MLB Rankings and Matrix Methods

Zachary Carifa, Aidan Bartz - Sec 003

December 11, 2024

Abstract

This paper demonstrates the importance of linear algebra, as well as its setbacks, in sports analytics and in the development of team rankings. In the world of sports, Cinderella stories are prevalent, where the teams with the longest odds somehow find a way to win. Linear algebra serves useful in the world of sports analytics, in how it allows analysts to combine a plethora of stats and figures to deduce things like which team is the best, and who will win the most games. However, these rankings sometimes fail to accurately describe the true outcome of a season of sport. By using the Markov Chains and Perron-Frobenius Theorem, this paper develops a ranking algorithm for MLB teams based on statistics like: ERA+ and OPS+. Through scaling each statistic according to its correlation to team wins, this paper also discusses why certain teams that on paper should have won, failed and why statistically bad teams overcame the odds, as well as the shortcomings of statistical analysis in the world of sports.

Attribution

Zack performed the coding to organize our data, scale it, create the transition matrix, and calculate our rankings. He also wrote the abstract and Discussion sections of the paper and created most of the visuals for the paper. Aidan performed coding to calculate how to weigh the different statistics and wrote much of the paper.

Zachary Carifa
Aidan Bartz

Introduction

The goal of this project is to apply team performance analytics and the Perron-Frobenius Theorem to rank sports teams. Rather than ranking teams solely based on win-loss record, we will analyze a handful of statistics to evaluate teams more accurately. This way we can go more into a more in-depth analysis as to which teams are statistically better or worse than their record shows. For this project we will analyze the thirty teams in Major League Baseball over the 2024 season and evaluate them against each other. For our analysis, we selected three offensive stats and three pitching stats: OPS+ (On Base Percentage and Slugging Plus), which is the addition of on base percentage and slugging percentage adjusted to the league average and for ballpark dimensions, SO/9 (Strikeouts per 9 innings), and Runs Scored for offense. And for pitching, we selected: ERA+, which is the number of earned runs allowed per game adjusted to league average, SO% (Percentage of opponent hitters struck out), and Runs Allowed as our defensive statistics. All the statistics used in this project came from Baseball Reference, a reliable website that contains numerous different baseball statistics from as far back as 1871.

Markov Chains and Perron-Frobenius Theorem

The Perron-Frobenius Theorem states that a real square matrix will have a dominant eigenvalue, which refers to the eigenvalue with the largest absolute value. There will also be a dominant eigenvector, of all positive values that corresponds to the dominant eigenvalue.

Markov Chains are processes that show how states change over time. The transition matrix is a matrix that is formulated of probabilities that refer to the change of states. The rows of the matrix are the current states, with the columns being the state it could change to. The values are the probability that the current state (row value) will change to the new state (column value). For this project, the row variables of the transition matrix will be the teams. Suppose we are looking at Team A's row, the column variables will be the team that Team A is hypothetically playing. Let's take row variable Team A, and column variable Team B. Instead of the values referring to the probability that Team A changes to Team B, it explains the probability that Team A would beat Team B.

- *Col 1 – Team A, Col 2 – Team B, Col 3 – Team C*
- *Row 1 – Team A, Row 2 – Team B, Row 3 – Team C*

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Applying the Perron-Frobenius theorem to a transition matrix, you will get a dominant eigenvalue and dominant eigenvector. The dominant eigenvector is the steady state distribution of the Markov Chain.

Zachary Carifa
Aidan Bartz

Process

To accurately rank the teams based on our chosen statistics, we must create a transition matrix, consisting of the Markov Chain values that correspond to the probability that each team will beat one another. Then, following the Perron-Frobenius theorem, we find the dominant eigenvalue, and use that to find the dominant eigenvector. The dominant eigenvector, being the steady state distribution of the Markov chain, accounts for how a team would fare against every other team. This value is how we rank the teams against each other.

