average score per match and per win. Like the previous heatmap, we cannot find a relationship between the three variables. The values are scattered across the map and indicate no relationship with each other.

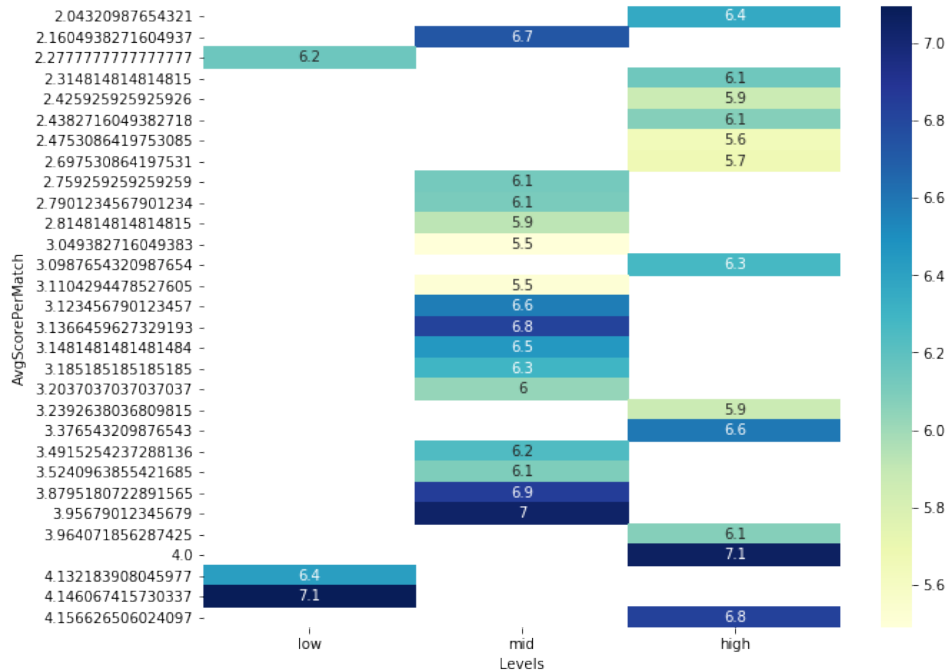


Figure 7: Heatmap with the average score each team earned per match and the average score per win

2.5 Network Graph

In this network graph, the nodes are the teams and the arrows point from the winning team to the losing team. It also shows us which team played against who. For example, the New York Yankees have not played a single game against the Los Angeles Dodgers. We can also see that in the tables below. As expected, the teams are mostly divided between the two leagues, with much fewer plays between leagues.

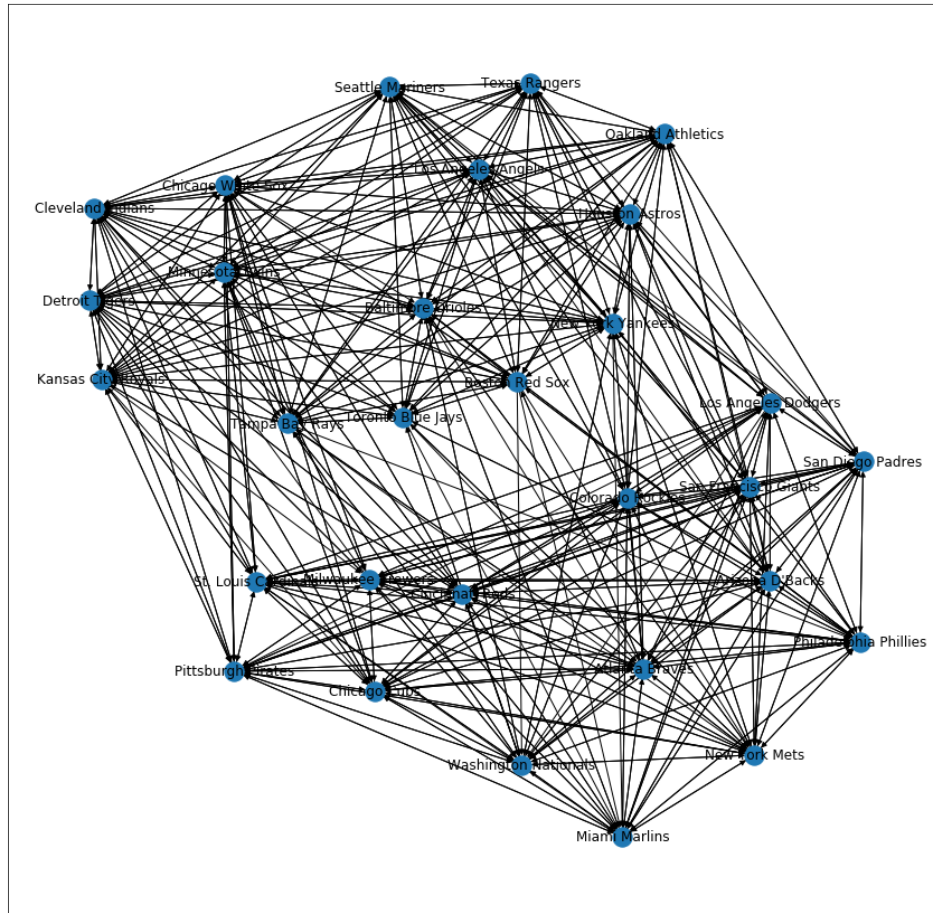


Figure 8: Heatmap with average score each team earned per match

2.6 Adjacency Table

This adjacency table (split into two vertically), shows us which team played with which team, which team won, and how many times the teams won against each other. The rows are the winning teams and the columns are the losing teams. For example, the Arizona D'Backs won 9 times against the Colorado Rockies. The Colorado Rockies won 10 times against the Arizona D'Backs. The tables also show us how often a team wins or loses to another team. Arizona D'Backs and Colorado Rockies seem to have an almost equal win rate when they play against each other. Milwaukee Brewers on the other hand, won 14 times when playing with the Pittsburgh Pirates. Pittsburgh has only won 5 times against Milwaukee.

