Basketball Game Simulation

Lee Richardson, Roya Soleymani and Nathan Zimmerman

August 23, 2013

Abstract

When you watch a pre-game basketball show, commentators usually speak with utmost confidence about what will happen in the coming game. Phillip Tetlock [3] would certainly consider this Hedgehog behavior. In reality, while teams certainly have advantages and disadvantages prior to a game, games aren't as deterministic as the commentators make them out to be. For example, last season the World Champion Miami Heat lost back to back games against the passionate Detroit Pistons, and Milwaukee Bucks. While a two game stretch like this is certainly unlikely, it exemplifies the stochastic nature of basketball games and calls into question how much you should trust a person who guarantees results of games. Our goal is to create a basketball game simulation that is as realistic as possible using Monte Carlo methods. To make this work, we will be using actual data to represent teams and players and the probabilistic nature of a game.

1 History and Previous Work

The first traces of mathematical involvement in sports were from former North Carolina Basketball coach, Frank McGuire, who used per-possession statistics rather than per-game statistics. Analyzing basketball at the per possession level opposed to the traditional per game level has been at the cornerstone of the growing sophistication in basketball analysis. The most significant breakthrough, however, may have came from his replacement, Dean Smith. Smith is a legendary basketball coach who graduated with a degree in, you guessed it, Mathematics! [11]

At around the same time Dean Smith was coaching, Bill James was gaining popularity from his Baseball Abstracts. A lot of James' ideas had obvious carryover to basketball, and people started creating systems for objectively analyzing players. The most famous book written about basketball analysis, and one any basketball nerd will direct you to, is 'Basketball on Paper' by Dean Oliver [2]. Ironically, Oliver also had ties to North Carolina, receiving his PHD in statistics and environmental applications from UNC. Aside from a per-possession driven approach to analyzing players [1], Oliver also coined Four Factor Analysis. Four Factor analysis basically says that there are only four significant things that can

effect whether a team wins or loses (Shooting, Turnovers, Rebounding, and Free Throws). This is the cornerstone of our simulation method. We know there is a finite number of events that can happen on each possession, and there are only a finite number of possessions in an entire game. If we give each one of these events a probability, then by using Monte Carlo methods we can create a realistic simulation for a game.
In terms of full game simulations, there have been attempts made. Obviously, video games, notably the NBA 2K series, are full basketball game simulators. When I posed the question on the APBR (Association for Professional Basketball Research Metrics) message board [11], I was directed to another game simulation effort under the domain name xohoops.com, and another at http://play.basketball-gm.com/. The key difference between these websites and our simulator is that we are using real statistics to simulate the games, whereas these two used skill ratings to simulate the games. We hope that this will enhance the accuracy of our simulations.

2 Simulation Method and Assumptions

2.1 How the simulation takes place

The process in which we determine what happens in the game is through Random Number Generation. Using the Math.random() command in Java, we can generate random numbers from the Uniform Distribution:

$$\mathcal{U}(0,1)$$

throughout the game. We assign specific probabilities to events, and then use this random number to determine which event takes place.

2.2 Assumptions For our Model

When simulating a basketball game, one needs to make some simplifying assumptions about the game. The approach we took, and that a lot of others have taken in the basketball statistics community, is to view the game as a series of independent possessions. The independence issue is somewhat justified based on the "Hot Hand Fallacy" [4] where they address the misconception of random sequences, specifically related to shooting. In terms of actual results of a possession, the independence assumption comes under more scrutiny, which is exemplified by Carlin and Skinner's "Price of Anarchy" [6]recently published in Significance Magazine. The relation to basketball here is that the more a team runs a specific play, the less efficient the play will become. This is why Lebron James can't simply drive in for a lay-up every time. While this may be the most efficient play, the defense will adjust if it becomes overly predictable. So while possessions aren't a truly independent process, it is an assumption we have to

make in our model.