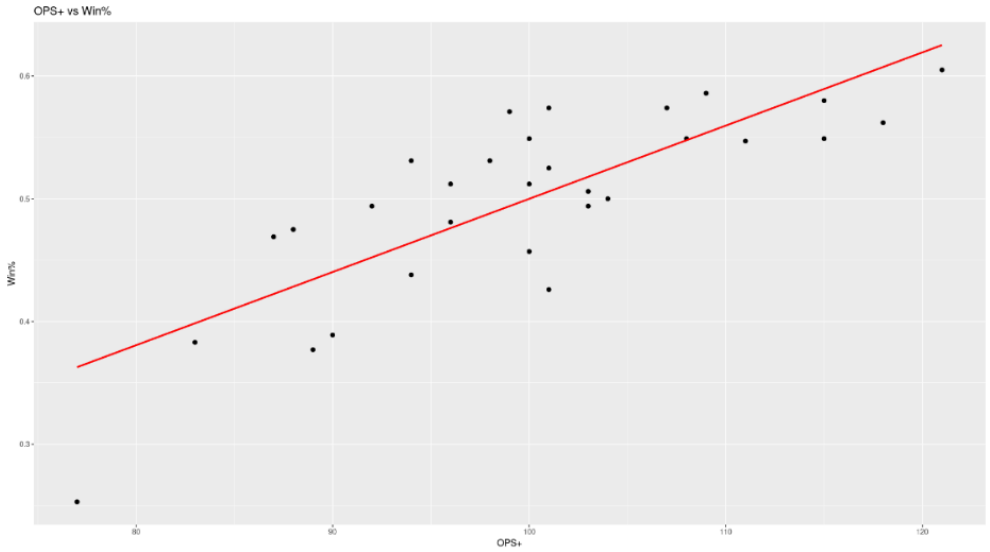
To start off, we create two 31-by-3 matrices, one for offensive statistics, and one for defensive statistics. We have teams as the rows of the matrices, and the statistics as the columns. The matrices are shown below. There are 31 rows for the 30 MLB teams and 1 for league average. (SEE APPENDIX 1)

a. Offense - **O**
   i. $$\begin{pmatrix} OPS+_1 & SO\%_1 & RS_1 \\ OPS+_2 & SO\%_2 & RS_2 \\ \dots & \dots & \dots \\ OPS+_{31} & SO\%_{31} & RS_{31} \end{pmatrix}$$
b. Defense - **D**
   i. $$\begin{pmatrix} ERA+_1 & SO9_1 & RA_1 \\ ERA+_2 & SO9_2 & RA_2 \\ \dots & \dots & \dots \\ ERA+_{31} & SO9_{31} & RA_{31} \end{pmatrix}$$

Note: for a couple of the stats, like SO/9 and RA, having a higher number is worse. To fix this, we will invert these values so that the larger number is now better than a lower number, without altering the data or rankings. These simplifies the calculation. (SEE APPENDIX 1)

One issue we ran into is that all the statistics are in different units. Run Scored and Runs Allowed are measured in runs, SO/9 and SO% are measured in strikeouts, and OPS+ and ERA+ are their own units that are adjusted to the league average. We must scale these values so that we can put them through the algorithm together. To do this we will scale all the statistics against the league average, whereas league average for each scaled statistic is 1. (SEE APPENDIX 1)

The next step is to weigh each statistic. Not all statistics are of the same level of importance. It is important that we weigh the different statistics based on the correlation it has with team wins. To determine how to weigh the different statistics, we investigated how much each metric was correlated with win percentage. Coding in R, we fit models of win percentage as a function of each individual variable, creating six models. To the right is the graph of OPS+ vs Win%. (SEE APPENDIX 10)



Next, we recorded the R squared value from each model to determine correlation; 1 being perfect correlation, 0 being no correlation. We will use these R squared values to weigh the variables. We then created a vector of the R squared values for the offensive and defensive stats respectively. Multiplying these vectors by the identity matrix, we get a resulting offense and defense matrix with the weight values on each column. We then multiply these matrices by our offensive and defensive stat matrices respectively. Now we have weighed each variable value by its correlation to win percentage. (SEE APPENDIX 2)

c. **OW**
   i. $\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$
d. **DW**
   i. $\begin{pmatrix} d & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & f \end{pmatrix}$

e. **OW * O** and **DW * D** represents the weighted offensive and defensive "scores" for each team, respectively. Call them **WO and WD**
f. a, b, c, d, e, f represents each metric's correlation to wins.