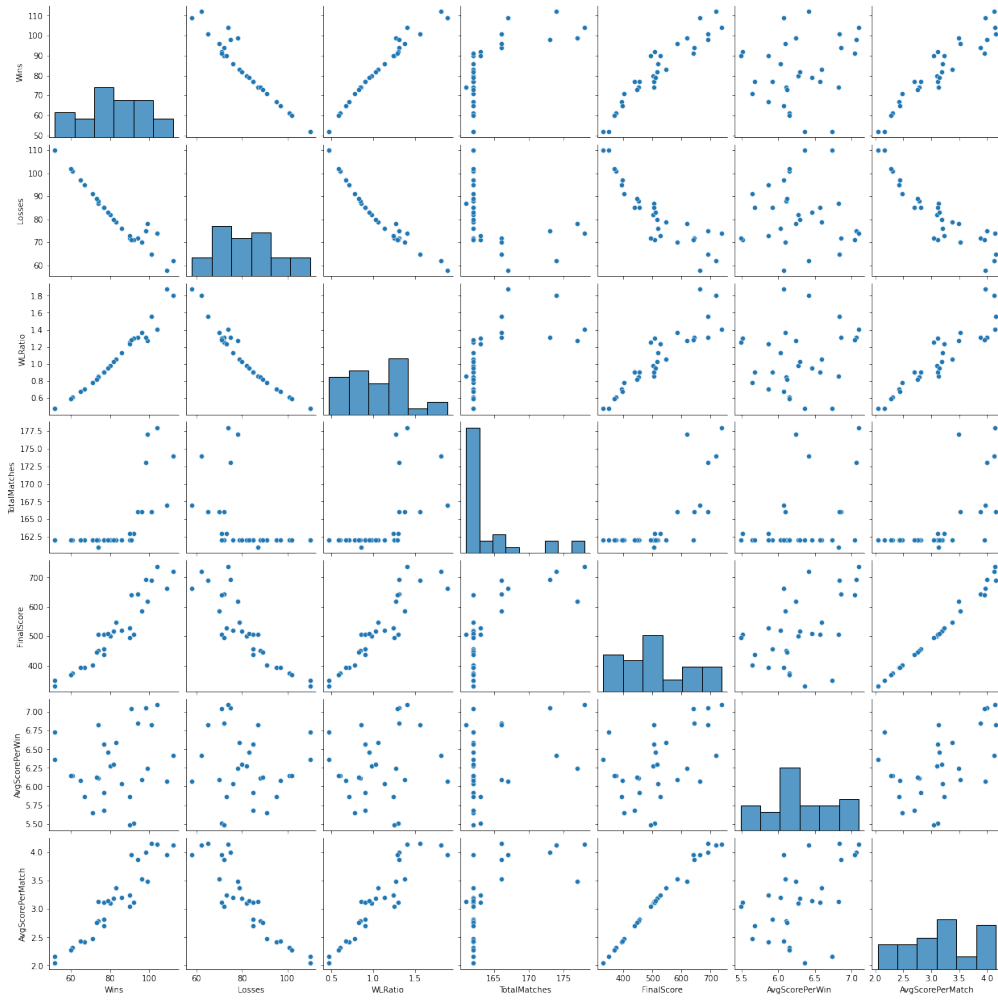
	Arizona D'Backs	Atlanta Braves	Baltimore Orioles	Boston Red Sox	Chicago Cubs	Chicago White Sox	Cincinnati Reds	Cleveland Indians	Colorado Rockies	Detroit Tigers	Houston Astros	Kansas City Royals	Los Angeles Angels	Los Angeles Dodgers	Miami Marlins
Arizona D'Backs	0	3	0	0	2	0	5	0	9	0	1	0	0	3	2
Atlanta Braves	4	0	3	1	5	0	4	0	2	0	4	0	0	6	11
Baltimore Orioles	0	0	0	6	0	0	0	2	0	2	3	4	2	0	2
Boston Red Sox	0	3	13	0	0	3	0	4	0	3	4	5	3	0	3
Chicago Cubs	4	2	0	0	0	1	8	1	3	2	0	0	0	4	1
Chicago White Sox	0	0	7	4	5	0	3	10	0	12	3	9	2	0	0
Cincinnati Reds	1	3	0	0	11	1	0	3	5	1	0	2	0	3	5
Cleveland Indians	0	0	5	2	3	9	3	0	0	12	1	14	5	0	0
Colorado Rockies	10	4	0	0	3	0	2	0	0	2	0	0	1	6	4
Detroit Tigers	0	0	5	3	1	7	2	7	0	0	5	8	1	0	0
Houston Astros	2	2	3	9	0	8	0	6	2	2	0	3	13	2	0
Kansas City Royals	0	0	3	2	3	10	1	5	0	11	4	0	2	0	0
Los Angeles Angels	3	0	4	3	0	5	0	1	2	6	6	4	0	3	0
Los Angeles Dodgers	16	6	0	0	3	0	3	0	13	0	2	0	3	0	3
Miami Marlins	5	8	2	0	5	0	2	0	2	0	0	0	0	4	0
Milwaukee Brewers	6	4	0	0	15	2	10	3	5	1	0	0	0	3	3
Minnesota Twins	0	0	4	2	2	6	2	11	0	11	4	9	2	0	0
New York Mets	5	9	3	0	3	0	3	0	5	0	0	0	0	1	10
New York Yankees	0	3	11	9	0	5	0	4	0	3	4	4	3	0	3
Oakland Athletics	4	0	3	3	0	3	0	4	2	6	8	5	15	1	0
Philadelphia Phillies	3	9	2	3	5	0	2	0	2	0	0	0	0	2	9
Pittsburgh Pirates	2	3	0	0	5	1	6	2	2	4	0	1	0	0	5
San Diego Padres	11	2	0	0	1	0	6	0	8	0	4	0	2	7	4
San Francisco Giants	17	3	0	0	6	0	6	0	15	0	2	0	3	12	4
Seattle Mariners	4	0	4	3	0	3	0	4	2	1	8	3	11	1	0
St. Louis Cardinals	6	1	0	0	10	1	9	2	4	1	0	5	0	3	6
Tampa Bay Rays	0	2	18	12	0	3	0	6	0	3	2	4	6	0	5
Texas Rangers	3	0	3	4	0	1	0	2	2	1	5	4	8	1	0
Toronto Blue Jays	0	6	14	9	0	3	0	5	0	3	2	4	3	0	4
Washington Nationals	4	5	3	0	3	0	2	0	2	0	0	0	0	0	11