The next assumption comes at the actual possession level. While there are certainly a lot of crazy things that can happen on each possession, we limit the potential events to:

1. Offensive Team Turns the Ball Over
2. Offensive Team Shoots a Two Point Shot
3. Offensive Team Shoots a Three Point Shot
4. Offensive Team Shoots Two Free Throws

Of course, there is the possibility of a possession continuing after one of these events take place, namely when the offensive team misses the shot, but also secures the offensive rebound. When this happens, these four events all have a possibility of happening again. There certainly is an argument to be made that the probabilities of certain events should be different once an offensive rebound has occurred. For example, the offensive rebound will typically get the ball closer to the rim, and thus be likely to attempt a 2 point "put-back" shot, in which the odds of making it are quite higher. For the purposes of the simulation, however, we will assume that after a team gets an offensive rebound, the corresponding probabilities of events on that possession will be the same as they were at the beginning of the possession.

Another assumption we make is that the number of possessions for each team will be equal in each game. This is generally true in the game of basketball, as teams alternate possessions throughout the game. There are some ways, such as technical and flagrant fouls, where a team can randomly gain an extra possession (free throws) throughout the course of the game. Another way this assumption can be off in real basketball is if the team gets the last possession of every quarter. Since each team starts off two of four quarters with the ball, this would give said team an extra two possessions in the game. We ignore these subtleties in our simulation, the former for the question of how to insert a rare event like a technical foul into a simulation and the latter because of the hierarchical structure we chose to simulate our game with, which will be explained in the Methodology Section.

A couple of other assumptions we make for simplicity are:

- There is no overtime. If a team ends in a tie, we will leave it at that.
- There are no "And-1" plays, where the offensive team makes a shot, and gets fouled simultaneously

3 Methodology

3.1 General Structure of the Simulation

The structure of our game is tree-like, we start by assigning each team specific input data, and use this to determine the length of the game in terms of the number of independent possessions (equal for each team, see assumptions section). Then, we simulate that number of possessions for Team A, then Team B, while recording the score of each individual possession. Once we have done this, we compare the scores for each team to determine the winner of the game. To determine the length of the game, we used data from TeamRankings.com [10] to find the average number of possessions for each team in an NBA game. We were not however able to find the variance for each team in terms of number of possessions per game. To make our game more realistic, we assumed that the length of a game followed a Normal Distribution, with a mean that is the average of the two teams' possessions per game, and a standard deviation of four. We chose four randomly, and this is something we should look into further.

3.2 Options for the Possession

Based on this structure, the possession is our most important algorithm. The possession starts by generating a random number from the uniform distribution mentioned earlier, and the events of the possession will go from there. Note that this is not the only time this random number generator is used during the possession. When another uncertain event, such as a rebound, is going to take place, we use a random number to determine which team gains possession of that rebound after a missed shot. If we used the same RV for this that we started the possession with, these two events wouldn't be independent. The next decision was to decide how to use this initial random number to determine the outcome of a possession. We used two different methods to determine the outcome of individual possessions, one by only considering team based statistics, and the other by forming teams made up of players, and using both player and team statistics to assign results.

For a Team focused simulation, we assign each team the following input data:

1. Average shooting percentage on free throws, two point shots, and three point shots
2. Team's offensive distribution in terms of events. i.e: The percent chance the team will shoot free throws, two point shots, three point shots, or make a turnover
3. The team's offensive and defensive rebounding percentages

With these data, we assign the original random number to an event based on the team's offensive distribution, use the team's shooting percentages to determine the result of the shot (if the possession didn't end in a turnover), and use

	Heat	Lakers
Points	109.23	100.36
Standard Deviation	11.79	11.70
Wins out of 1000	707	293

the offensive rebound and defensive rebounding percentages of both teams to determine the result of a rebound if a shot was missed. If an event corresponding to a possession ending takes place (defensive rebound, made shot, turnover), we return the point total associated with this event (in our case, this is either 0, 1, 2, or 3). The possession will keep looping through events until a possession-ending event takes place. Theoretically, a possession could last forever if a team keeps missing and getting the offensive rebound. This is also possible in real basketball. However, there has never been an infinite possession in real basketball, due to the time limit. The number of shots per possession could probably be modeled with a Poisson Distribution, in which it is theoretically possible to have an enormous number, but the odds make it extremely unlikely.