Then, add our offensive and defensive matrices together. We proceed to sum along each row of the matrix to get a vector that gives us the score for each team. Now we have a score for each team consisting of the sum of the scaled and weighted stats. (SEE APPENDIX 3)

g. **WO + WD = M**
   i. $M_{ij} = \text{WO}_{ij} + \text{WD}_{ij}$
h. $Scores \; -\!-\!> \; S_i = \sum_{j=1}^{3} M_{ij}, \; for \; i \in [1{,}31]$
i. Each row in *Scores* corresponds to a team, with i = 31 being league average, and i = 1 being the Arizona Diamondbacks.

i. $\mathbf{S} = \begin{pmatrix} Arizona\ Score \\ Atlanta\ Score \\ \dots \\ Washington\ Score \\ League\ Average\ Score \end{pmatrix}$

The next step is to create a transition matrix for the Markov Chain. We do this by taking each team's score and dividing it by the sum of itself and another team's score. We repeat this process until we have calculated the result for every combination of 2 teams. We end up with a square matrix, this is our transition matrix. This process compared each team's score against all 30 other teams in the matrix. (SEE APPENDIX 4)

j. The square transition matrix, A, compares the scores of each team against all the other teams, providing the probability of team i beating team j.
k. $\boldsymbol{A_{ij} = S_i/(S_i + S_j)}$, for i ≠ j; and if i = j, $A_{ij} = 0$

With our transition matrix, we must normalize it so that each column adds up to 1. This makes the matrix stochastic, a requirement for Perron-Frobenius theorem. To do this we divide each value by the sum of the column it's in. (SEE APPENDIX 4)

l. $A_{ij} = \frac{A_{ij}}{\sum_{j=1}^{31} R_{ij}} \; for \; i \in [1, 31]$

Next, we utilize the Perron-Frobenius Theorem. We find the eigenvalues of our matrix, singling out the largest one. Then we calculate the eigenvector corresponding to the largest eigenvalue. (SEE APPENDIX 4)

a. **A * R = λ * R**
b. **R** is the **Dominant eigenvector** that satisfies the above equation, corresponding to the **Dominant eigenvalue, λ = 1.** This is the largest eigenvalue in any stochastic matrix.
c. **R** represents the steady-state distribution of rankings
d. Below is the dominant eigenvector, **R** corresponding to λ = **1**

```
[0.18463089 0.18550109 0.18503686 0.18060704 0.18181458 0.16320985
 0.17763413 0.18406219 0.16673855 0.18023941 0.18608861 0.18339455
 0.17057787 0.18800428 0.16905574 0.18496617 0.18118213 0.18356477
 0.18815745 0.17376292 0.18539019 0.17402702 0.18558565 0.18240756
 0.17893139 0.17722338 0.17629744 0.17600651 0.17656168 0.1743145
 0.17944008]
```

Now that we have our eigenvector, we normalize it to 1 by dividing every value by the sum of all the values. Lastly, we scale the values from 0 to 100 to make the scores easier to interpret. This results in our final team rankings. (SEE APPENDIX 5, 6, 8)

e. The normalized eigenvector $\mathbf{R_{final}}$ represents the rankings of all the teams.
   i. $R_{final,i} = \frac{R_i}{\sum_{j=1}^{31} R_j}$
f. $R_{scaled,i} = \frac{R_{final,i} - \min(R_{final})}{\max(R_{final}) - \min(R_{final})} * 100$