Figure 9: Adjacency Table First Half

	Milwaukee Brewers	Minnesota Twins	New York Mets	New York Yankees	Oakland Athletics	Philadelphia Phillies	Pittsburgh Pirates	San Diego Padres	San Francisco Giants	Seattle Mariners	St. Louis Cardinals	Tampa Bay Rays	Texas Rangers	Toronto Blue Jays	Washington Nationals
Arizona D'Backs	1	0	1	0	0	4	4	6	2	2	1	0	1	0	3
Atlanta Braves	6	0	10	1	0	10	4	3	0	6	1	0	0	0	14
Baltimore Orioles	0	2	1	6	3	1	0	0	3	0	1	4	5	3	3
Boston Red Sox	0	5	4	11	3	3	0	0	4	0	11	3	10	3	3
Chicago Cubs	4	2	4	0	0	2	14	5	1	0	9	0	0	0	4
Chicago White Sox	1	13	0	1	4	0	3	0	0	3	2	3	5	4	0
Cincinnati Reds	9	2	3	0	0	4	13	1	1	0	10	0	0	0	5
Cleveland Indians	0	8	0	3	2	0	1	0	0	3	2	1	4	2	0
Colorado Rockies	2	0	2	0	1	5	4	11	4	2	3	0	4	0	4
Detroit Tigers	3	8	0	3	1	0	2	0	0	5	3	4	6	3	0
Houston Astros	0	3	0	2	11	0	0	2	1	11	0	4	14	4	0
Kansas City Royals	4	10	0	2	2	0	3	0	0	4	1	2	2	3	0
Los Angeles Angels	0	5	0	4	4	0	0	2	1	8	0	1	11	4	0
Los Angeles Dodgers	4	0	6	0	2	4	6	12	12	3	5	0	2	0	7
Miami Marlins	3	0	9	0	0	10	2	3	3	0	0	1	0	0	6
Milwaukee Brewers	0	2	4	0	0	2	14	5	4	0	6	0	0	0	5
Minnesota Twins	4	0	0	1	1	0	1	0	0	2	1	3	4	3	0
New York Mets	2	0	0	4	0	9	3	4	1	0	2	0	0	2	11
New York Yankees	0	6	2	0	4	2	0	0	0	5	0	8	6	8	2
Oakland Athletics	0	5	0	3	0	0	0	2	2	4	0	4	10	2	0
Philadelphia Phillies	5	0	10	2	0	0	4	4	2	0	4	0	0	1	13
Pittsburgh Pirates	5	2	4	0	0	3	0	3	4	0	7	0	0	0	2
San Diego Padres	2	0	3	0	2	2	4	0	8	3	3	0	3	0	4
San Francisco Giants	3	0	5	0	4	4	3	11	0	1	2	0	3	0	5
Seattle Mariners	0	4	0	2	15	0	0	0	2	0	0	6	13	4	0
St. Louis Cardinals	11	2	5	0	0	3	12	3	4	0	0	0	0	0	2
Tampa Bay Rays	0	3	3	11	3	4	0	0	0	1	0	0	3	11	1
Texas Rangers	0	3	0	1	9	0	0	0	1	6	0	4	0	2	0
Toronto Blue Jays	0	4	1	11	5	2	0	0	0	2	0	8	4	0	1
Washington Nationals	1	0	8	1	0	6	4	3	2	0	4	3	0	3	0

Figure 10: Adjacency Table Second Half

2.7 Pairplots



Pairplots allow us to see both distribution of single variables and relationships between two variables. Here we can see that losses have a negative correlation with win loss ratio. We can also conclude that wins and total matches are not linearly correlated. Wins are also positively correlated with average score per match. Losses are negatively correlated with Average score per match.

3 Statistical Rigor

3.1 Bayesian Principles

Statistical methods and models are like tools; they pertain to specific use cases and new ones can be invented for a problem. Assumptions allow for a reduction of the problem, and mathematics allows for a transformation so that this tool can be applied to the real world. The convolution within the transformation can cause theoretical oversights that have a real impact. Take the 1977 Case of *Castaneda v. Partida* as an example, in which Partida argued the court discriminates against Hispanics through jury selection. Partida demonstrated this with a binomial distribution (p =fraction of population that is Hispanic), as the last 120 jury members did not represent a random sample of the population. Partida was likely correct about the discrimination, but it was not a binomial distribution; it was a posterior beta distribution conditional on priors of observed population statistics (DeGroot, 2011, p.278 & p.293) [4], a minor correction but also a rejected plea.

In modern times, both the level of convolution and the number of statistical tools available have become greater, and priors and posteriors are often poorly considered. In place, the 1950's Rao-Blackwell theorem seems to have developed a unique relationship with the algorithms in modern technology wherein individuals can simply (and are often better off) add more crude estimators to explain variance instead of literally explaining the variance. Someone might use k-means to predict what clothes customers will buy, and not realize that they are assuming that someone who just bought a bunch of something would still want to buy a bunch more of basically those exact things. A tool may be approximately created for a contained problem, but be taken by others and confidently used on completely different and far more dynamic problems.

3.2 Emphasis

Statistical modelling requires context, and the MLB is an organization of the 30 best North American teams. A natural question that arises for statisticians then, is which teams are the best of the best. In fact this question is an existing commercial job for many statisticians in the industry of sports betting. In 2021, the Atlanta Braves won the World Series to the surprise of fans and statisticians alike, determining the context for our analysis. We will examine an existing ranking system, its assumptions, and how it pertained to rankings that other sports betters obtained.

With the MLB season, many constraints must be imposed in approaching a problem of scope, complexity, dynamicism, and feature dimensionality and variability. In certain systems such as a specific category of chess (i.e. Bullet), the ELO system is a fairly sufficient metric for player ranking. But we are not going to try to implement ELO, nor are we going to throw a strongest open source machine learning ensemble at it. It is important to be humble when approaching a problem of such complexity. There are issues beyond overfitting and underfitting, such as model survivorship bias; after 100 attempts to model the team's rankings, if there is a model or method that happens to meet their goal, the modeler should really try to keep that in mind. To flow through transformations of the problem we will be relying on the concept of "*information transitivity*" [5] and emphasizing critical assumptions, as

"There is no way that any modelling structure, such as the linearity in ranking, can model away the difficulties in accommodating the empirically observed divergent information and simultaneously achieve the nearly impossible task of retaining the classical transitivity constraint. The derived likelihood function simply ignores the divergent information and provides a compromised answer in a heavy-handed but an unknown fashion." [5]

3.3 Means for comparison

When comparing our team rankings to what others at the time concluded, we will be comparing them to the CBS Sports betting moneyline for the matches. Having observed the outcome, to us the probability that the Braves win the World Series is 100%. The posterior distribution has been collapsed and it may be easy to develop strong confirmation bias for particular parameter levels and models, so we will try to minimize convolution; Which is to idiomatically say "There exists many models that predict the Braves would win, but most of them aren't going to be good". On the flip-side however, analysts for the MLB do have the opportunity to dramatically adjust their odds estimates after bracket games, namely because the MLB is a fairly closed system in-between divisions, and

later changes have a smaller window to be adjusted. So the bracket itself will be considered as well, along with updating updating estimates.