When we base our game on player centered data, the method is essentially the same in terms of the event distribution, except with one more key input, Player Usage Rate [5]. Usage rate corresponds to the percentage of possessions that a player utilizes while on the court. The statistical calculation of usage rate is a combination of players' free throws, field goals, and turnovers, relative to the teams total. If these usage rates add up to more than 100 percent, we can adjust them so they add to only 100 percent, by taking each player's individual usage rate and dividing by the sum of the usage rates for each of the other 5 players on the floor. This way, our random number generation at the beginning still makes sense. Using these adjusted usage rates, we can assign the possession to one of the five players on the court with our original random variable. In this case, each player needs an offensive distribution and shooting percentage assigned to him, which are the same structure as the ones used in the team only simulation. We do the same thing here with team offensive and defensive rebounding percentages if the selected player misses the shot as was done in the team simulation.

These are the two approaches we used to simulate basketball games. Due to the additional complexity of the second method, we used this one for our simulations. One thing we could have tried is to simulate a game 1000 times using each method, and compare the results to see if either reduced the uncertainty or changed the winners and losers drastically.

4 Results

To test our simulator, we used input data from the 2012-2013 season for the Miami Heat and the Los Angeles Lakers. The two teams actually played twice

this year, and the Heat won both games: 99-90 and 107-97. We chose the 5 players from each team who started the most games, and obtained input data from Hoopdata [9], NBA.com stats [8], and teamrankings.com [10].
The results of our simulations are displayed in the table above. Both teams had practically identical variances, and the Heat averaged about 9 more points per game. The Heat came out victorious in about 70 percent of the games. This seems pretty reasonable, given that the Heat won both games in the regular season. Also, the Heat won about 80 percent of their games in the actual 2012-13 regular season. Given the Zero-Sum nature of a basketball game, their average opponent would theoretically have a winning percentage of around 50 percent. Because the Lakers had a winning percentage that was higher than 50 percent, it would make sense that while the Heat would still win the majority of the games, it would be less that 80 percent if they had to play a team who had a winning percentage higher than 50 percent.
When we recorded the individual players scoring totals throughout the games, they did a good job reflecting the Usage Rates as input data. Kobe Bryant had the highest usage rate out of any player in our simulations, and he usually took around 26 shots. It would be good to record the total number of points per player, and analyze these results, and that could give us further information as how to judge single player performances in a game. Simulating games as realistically as possible gives us a good way of quantifying the uncertainty inherent in a game. We could use this to generate distributions for number of points scored by a team, player, and more.
While our simulation did a reasonable job representing an NBA game, there were things we could have added to make it more realistic. They are discussed below.

5 Future Additions and Analysis

One of the main issues with this simulation is that it only compares teams' offenses with one another. The only defensive statistic that is inputted is defensive rebounding. While this is an important part of defense, there are certainly other things that defenses do to effect the outcome of a game. Because of this, a team with a great defense and mediocre offense would appear to be a mid level team, rather than above average. Some ways to address this is by recording where each shot is being taken, and then figuring out which defensive player would be most likely to defend that player, and use that defensive player's history to adjust the probability of making a shot. While that is theoretically possible, it was beyond the scope of this project.
Another issue we did not have time to address was that in our simulation, the same 5 players played the entire game. Because this isn't actually what happens in basketball, it skews the results and favors teams that are more top heavy, and doesn't reward teams that have a lot of depth. One way we thought to address this was to use player's minutes per game statistics to assign the player a probability of being involved on every possession. i.e: if Lebron averages 36

minutes per game out of 48 possible minutes, then each possession he has a 75 percent chance of being on the court. We could use these probabilities before every possession to determine the 5 players involved, and then using those players, simulate the possession as we normally do. The process would have to be done in the same way the NBA lottery chooses the draft order, and we couldn't quite implement this for our project.