Zachary Carifa
Aidan Bartz

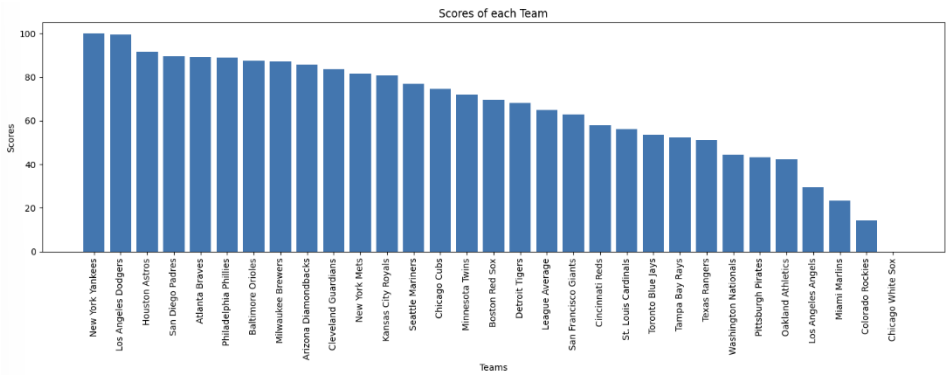
Results

Our calculated rankings differ slightly from the true performance of each team; however, our results are very promising. In particular, the White Sox remain in last between the rankings, though this makes sense as the 2024 White Sox had the worst ever season in the 100+ year history of the MLB. Our rankings had the Yankees and Dodgers at 1 and 2 respectively, who were the 2 teams to compete in the World Series this year, although it was the Dodgers that won. A noticeable difference between our rankings and the standings was the Philadelphia Phillies. The Phillies had the 2nd best record but were 6th in our rankings. They ended as a first-round exit in the playoffs. There were also a lot of similarities between our rankings and the standings, with 15 teams being within one spot of their actual ranking. These rankings can help us understand which teams are overperforming or underperforming their expectations. (SEE APPENDIX 7,8,9)

Our rankings (Actual Final Season Rank):

1. New York Yankees (3)
2. Los Angeles Dodgers (1)
3. **Houston Astros (11)**
4. San Diego Padres (5)
5. Atlanta Braves (9)
6. Philadelphia Phillies (2)
7. Baltimore Orioles (7)
8. Milwaukee Brewers (4)
9. Arizona Diamondbacks (8)
10. Cleveland Guardians (6)
11. New York Mets (10)
12. Kansas City Royals (13)
13. Seattle Mariners (14)
14. Chicago Cubs (15)
15. Minnesota Twins (17)
16. Boston Red Sox (18)
17. **Detroit Tigers (12)**

    LEAGUE AVERAGE
18. San Francisco Giants (19)
19. Cincinnati Reds (22)
20. St. Louis Cardinals (16)
21. Toronto Blue Jays (24)
22. Tampa Bay Rays (20)
23. Texas Rangers (21)
24. Washington Nationals (25)
25. Pittsburgh Pirates (23)
26. Oakland Athletics (26)
27. Los Angeles Angels (27)
28. Miami Marlins (28)
29. Colorado Rockies (29)
30. Chicago White Sox (30)



We can calculate the accuracy of our ranking system by taking the absolute values of the differences and averaging them out. We use the following equation:

$$\frac{|NYY_{real} - NYY_{ours}| + |LAD_{real} - LAD_{ours}| + \ldots + |CWS_{real} - CWS_{ours}|}{30}$$

This gives us an error per team of **2 ranking spots**. Meaning our ranking is accurate to +/- 2 spots in the actual rankings.