4 Methodology

4.1 A Formulation of the Problem

In the interest of scope and feasibility, we will be ranking the teams through various methods based off only the scores of games during the MLB 2021 season. Since games in the MLB season exist in a closed system between teams in the MLB, we will consider these games a series of *pairwise conflicts* between units in dominance hierarchy. This constraint allows for the representation of teams as nodes in their network: the MLB. That is, the MLB and teams are also a directed graph of vertices and indexes for a non-symmetric adjacency matrix. The games are links in a network, nonzero entries in an adjacency matrix, or transition probabilities in a Markov chain.

How strong a team is depends on the strength of the teams it plays against and the same applies for the teams that the team plays against. It is a transitive recursive problem, meaning that paths are not bi-directional in this formulation. We can infer that if team A beats team B , and team B beats team C , then team A beats team C . This is the concept of a "*dominance path*", and we use it understanding that it does not necessarily hold true even after eliminating outcome variance with the CLT in a system where certain teams can counter another team (i.e. a system like rock paper scissors). This classical transitivity constraint is commonly imposed implicitly or explicitly in ranking hierarchies [5].

4.2 A Ranking System: Representation

As many different representations of the problem, names, representations, and notational norms have been created for various similar network ranking problems already, we must pick a representation to stick to. These various representations are fairly subjective relative to the rigor of information transitivity. As problems and methods become increasingly complex, an intuitive understanding of the transformation at hand becomes a stronger way of communicating the transformation. So we begin by declaring some notation.

Let $A_{new} \in \mathbb{R}^{n \times n}$ be a non-symmetric adjacency matrix representing the n teams in the MLB. If team i , t_i , loses to team j , t_j , then the entry of A_{ij} is the nonzero integer c_{ij} denoting the number of times that t_i lost to t_j . The sum of the i_{th} row of A , A_i , is simply the total number of times that t_i has lost, and of course then the sum of the j_{th} column of A , A_j , is the total number of times that t_j has won.

It is perhaps common knowledge amongst statisticians that a symmetric adjacency matrix of bi-directional connections in a graph, when raised to the power of two, has an interesting property: it provides the number of ways the number of walks of length 2 from index i to j . For the purpose of later transformations, we will provide two explanations of why this occurs. First, a proof of this claim, \mathbb{L}_2 ,

$$\begin{aligned} \mathbb{L}_2 : A_{ik} = 1, A_{kj} = 1 \\ (A^2)_{ij} &= \sum_{k=1}^n A_{ik} A_{kj} \\ &= |\{(A_{ik}, A_{kj}) | A_{ik} = 1, A_{kj} = 1, k \in \mathbb{R}^n\}| \end{aligned}$$

But more importantly, a meaningful interpretation, as these data structures are also interpretive. If you can walk from i to k , and you can walk from k to j , then you can walk from i to j in 2 trips through k . Or, equivalently, if team i gets consistently destroyed by team k , and team k gets consistently destroyed by team j , then you can infer that i will lose to team j .

Following the 1998 Google Pagerank algorithm, we can infer that team t_j 's dominance $dom(t_j)$ should generally follow this formula.

$$dom(t_j) = \sum_{t_i \in B(t_j)} \frac{dom(t_i)}{|D(t_i)|}$$

where $B(t_j)$ is all the teams that t_j beat.

This is a recursive function meaning that the normalized ranks for teams can be calculated with conventional recursion (with great difficulty and complex base/stop cases) and iteration in Python without any actual linear algebra, just like Google's Pageranks can. But this can also be calculated with linear algebra, which allows for more dynamicism in the problem.

In lieu of statistical rigor, it should be noted that the earlier concern regarding counterplay strategies, by this point, has been thrown out of the window. Wherein the above example provided by Hsieh

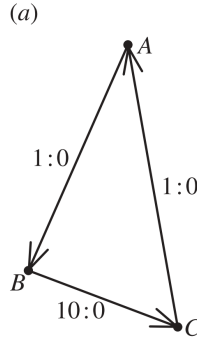


Figure 11: A Network [5]

demonstrates our described rock, paper, scissors scenario. In the above scenario, despite the only data on A being that they beat B and lost to C, neither of these are reflected under the Bradley-Terry Model's maximum likelihood estimates. This is very well explained as the wins A over B becoming a "fraction of a win".

4.3 An Existing Ranking System

Google's 1998 Pagerank algorithm is a popular ranking algorithm for directed graphs. Using generalized Pagerank as a guide, we can discuss the properties that allow for matrices such as A and transformations of the matrix in ranking MLB teams. The matrix power property above \mathbb{L}_2 holds for higher powers, providing us its utility. We can look at it as some sort of action, or a propagation through time.

What we are looking for from A is a matrix that allows for propagation, or perhaps a meaningful idempotent projection matrix that pertains to the problem. Given the above adjacency matrix, we can easily define an initial transformation on A to obtain our first projector P^T by normalizing the rows of A to obtain P .

$$P = [P_1, P_2, \dots, P_n]^T, P_i = \begin{cases} \frac{A_i}{|A_i|} & \text{if } |A_i| > 0 \\ 0_n & \text{otherwise} \end{cases} \quad (1)$$

Given a matrix of normalized team dominance scores, such a matrix should allow for the propagation of a team's dominance through their *dominance path*. The k^{th} state of the teams dominance scores, x_k , are propagated into the next $(k+1)^{th}$ state in this manner, and begins to reach an equilibrium after 60 iterations.

$$x_{k+1} = P^T x_k \quad (2)$$

4.4 Initial Look

Below is an implementation of P using games from the MLB 2021 Season. In this implementation, all the games have equal weight and the matrix can not account for changes in performance throughout the year. The above P matrix was normalized on A which contained absolute counts c_{ij} of the times team i lost to team j ; the propagation matrix is deterministic.

