Who is Our NBA Champion?

Predicting NBA Games using Monte Carlo Simulations and MCMC Methods

Ryan Duchemin and Tomas Meade
Vassar College
December 22, 2020
Ming-Wen An

Introduction

The COVID-19 pandemic led to a mass cancellation and postponement of sporting events worldwide. One of these sporting events was the United States' NBA basketball season. When one player, Utah Jazz center Rudy Gobert, received a positive COVID-19 diagnosis on March 11, 2020, the NBA elected to suspend the final month of the season [1]. At the time of postponement all teams had played between 63 and 67 of their traditional 82 game schedule. A few months later, the NBA announced that some teams could compete in a "bubble" in Orlando, FL. Following these games, the NBA standings would be set, and the playoffs would commence. In October, the Los Angeles Lakers defeated the Miami Heat in a best-of-seven series to win the 2020 NBA Finals. This project's focus is to investigate and predict if these results, both the playoff bracket and NBA champion would change had the COVID-19 pandemic not occurred. We ran a simple simulation using data from the 2019-2020 NBA games prior to postponement to predict the outcomes of the 2019-2020 NBA games postponed. We then created a playoff bracket based around the team standings after our regular season simulation. The playoffs were then conducted up to the NBA finals. Our model was then improved upon by implementing Markov Chain Monte Carlo methods, specifically the Metropolis-Hastings (MH) method, to better estimate the results of the 2020 NBA Finals.

Methodology

MCMC Chains

One challenge in Bayesian inference is posterior distributions without closed-form expressions. Markov Chain Monte Carlo (MCMC) methods offer ways to sample from such posterior distributions. The MCMC is made up of two parts: Monte Carlo methods and Markov Chains. Monte Carlo methods refer to simulation methods [3]. Markov Chains implement the Markov property, which states that a conditional probability distribution of future states depends only on the present state, not any previous state. In other words, each random sample calculated is a stepping stone for the random sample succeeding it. The stone passed two steps prior has no impact on the next step you take.

A common algorithm used for this stepping process is called Metropolis-Hastings, the method used in our example. It randomly walks in the distribution space by accepting or rejecting jumps to new positions based on the following ratio, where θ is the current value and D represents the data:

$$\frac{P(\theta'/D)}{P(\theta/D)}.$$

Specifically, if this ratio exceeds 1, then we always accept θ'; otherwise we accept θ' with probability equal to this ratio [3]. In our computational example of the predicted NBA finals we use this method in order to estimate the distribution and value of the standard deviation for the points scored by and against each team in the finals.

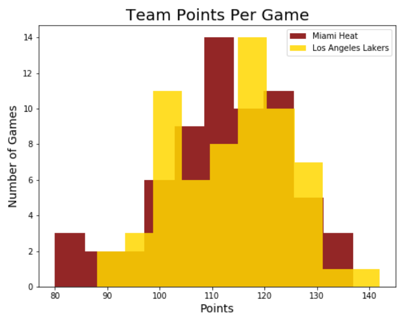

Figure 1: MIA vs. LAL (PPG)

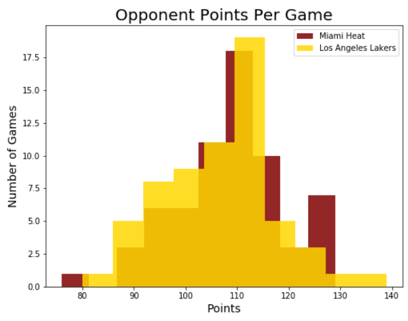

Figure 2: MIA vs. LAL (OppPPG)

Data

We obtained team schedule, opponent, team points, opponent points, and a cumulative column on wins and losses for each team from www.basketball-reference.com and information on suspended regular season games from https://en.hispanosnba.com.

Our Model

We implemented a simple simulation for the remaining regular season games as well as the first 3 rounds of the 16 team playoffs. We assumed a normal distribution for the parameters of interest, points per game (PPG) and opponent points per game (OppPPG) (reasonable assumption; see Figures 1 and 2). We estimated the means and variances with sample estimates from the observed regular season games (pre-postponement). From here, we developed a model to predict NBA game outcomes. Specifically, our model takes the average of a randomly drawn value for PPG from a normal distribution with mean and variance estimated from that team's observed PPG and a randomly drawn value for opponent's PPG from a normal distribution using mean and variance from the observed OppPPG. We would then calculate the average of these values to obtain a given team's score. This process was then repeated for the opponent, creating an estimated score for the two teams in this given game. The team with a greater point value was declared the winner. This process was repeated 10,000 times to determine each team's win probability. The team with the greater win probability "won" the match-up. Our model was inspired by Ken Jee's "How to Simulate NBA Games in Python" [2]. Our end-of-season standings for the Eastern and Western Conference are shown in Figure 3. The bolded teams are the teams who made the 16 team playoff.