Discussion

Through analysis of our results, we discovered that our algorithm does a very good job of determining team rankings, with an error of 2 ranking positions. Our results were largely as expected, with little to no change with most of the teams, however, there were a few outliers. For example, the Houston Astros finished the season 8 spots below where our algorithm predicted them to finish, while the Detroit Tigers finished 5 spots above their predicted rank. Teams like these add a large amount of excitement, and disappointment, to the world of sports. Two teams exhibiting both extremes of the unpredictable are the 2010 Chargers and the 2007 Giants, both NFL. The San Diego Chargers finished the 2010 NFL Season with the #1 Offense and #1 Defense in the NFL. Numbers like these would incline a reader to believe they were the best team in the league that year. However, the Chargers ended up having the worst special teams in the NFL that year and missed the playoffs, falling well short of projections and their statistical ranks. On the other hand, the 2007 New York Giants barely made the playoffs, making it in as a Wild Card Team (one that does not win the division, but won enough games to make the playoffs). Despite all odds, they made it all the way to the Super Bowl to defeat the undefeated New England Patriots (arguably the greatest team of all time). Examples like these emphasize the scenarios where analyzing statistics and ranking algorithms come up short in predicting outcomes. The results we found also provide insight into which statistics play a bigger role in winning games. By inputting different statistics, rather than the ones used, we can compare the true results of the 2024 MLB season to each variation of our algorithm, each a different variety of statistics.

We originally set out to create an accurate team-ranking algorithm applicable to many sports. Despite focusing on a single season of baseball, our code can be applied to any sport by simply changing the statistics used, as well as the weights of each statistic. Throughout this whole process, we determined that developing an accurate ranking system is possible, but there will always be outliers like the 2007 Giants and 2010 Chargers that can’t be accounted for, due to unknown factors. Although sports analytics is not a perfect science, there is always more work to be done. The MLB recently (10 years ago) introduced Stat Cast, which provided new values to analyze, such as swing velocity, launch angle, and pitch spin rate. This is just one example of the never-ending progression of sports analytics. There are always more metrics to analyze, other factors to winning, and unknown combinations of these metrics that create more accurate ranking systems. We can only build a stronger understanding of sports analytics through comparing different statistics, comparing different eras of different sports, and finding correlation.

References

Baseball Reference. “MLB Stats, Scores, History, & Records | Baseball-Reference.com.” *Baseball-Reference.com*, www.baseball-reference.com/.

Sekhon, Rupinder, and Roberta Bloom. “10.1: Introduction to Markov Chains.” *Mathematics LibreTexts*, 22 Mar. 2020, math.libretexts.org/Bookshelves/Applied_Mathematics/Applied_Finite_Mathematics_(Sekhon_and_Bloom)/10%3A_Markov_Chains/10.01%3A_Introduction_to_Markov_Chains.

Wild Card Standings MLB.com. “Wild Card Standings.” *MLB.com*, www.mlb.com/standings/wild-card.

The ranking of incomplete tournaments: A Mathematician’s guide to popular sports, by T Jech, The American Mathematical Monthly (1983).

The Perron-Frobenius theorem and the ranking of football teams, by JP Keener, SIAM Review (1993).