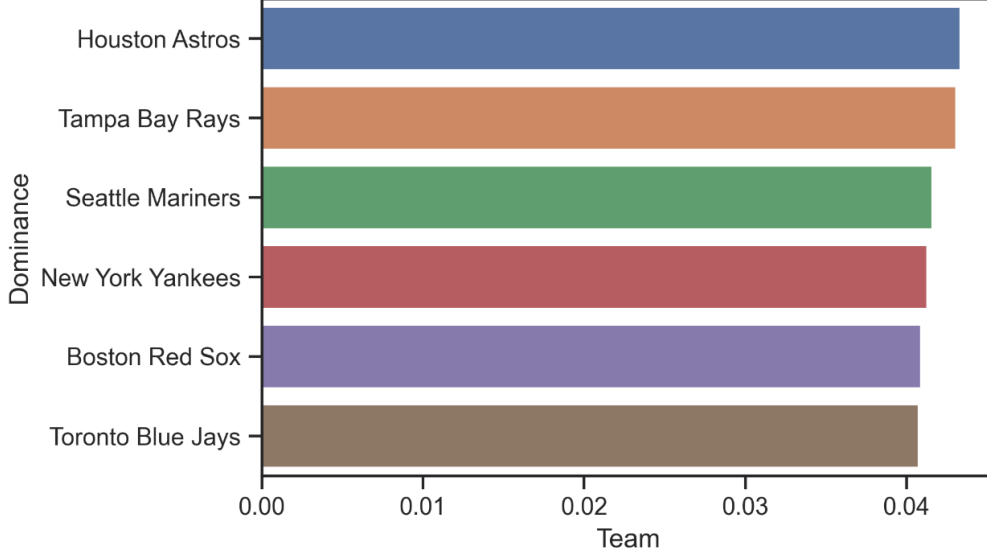


Figure 12: Ratings for Teams

Looking at the ranking of the teams, we can see that the stronger teams are indeed higher up. The only teams here that did not get to compete in the postseason bracket are the Toronto Blue Jays and the Seattle Mariners. The Mariners were second to the Houston Astros in their division, and the Blue Jays had extremely similar overall game statistics to the Yankees and Sox, two other teams in the AL East Division that made it to the bracket.

4.5 Damping parameter

In the MLB there are relatively few inter-league games during the season and also high volatility within the game itself. When creating a deterministic system for providing a one dimensional measure, the final order of the ranked teams, it intuitively follows that there needs to be some damping of the specific outcomes (i.e. bootstrapping). Once again, the Google Pagerank has a damping factor, for which we will now define a matrix M . With the damping factor d , and $n \times n$ matrix A ,

$$M = [M_1, M_2, \dots, M_n]^T, M_i = \begin{cases} d \frac{A_i}{|A_i|} + \frac{1-d}{n} & \text{if } |A_i| > 0 \\ (\frac{1}{n})_n & \text{otherwise} \end{cases} \quad (3)$$

The matrix M is to be used once again, in the form,

$$\pi_{k+1} = P^T \pi_k \quad (4)$$

or conversely in its dense form, since we are already assuming M 's convergence to a steady state, a simple linear regression problem.

$$\begin{aligned}\lim_{k \rightarrow \infty} x_k &= M^T x_k \\ \lim_{k \rightarrow \infty} x_k - M^T x_k &= 0_n \\ \lim_{k \rightarrow \infty} (I - M^T)x_k &= 0_n \\ x &= (I - M^T)^{-1}0_n\end{aligned}$$

In solving this for a nonzero solution, the first team is turned into a dummy variable with a row of zeros and a rank of 1. This corresponds to the Arizona Diamondbacks getting turned into a perfect rank team that has never beaten anyone. This adjustment is concerning, and should be typically avoided for the iterative method even if Arizona happens to be the weakest team in NL West.

5 The Team Rankings

With the damping parameter d set at a fairly standard 0.85, and now looking at the more recent 60% of games (arbitrarily selected) in 2021 for relevant comparison to postseason betting odds, the dense solution gives this.

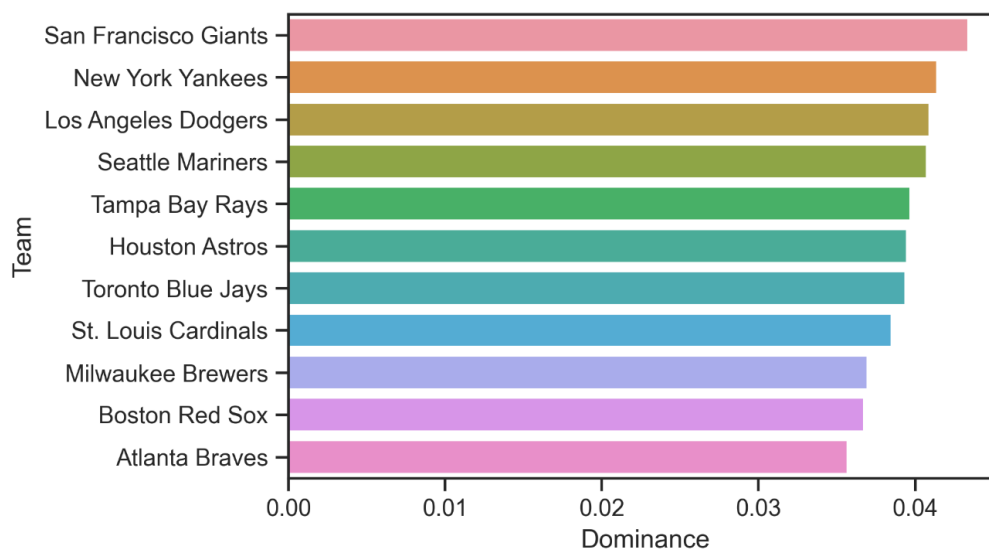


Figure 13: Top 11 Teams via projector M

5.1 Comparison

We decided to include the top 11 teams to include the Braves (who got placed at 11th). At a glance, we can see that these are all very strong teams. In terms of betting averages, M^T projection rates the Yankees well above the Boston Red Sox, which is in line with the Yankees being an overwhelming -175 favorite (bet 175 on Yankees to make 100 dollars). It also rates the Tampa Bay Devils higher than Boston, in line with Tampa being put the -135 favorite by sports analysts. Ironically, Boston would go on to beat both the Yankees and Tampa bay back to back in their first two games in the bracket. It doesn't feel so bad to be wrong when everyone (except Boston fans) is wrong. The rankings do not agree with analysts on the Dodgers vs the Giants, for which the analysts were right this time with the Dodgers winning 3-2 (games). The rankings put Houston above Boston, which is not in line with sports analysts having Boston as the -135 favorite for the League Champion Series, but Houston would end up winning 4-2 (games). [6]