Following the finalization of the end-of-season records, we then simulated the conference specific quarterfinals, semifinals and finals using the same model from before. The top seed in each conference, the Milwaukee Bucks and Los Angeles Lakers, both won their respective conferences. This finding was unsurprising to us, as both the Lakers and Bucks led their respective conferences by a multi-game margin at the point of season postponement.

To simulate the finals, we adopted a similar model as before. However, rather than use observed SD's, we used Metropolis-Hastings to estimate the SD for both teams. To estimate the value of σ, we took the average of the last 10 accepted values out of 5000 total values.

Final Season Standings: Eastern Conference and Western Conference

Seed	Name	Wins	Losses	Name	Wins	Losses
1	**Milwaukee Bucks**	**70**	**12**	**Los Angeles Lakers**	**68**	**14**
2	**Toronto Raptors**	**60**	**22**	**Los Angeles Clippers**	**61**	**21**
3	**Boston Celtics**	**58**	**24**	**Denver Nuggets**	**54**	**28**
4	**Miami Heat**	**54**	**28**	**Utah Jazz**	**54**	**28**
5	**Philadelphia 76ers**	**53**	**29**	**Dallas Mavericks**	**53**	**29**
6	**Indiana Pacers**	**49**	**33**	**Houston Rockets**	**53**	**29**
7	**Brooklyn Nets**	**40**	**42**	**Oklahoma City Thunder**	**50**	**32**
8	**Orlando Magic**	**38**	**44**	**New Orleans Pelicans**	**41**	**41**
9	Charlotte Hornets	27	55	Memphis Grizzlies	36	46
10	Washington Wizards	26	56	Portland Trail Blazers	35	47
11	Detroit Pistons	24	58	Sacramento Kings	34	48
12	Chicago Bulls	23	59	San Antonio Spurs	33	49
13	New York Knicks	23	59	Phoenix Suns	32	50
14	Atlanta Hawks	22	60	Minnesota Timberwolves	21	61
15	Cleveland Cavaliers	21	61	Golden State Warriors	15	67

Figure 3: Final standings after simulation.

Results

After using our adapted MCMC model to predict the NBA Finals, the Milwaukee Bucks bested the Los Angeles Lakers in straight games to win the series 4-0. Milwaukee won approximately 56 percent of the simulated match-ups for each game in the NBA Finals. The MH-based SD of PPG and oppPPG were slightly larger than the observed SD's (Milwaukee: 11.94 vs. 11.78 and Los Angeles: 11.67 vs. 11.53). For comparison, we also ran the old model (using observed SD's) with the Lakers and the Bucks and obtained a Milwaukee win percentage of 57 percent. These findings are likely due to the simple nature of our models, as Milwaukee ended up losing in the second round of the real 2020 playoffs to the Miami Heat, so they obviously did not win the NBA Championship.

Discussion

Our model was simple and the application of MH for estimating SD's in the finals wasn't necessary since we could have used the observed SD's. Nevertheless it served as an illustrative example of the MH algorithm. One of the ways we could improve on this model is by adding new variables such as Home/Away status or win/loss streaks. If we were to fully expand this project, we could look to incorporate multiple game predictors which may yield more accurate results than our model. In light of the year 2020 has been, we felt this project allowed us to apply a new topic (MCMC methods) to a contemporary issue in a field of our interest.

The code for the project can be found on GitHub using the link below.
https://github.com/twarnemiagh/Predicting-NBA-Games-using-Monte-Carlo-Simulations-and-MCMC-Chains

References

[1] ESPN. Nba suspends season until further notice after player tests positive for the coronavirus, Mar 2020.

[2] Ken Jee. How to simulate nba games in python, Dec 2019.

[3] Don Van Ravenzwaaij, Pete Cassey, and Scott D. Brown. A simple introduction to markov chain monte–carlo sampling. *Psychonomic Bulletin & Review*, 25(1):143–154, 2016.