Appendix

1. Defining Variables, Vectors, and Matrices (All team stats come from pro baseball (reference)

```
# Import Necessary Packages
import numpy as np
import math

# Define a separate teams vector for ease of calculation. Contains all MLB teams in Alphabetical Order, with league average at the end.
Teams = np.array(['Arizona Diamondbacks', 'Atlanta Braves', 'Baltimore Orioles', 'Boston Red Sox', 'Chicago Cubs', 'Chicago White Sox',
                  'Cincinnati Reds', 'Cleveland Guardians', 'Colorado Rockies', 'Detroit Tigers', 'Houston Astros', 'Kansas City Royals',
                  'Los Angeles Angels', 'Los Angeles Dodgers', 'Miami Marlins', 'Milwaukee Brewers', 'Minnesota Twins', 'New York Mets',
                  'New York Yankees', 'Oakland Athletics', 'Philadelphia Phillies', 'Pittsburgh Pirates', 'San Diego Padres', 'Seattle Mariners',
                  'San Francisco Giants', 'St. Louis Cardinals', 'Tampa Bay Rays', 'Texas Rangers', 'Toronto Blue Jays', 'Washington Nationals',
                  'League Average'])

# Define a vector for each statistic we use. The ith entry in each statistic vector corresponds to the ith team in the teams vector.
# Also normalize each value compared to league average. This makes the average value equal to 1 for each metric.
# For statistics like Offensive Strikeouts and Runs Allowed, a larger number is worse, and less is better.
  # For these values, we invert them before comparing to league average, so values greater than 1 are still "better" than values less than 1
   # This does not altar the rankings

# OFFENSE
# OPS+ - On-Base% Plus Slugging normalized across the league - league avg = 100
OPS = np.array([115, 100, 118, 104, 100, 77, 88, 99, 89, 94, 111, 98, 90, 121, 83, 101, 103, 108, 115, 101, 109, 87, 107, 103, 101, 96, 92,
                96, 100, 94, 100]) / 100

# SO - Offensive Strikeouts per game (inverted) - league avg = 8.6
SO = (162 / np.array([1265, 1461, 1359, 1570, 1362, 1403, 1459, 1196, 1617, 1461, 1176, 1161, 1416, 1336, 1410, 1459, 1306, 1382, 1326, 1502, 1370,
               1506, 1077, 1625, 1452, 1318, 1485, 1284, 1233, 1220, 1373]))*8.6

# Runs - Runs scored per game - League avg = 4.39
Runs = (np.array([886, 704, 786, 751, 736, 507, 699, 708, 682, 682, 740, 735, 635, 842, 637, 777, 742, 768, 815, 643, 784, 665, 760, 676, 693, 672, 604, 683,
                  671, 660, 711]) / 162)/4.39

# DEFENSE
# ERA+ - Earned Runs allowed per 9 innings normalized across the league - league avg = 100
ERA = np.array([91, 120, 96, 106, 106, 89, 108, 114, 84, 113, 106, 113, 92, 99, 96, 116, 98, 100, 110, 91, 106, 101, 107, 106, 95, 104, 106,
                91, 94, 94, 101]) / 100

# SO9 - Strikeouts pitched per 9 innings - league avg = 8.6
SO9 = np.array([8.2, 9.7, 8.6, 8.4, 8.5, 8.7, 8.6, 8.9, 7.1, 8.4, 9.3, 8.4, 7.9, 8.7, 8.2, 8.5, 9.4, 9.1, 9, 7.9, 8.9, 8.5, 9.1, 8.9, 9, 8.2,
                8.8, 8.6, 8.3, 8.2, 8.6])/8.6

# RunsP - Runs allowed per game - league avg = 4.39
RunsP = (162 / np.array([788, 607, 699, 747, 669, 813, 694, 621, 929, 642, 649, 644, 797, 686, 841, 641, 735, 697, 668, 764, 671, 739, 669, 607,
                    699, 719, 663, 738, 743, 764, 711]))*4.39

# Combine the offensive statistics into an offensive matrix, and the defensive statistics into a defensive matrix
# Each stat-vector represents a column in the matrix. Each row still corresponds to the alphabetized teams
O = np.column_stack([OPS, SO, Runs])
D = np.column_stack([ERA, SO9, RunsP])
```

2. Defining Weight Matrices

```
# Define 2 Diagonal matrices, containing the correlation between each statistic and team wins.
# This weighs each statistic in order to more accurately represent the data
Ow = np.column_stack([[0.6388, 0, 0], [0, 0.1039, 0], [0, 0, 0.6613]])
Dw = np.column_stack([[0.3822, 0, 0], [0, 0.3110, 0], [0, 0, 0.5338]])

# Multiply the Offensive weight matrix and the Offensive Statistic matrix. This gives us a weighted "score" for each statistic
# Do the same for defesne
WeightedO = np.dot(O, Ow)

WeightedD = np.dot(D, Dw)

# Check the matrices
WeightedO, WeightedD
```