We can see that the team of easily the most interest, the underdog that had their first World Series victory since 1995, is not in the top 10 of this ranking, despite having an extremely strong performance in the bracket. So what did the sports analysts think?

5.2 How Statisticians Were Wrong About Atlanta

The Braves we ranked 11th first bracket game was against the Milwaukee Brewers, who we ranked 9th. Sports analysts thought it looked worse for the Braves, as the Brewers were a strong favorite. *"Milwaukee is the -153 favorite (risk \$153 to win \$100) on the money line in the latest Brewers vs. Braves"*. [7] After the Braves proved the analysts wrong by winning 4-2, the analysts adjusted their estimates and said *"Los Angeles is the -145 favorite (risk \$145 to win \$100) in the latest Dodgers vs. Braves odds from Caesars Sportsbook, while Atlanta is a +143 underdog"*: a fairly even spread. The Braves definitely surprised a statistician or two as they beat the Dodgers in a sound 4-2 (games) victory. Maybe the analysts gave up a little after that, because in the world series, they only considered Houston a -115 favorite to win.

There seems to be a lesson to be learnt here, when the expected value of flipping a coin for each game is 1.5 games right and the analysts get a 0 for 3. The spread on the Dodgers versus Braves game seems to reflect a level of overconfidence in a black-box machine learning model. Bragging about model complexity is not necessarily a good after the fact.

2021 MLB playoffs: Red Sox vs. Rays odds, ALDS Game 2 picks, predictions from proven computer model

SportsLine's proven model has simulated Game 2 of Rays vs. Red Sox 10,000 times and revealed its MLB picks

By CBS Sports Staff Oct 8, 2021 at 5:08 pm ET • 3 min read

5.3 Limitations of Ranking

One of the predominant accepted ranking method for teams employed by the reputable organization FiveThirtyEight [3], since 2006, is elo rating. Every season the teams are rated and the qualifying teams are predicted through Monte Carlo simulations based off those ratings. As alluded to earlier, the elo ranking has a K weight that is determined by the volatility of the matchup. Generally, a newer player is more volatile because they are not yet established within the system. A lower K means weaker sensitivity that will fail to adequately rank a player early in their career.

The Atlanta Braves had 4 rookies in their 2021 lineup and several other relatively new players. Like most other prediction or ranking algorithms, we see that problems arise in the Elo ranking when there is a limited amount of data. This restriction is one of the contributing factors as to why the Braves were underestimated: Elo ratings are comparative based.

Referring back to the FiveThirtyEight [3] predictions, the Braves were ranked the seventh team in the preseason predictions and had a poor start. The rankings, updated each week, saw the Braves drop steadily, as in the 14th week of the season (out of 26), they dropped to rank 12 with less than a 1% chance to win the world series. One of the identified factors to their surprise win is the acquisition of key players mid-season. Some of these players made records with their excellent performances, but the models failed to account for these acquisitions as they happened. Would a more comprehensive model lead to a more accurate prediction? It is possible, but relying on player metrics to measure player rankings exposes similar new problems.

6 Conclusion

The data available for analyzing the 2021 MLB set of games is vast and readily available. Even with high powered machine learning algorithms, there is no predictive model or adequately relevant rankings available. The surprise of the Atlanta Brave's world series win is proof of this. This is despite the massive betting advantage that gives individuals monetary incentives to create such predictions. We have seen that the creation of such rankings rely on pairwise conflicts or interactions that are not equally distributed among the thirty teams. These class of rankings are susceptible to the changes within the synergistic features of each team, of which data and events occur during the season where bets may have already been made. For example, the addition of mid-season players performing exceptionally well are unpredictable events that are impossible to account for in preliminary models. This is also attributed to information transitivity- it is practically impossible to create such a perfect prior model.

References

- [1] World football elo ratings. <http://www.eloratings.net/about>, 2020.
- [2] Mlb betting odds. <https://sportsbook.draftkings.com/leagues/baseball/88670847>, 2021.
- [3] Boice, J. and Silver, N. 2021 MLB Predictions. <https://projects.fivethirtyeight.com/2021-mlb-predictions/>, 2021.
- [4] M. Degroot. *Probability and Statistics*. Pearson, 2012.
- [5] H. Fushing, M. P. McAssey, and B. McCowan. Computing a ranking network with confidence bounds from a graph-based beta random field. *Proceedings Royal Society. A*, 467:3590–3612, 08 2011.
- [6] CBS Sports Staff. 2021 mlb playoffs: Red sox vs. rays odds, alds game 2 picks, predictions from proven computer model, 2021.
- [7] CBS Sports Staff. Mlb weekend recap: Angels’ losing streak hits 11 as bryce harper leads comeback; yankees’ streak up to six, 2022.
- [8] Wikipedia contributors. 2021 major league baseball season — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=2021_Major_League_Baseball_season&oldid=1087540932, 2022.