3. Combining matrices into one “score” vector

```
# Add the weighted offensive and weighted defensive matrices. Then sum each row to attain a column vector representing the "scores" of each team.
# The ith entry in score corresponds to the ith team in Teams.
M = WeightedO + WeightedD
Score = np.sum(M, axis = 1)
```

4. Use of Perron-Frobeneus Theorem

```
# Create a square transition matrix, to allow for the use of Perron Frobeneus theorem
transition_matrix = np.array([[Score[i] / (Score[i] + Score[j]) if i != j else 0 for j in range(len(Score))]for i in range(len(Score))])

# Normalize the matrix so each column sums to 1
transition_matrix = transition_matrix / np.sum(transition_matrix, axis=0)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(transition_matrix)

# Find the eigenvector corresponding to the largest eigenvalue (1)
dominant_eigenvalue_index = np.argmax(eigenvalues)
dominant_eigenvector = np.real(eigenvectors[:, dominant_eigenvalue_index])

# Normalize the eigenvector for rankings
final_rankings = dominant_eigenvector / np.sum(dominant_eigenvector)
```

5. Scale the rankings

```
# Scale rankings from 0 to 100
scaled_rankings = (final_rankings - min(final_rankings)) / (max(final_rankings) - min(final_rankings)) * 100
```

6. Display data sorted from best to worse

```
# Display the rankings sorted from best to worst
# Store the ranked team names and ranked scores
team_names = []
team_scores = []
for team, rank in sorted(zip(Teams, scaled_rankings), key=lambda x: x[1], reverse=True):
    print(f"{team}: {rank:.0f}")
    team_names.append(team)
    team_scores.append(rank)
```

7. Graph the Data

```
# Plot a bar chart of each teams rankings
# Import plotting library
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 6))
plt.bar(team_names, team_scores)
plt.xlabel("Teams")
plt.ylabel("Scores")
plt.title("Scores of each Team")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

8. Code Results

```
New York Yankees: 100
Los Angeles Dodgers: 99
Houston Astros: 92
San Diego Padres: 90
Atlanta Braves: 89
Philadelphia Phillies: 89
Baltimore Orioles: 87
Milwaukee Brewers: 87
Arizona Diamondbacks: 86
Cleveland Guardians: 84
New York Mets: 82
Kansas City Royals: 81
Seattle Mariners: 77
Chicago Cubs: 75
Minnesota Twins: 72
Boston Red Sox: 70
Detroit Tigers: 68
League Average: 65
San Francisco Giants: 63
Cincinnati Reds: 58
St. Louis Cardinals: 56
Toronto Blue Jays: 54
Tampa Bay Rays: 52
Texas Rangers: 51
Washington Nationals: 45
Pittsburgh Pirates: 43
Oakland Athletics: 42
Los Angeles Angels: 30
Miami Marlins: 23
Colorado Rockies: 14
Chicago White Sox: 0
```

9. MLB 2024 Final Regular Season Standings

TEAM	W	L	PCT
Los Angeles Dodgers z	98	64	.605
Philadelphia Phillies y	95	67	.586
New York Yankees z	94	68	.580
Milwaukee Brewers y	93	69	.574
San Diego Padres w	93	69	.574
Cleveland Guardians y	92	69	.571
Baltimore Orioles w	91	71	.562
Arizona Diamondbacks	89	73	.549
Atlanta Braves w	89	73	.549
New York Mets w	89	73	.549
Houston Astros y	88	73	.547
Detroit Tigers w	86	76	.531
Kansas City Royals w	86	76	.531
Seattle Mariners	85	77	.525
Chicago Cubs	83	79	.512
St. Louis Cardinals	83	79	.512
Minnesota Twins	82	80	.506
Boston Red Sox	81	81	.500
San Francisco Giants	80	82	.494
Tampa Bay Rays	80	82	.494
Texas Rangers	78	84	.481
Cincinnati Reds	77	85	.475
Pittsburgh Pirates	76	86	.469
Toronto Blue Jays	74	88	.457
Washington Nationals	71	91	.438
Athletics	69	93	.426
Los Angeles Angels	63	99	.389
Miami Marlins	62	100	.383
Colorado Rockies	61	101	.377
Chicago White Sox	41	121	.253

10. Code in R providing weight values for each statistic