7 Appendix

```
import math
import random
import statistics
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
import requests
import networkx as nx
from bs4 import BeautifulSoup
from timeit import default_timer
from numpy import unique, ravel
from numpy import sqrt, dot, array, diagonal, mean, transpose,
eye, diag, ones
from numpy import transpose, diag, dot
from numpy.linalg import svd, inv, qr, det
from sklearn.linear_model import LinearRegression

teams = pd.read_table('MLB_Teams_2021.txt', delimiter = ',')
teams.columns = teams.columns.str.lstrip()
teams.columns = teams.columns.str.rstrip()
teams["TotalMatches"] = teams["Wins"] + teams["Losses"]
games = pd.read_table('MLB_Games_2021.txt', delimiter = ',')
games.columns = games.columns.str.lstrip()
games.columns = games.columns.str.rstrip()
games["Number"] = np.arange(len(games))
games['WinningTeamName'] =
np.where(games['WinningTeam'] == 1, games['Team1'],
games['Team2'].str.lstrip())
games['FinalScore'] = np.where(games['WinningTeam']
== 1, games['Score1'], games['Score2'])
games['LosingTeamName'] =
np.where(games['WinningTeam'] == 2,
games['Team1'], games['Team2'].str.lstrip())
games.groupby(["WinningTeamName", "LosingTeamName"]).size()
winvl = pd.crosstab(games.WinningTeamName,
games.LosingTeamName).replace(0, np.nan).
stack().reset_index().rename(columns={0: 'Time'})
s = winvl.groupby(['WinningTeamName']).max()
wins = games[["WinningTeamName", "FinalScore"]].groupby("WinningTeamName").sum()
wins.index.name = 'WinningTeamName'
wins.reset_index(inplace=True)
df = pd.merge(teams,
              wins,
              left_on='Name',
              right_on='WinningTeamName',
              how='left')
df['AvgScorePerWin'] = df['FinalScore']/df['Wins']
df['AvgScorePerMatch'] = df['FinalScore']/df['TotalMatches']
Number = teams["Wins"].to_numpy()
WinningTeamName = teams["Name"].to_numpy()
height = Number
bars = WinningTeamName
y_pos = np.arange(len(bars))
figure(figsize=(12, 8), dpi=80)
plt.bar(y_pos, height)
plt.xticks(y_pos, bars, rotation='vertical')
plt.show()
Number = df["WLRatio"].to_numpy()
WinningTeamName = df["Name"].to_numpy()
```

```

height = Number
bars = WinningTeamName
y_pos = np.arange(len(bars))
figure(figsize=(12, 8), dpi=80)
plt.bar(y_pos, height)
plt.xticks(y_pos, bars, rotation='vertical')
plt.show()
Number = teams["TotalMatches"].to_numpy()
WinningTeamName = teams["Name"].to_numpy()
height = Number
bars = WinningTeamName
y_pos = np.arange(len(bars))
figure(figsize=(12, 8), dpi=80)
plt.bar(y_pos, height)
plt.xticks(y_pos, bars, rotation='vertical')
plt.show()
plt.scatter(df["FinalScore"], df["WLRatio"])
plt.scatter(df["FinalScore"], df["TotalMatches"])
sns.pairplot(df)
graph = nx.from_pandas_edgelist(winvl,
source = 'WinningTeamName', target = 'LosingTeamName',
edge_attr = 'Time', create_using = nx.DiGraph())
plt.figure(figsize = (15,15))
nx.draw_networkx(graph)
plt.show()
df1=df.sort_values("Wins", ascending=False, ignore_index=True)
df1["LevelsOfRatio"] = pd.cut(df["WLRatio"],
bins=[0, 0.6, 1.2, 1.9], include_lowest=True, labels=['low', 'mid', 'high'])
dfRatio = df1[["AvgScorePerWin", "Wins", "LevelsOfRatio"]]
dfRatio = dfRatio.pivot_table("AvgScorePerWin", "Wins", "LevelsOfRatio")
plt.figure(figsize=(10, 8))
ax = sns.heatmap(dfRatio, cmap="YlGnBu", annot=True)
dfRatio = df1[["AvgScorePerWin", "AvgScorePerMatch", "LevelsOfRatio"]]
dfRatio = dfRatio.pivot_table("AvgScorePerWin", "AvgScorePerMatch", "LevelsOfRatio")
plt.figure(figsize=(10, 8))
ax = sns.heatmap(dfRatio, cmap="YlGnBu", annot=True)
df = pd.crosstab(games.WinningTeamName, games.LosingTeamName)
idx = df.columns.union(df.index)
df = df.reindex(index = idx, columns=idx, fill_value=0)
df1 = df.iloc[:, 0:15]
df2 = df.iloc[:, 15:30]
print(df2)
dfi.export(df2, 'dataframe2.png')

```

```

import math
import random
import statistics
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from timeit import default_timer
from numpy import unique, ravel
from numpy import sqrt, dot, array, diagonal, mean, transpose, eye, diag, ones
from numpy import transpose, diag, dot
from numpy.linalg import svd, inv, qr, det
from sklearn.linear_model import LinearRegression

def topTeams(__PRs, __U, __n = 10, TR=False):
    dd = {i: __PRs[i] for i in range(len(__PRs))}
    dd = {__U[team]: rank for team, rank in sorted(dd.items(),
key=lambda kv: kv[1], reverse=True)}
    __teams = list(dd.keys())[:__n]
    dd = {team: dd[team] for team in __teams}

```

```

    if TR:
        return {"Link": list(dd.keys()), "Value": ravel(list(dd.values()))}
    return dd

def viewTopTeams(__PRs, __U, __n = 10):
    if type(__PRs[0]) in (list, np.array, np.matrix):
        __PRs = __PRs
    __topTeams = topTeams(__PRs, __U, __n)
    for __team in __topTeams:
        print(__team, __topTeams[__team])

import re
import requests
from bs4 import BeautifulSoup

link_raw = "https://www.baseball-reference.com/leagues/majors/2021-schedule.shtml"
link_data = requests.get(link_raw, headers={"Content-Type": "html"})
soup = BeautifulSoup(link_data.text, 'html.parser')
p = soup.find_all('p')

team_1 = []
score_1 = []
team_2 = []
score_2 = []

# 1,2,4,5
for i in range(15, 2686):
    game = str(p[i]).split("\n")
    if len(game) == 8:
        team1 = game[1].split("\>")[1][: -4]
        team_1.append(team1)
        team2 = game[4].split("\>")[1][: -4]
        team_2.append(team2)
        score1 = re.sub("[^0-9]", "", game[2])
        score_1.append(score1)
        score2 = re.sub("[^0-9]", "", game[5])
        score_2.append(score2)

score_1 = [int(s) for s in score_1]
score_2 = [int(s) for s in score_2]

def getOutcome(s1, s2):
    if s1 == s2:
        return 0
    if s1 > s2:
        return 1
    else:
        return 2

winning_team = [getOutcome(s1, s2) for s1, s2 in zip(score_1, score_2)]

df = pd.DataFrame({"Team1": team_1, "Team2": team_2, "Score1": score_1,
"Score2": score_2, "WinningTeam": winning_team})

team_names = list(unique(team_1 + team_2))

# which is also the number of teams in the MLB
print(len(team_names))

```

```

# p = 30 teams
# wins is just column sum
wins = [np.sum(A[:, j]) for j in range(p)]
losses = [np.sum(A[i]) for i in range(p)]
WL_ratio = [w/l for w,l in zip(wins, losses)]

# 2021 summary
dd_teams = {"Name": team_names, "Wins": wins, "Losses": losses, "WLRatio": WL_ratio}
df_teams = pd.DataFrame(dd_teams)

f1 = "MLB_Teams_2021.txt"
f2 = "MLB_Games_2021.txt"

team_vars = [dd_teams[key] for key in dd_teams]
header = ",_".join(list(df_teams.columns))

with open(f1, "w+") as f:
    f.write(header + "\n")
    for i in range(30):
        f.write(",_".join([str(col[i]) for col in team_vars])+"\n")

dd = {"Team1": team_1, "Team2": team_2, "Score1": score_1,
      "Score2": score_2, "WinningTeam": winning_team}
game_vars = [dd[key] for key in dd]
header = ",_".join(list(df.columns))

with open(f2, "w+") as f:
    f.write(header + "\n")
    for i in range(len(df)):
        f.write(",_".join([str(col[i]) for col in game_vars])+"\n")

p = len(team_names)
A = np.zeros((p,p))
A = np.matrix(A)

for i in range(1000, len(df)-50):
    # if team 1 won, then team 2 feeds/adds "dominance" to team 1
    if df.iloc[i]["WinningTeam"] == 1:
        ind_i = team_names.index(df.iloc[i]["Team2"])
        ind_j = team_names.index(df.iloc[i]["Team1"])
        A[ind_i, ind_j] += 1
    # vice versa
    if df.iloc[i]["WinningTeam"] == 2:
        ind_i = team_names.index(df.iloc[i]["Team1"])
        ind_j = team_names.index(df.iloc[i]["Team2"])
        A[ind_i, ind_j] += 1

# w count - l count
def makeB(_A, p): # p is variable count
    B = np.zeros((p,p))
    B = np.matrix(B)
    for i in range(p):
        for j in range(p):
            if i != j:
                c1 = _A[i, j]
                c2 = _A[j, i]
                if c2 > c1 and c1 != 0:

```

```

        B[j,i] = c2 - c1
    elif c1 > c2 and c2 != 0:
        B[i,j] = c1 - c2

    return B

# 1 0 who better
def makeC(_A, p): # p is variable count
    C = np.zeros((p,p))
    C = np.matrix(C)
    for i in range(p):
        for j in range(p):
            if i != j:
                c1 = _A[i, j]
                c2 = _A[j, i]
                if c2 > c1 and c1 != 0:
                    C[j,i] = 1
                elif c1 > c2 and c2 != 0:
                    C[i,j] = 1

    return C

B = makeB(A, p)
C = makeC(A, p)

def makeD(_A, p): # p is variable count
    D = np.zeros((p,p))
    D = np.matrix(C)
    for i in range(len(_A)):
        for j in range(len(_A[0])):
            rs = np.sum(_A[i])
            cs = np.sum(_A[:, j])
            ts = rs + cs - _A[i,j]
            D[i,j] = _A[i,j]/ts

    return D

D = makeD(A, p)

def convertToP(_A, tp=0.85):
    n = len(_A)
    ntp = (1-tp)/n
    ninv = 1/n
    P = []
    for i in range(n):
        r = []
        ri = np.sum(_A[i])
        if np.sum(_A[i]) > 0:
            for j in range(n):
                r.append(tp*( _A[i,j]/ri) + ntp)
        else:
            for j in range(n):
                r.append(ninv)
        P.append(r)
    return array(P)

def topTeamsFromMtx(__A, tp=0.85):
    P = convertToP(__A, tp)
    IP = np.eye(n) - P.T
    b = np.zeros(n).reshape(n,1)
    IP[0] = ones(n)
    b[0] = 1.0
    IP[0] = ones(n)
    u,d,v = svd(IP)

```

```

x = dot(v.T, inv(diag(d))).dot(u.T).dot(b)
viewTopTeams(x, team_names)

def topTeams2(__A, tp=0.85, cc=10):
    P = convertToP(__A, tp)
    IP = np.eye(n) - P.T
    b = np.zeros(n).reshape(n,1)
    IP[0] = ones(n)
    b[0] = 1.0
    IP[0] = ones(n)
    u,d,v = svd(IP)
    x = dot(v.T, inv(diag(d))).dot(u.T).dot(b)
    return topTeams(x, team_names, cc)

A_rnorm = A.copy()
for i in range(len(A_rnorm)):
    A_rnorm[i] = A_rnorm[i]/np.sum(A_rnorm[i])

A_cnorm = A.copy()
for j in range(len(A_cnorm)):
    A_cnorm[:,j] = A_cnorm[:,j]/np.sum(A_cnorm[:,j])

ttA = topTeams2(A, 0.85, 11)
sns.barplot(y="Team", x="Dominance",
data=pd.DataFrame({ "Dominance": ravel(list(ttA.values())),
"Team": list(ttA.keys()) }))

topTeamsFromMtx(A_rnorm)
topTeamsFromMtx(B)

P0 = convertToP(A)

x_PowSol = np.zeros(30).reshape(30,1)
for i in range(len(x_PowSol)):
    x_PowSol[i] = 1/len(x_PowSol)

for itr in range(100):
    x_PowSol = dot(P0.T, x_PowSol)

viewTopTeams(array(x_PowSol), team_names)

x_p1 = topTeams(array(x_PowSol), team_names, 6)
sns.barplot(x="Team", y="Dominance",
data=pd.DataFrame({ "Dominance": list(x_p1.keys()),
"Team": ravel(list(x_p1.values())) }))

name = df_teams['Name']
wl = df_teams['WLRatio']
fig, ax = plt.subplots(figsize=(16, 9))
ax.barh(name, wl)

for i in ax.patches:
    plt.text(i.get_width(), i.get_y(),
            str(round((i.get_width()), 2)),
            fontsize = 10, fontweight = 'bold',
            color = 'black')

```