```
library(tidyverse)

#Create stats vectors

OPS_plus <- c(115, 100, 118, 104, 100, 77, 88, 99, 89, 94, 111, 98, 90, 121, 83, 101, 103, 108, 115, 101, 109, 87, 107, 103, 101, 96, 92, 96, 100, 94)

hSO <- c(1265, 1461, 1359, 1570, 1362, 1403, 1459, 1196, 1617, 1461, 1176, 1161, 1416, 1336, 1410, 1459, 1306, 1382, 1326, 1502, 1370, 1506, 1077, 1625, 1452, 1318, 1485, 1284, 1233, 1220)

AB <- c(5522, 5481, 5567, 5577, 5441, 5383, 5325, 5310, 5454, 5442, 5530, 5421, 5357, 5522, 5522, 5472, 5490, 5510, 5450, 5432, 5534, 5477, 5526, 5330, 5460, 5507, 5389, 5472, 5410, 5374)

R <- c(886, 704, 786, 751, 736, 507, 699, 708, 682, 682, 740, 735, 635, 842, 637, 777, 742, 768, 815, 643, 784, 665, 760, 676, 693, 672, 604, 683, 671, 660)

G <- c(162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162, 162)

ERA_plus <- c(91, 120, 96, 106, 106, 89, 108, 114, 84, 113, 106, 113, 92, 99, 96, 116, 98, 100, 110, 91, 106, 101, 107, 106, 95, 104, 106, 91, 94, 94)

pSO <- c(1313, 1553, 1380, 1353, 1348, 1366, 1370, 1410, 1118, 1354, 1479, 1339, 1252, 1390, 1317, 1373, 1500, 1455, 1457, 1263, 1433, 1356, 1453, 1416, 1436, 1308, 1406, 1371, 1314, 1314)

RA <- c(788, 607, 699, 747, 669, 813, 694, 621, 929, 642, 649, 644, 797, 686, 841, 641, 735, 697, 668, 764, 671, 739, 669, 607, 699, 719, 663, 738, 743, 764)

Win <- c(.549, .549, .562, .500, .512, .253, .475, .571, .377, .531, .547, .531, .389, .605, .383, .574, .506, .549, .580, .426, .586, .469, .574, .494, .525, .512, .494, .481, .457, .438)

#Create a dataframe

DF <-data.frame(OPS_plus, hSO, AB, R, G, ERA_plus, pSO, RA, Win)
DF1 <- DF %>% mutate(hSO_per9 = hSO/9, R_per_game = R/G, pSO_pergame = pSO/9)
#Train model to estimate win percentage
model1 <- lm(Win~OPS_plus, data = DF1)
model2 <- lm(Win~hSO_per9, data = DF1)
model3 <- lm(Win~R_per_game, data = DF1)
model4 <- lm(Win~ERA_plus, data = DF1)
model5 <- lm(Win~pSO_pergame, data = DF1) model6 <- lm(WinRA, data = DF1) model7 <- lm(Win~.-SO-AB-G-R, data=DF1)

#Evaluate performance by finding R squared values, which gives the correlation of each statistic

summary(model1)
summary(model2)
summary(model3)
summary(model4)
summary(model5)
summary(model6)
summary(model7)

#Plot OPS+ vs Win%

ggplot(aes(x=OPS_plus, y=Win), data=DF1) +
        geom_point(color = "black", size = 2) + geom_smooth(method = "lm", se = FALSE, color = "red") +
        labs(title = "OPS+ vs Win%", x = "OPS+", y = "Win%")
```