Another part of the game that isn't realistic is the time factor. We didn't use time whatsoever in our game. If we were to use time, then we would have to incorporate the length of each possession, and each possession we could reduce the amount of time left in the given quarter. This would allow us to relax our assumption that each team has equal possessions. We would need data that corresponds to the average length of each possession for each team, as well as the variance of length of possessions. This was beyond the scope of our model, but could be incorporated in the future.

There are lots of other intricacies in a basketball game that we weren't able to account for with our simulator. The assumptions we made above could all be addressed, and we could make the game more and more complex. Figuring out a way to interact better with databases would have made our results more interesting, as then we could have simulated all sorts of different games and player combinations, but we will have to hope to do more of this in the future. The list of things you could add and try to account for goes on and on, and the more time we spent discussing what we could do with the project, more things came up that we could have added.

References

[1] Justin Kubatko, Dean Oliver, Kevin Pelton, and Dan T. Rosenbaum. "A Starting Point for Analyzing Basketball Statistics." Journal of Quantitative Analysis in Sports 3.3 (2007): n. pag. Print.

[2] Oliver, Dean. Basketball on Paper: Rules and Tools for Performance Analysis. Washington, D.C.: Brassey's, 2004. Print.

[3] "Expert Political Judgment: How Good Is It? How Can We Know? [Hardcover]." Amazon.com: Expert Political Judgment: How Good Is It? How Can We Know? (9780691123028): Philip E. Tetlock: Books. N.p., n.d. Web. 17 Aug. 2013.

[4] Gilovich, T. "The Hot Hand in Basketball: On the Misperception of Random Sequences*1." Cognitive Psychology 17.3 (1985): 295-314. Print.

[5] "An Advanced Stats Primer for the NBA - Golden State Of Mind." Golden State Of Mind. N.p., n.d. Web. 17 Aug. 2013.

[6] Skinner, Brian, and Brad Carlin. "The Price of Anarchy- on the Roads and in Football." Significance n.d.: n. pag. Print.

[7] Haley, Jeffrey. "An Introduction to Advanced Basketball Statistics: Understanding Possession Estimation and the Factors That Control Efficiency - Burnt Orange Nation." Burnt Orange Nation. N.p., n.d. Web. 17 Aug. 2013.

[8] Nba.com/Stats. NBA, n.d. Web. 18 Aug. 2013.

[9] Http://hoopdata.com/advancedstats.aspx. Hoopdata, n.d. Web. 18 Aug. 2013.

[10] "NBA Team Possessions per Game." NBA Stats. N.p., n.d. Web. 18 Aug. 2013.

[11] "APBR Metrics Forum." Rev. of Basketball Simulation Ideas. n.d.: n. pag. Web.

6 Code

```
\begin{tiny}
import java.util.*;
import org.uncommons.maths.random.*;

public class BasketballPlayerSim {

public static void main(String[] args) {
int teamOneScore = 0;
int teamTwoScore = 0;

// team one's five starting players
// LAKERS
Player oneTeam1 = new Player(.51,.324,.839,.317,.551,.189,.133,.128, "Kobe Bryant");
Player twoTeam1 = new Player(.519,.438,.922,.177,.529,.199,.193,.078, "Steve Nash");
Player threeTeam1 = new Player(.475,.286,.702,.205,.724,.037,.137,.103, "Pau Gasol");
Player fourTeam1 = new Player(.464,.342,.734,.176,.414,.414,.098,.075, "Metta World Peace");
Player fiveTeam1 = new Player(.581,.167,.492,.224,.595,.004,.166,.234, "Dwight Howard");

Team3 team1 = new Team3("Lakers", oneTeam1, twoTeam1, threeTeam1, fourTeam1, fiveTeam1,
97.8, .222, .73);

// team two's five starting players
// HEAT
Player oneTeam2 = new Player(.602, .406, .753, .298,.606,.14,.124,.13, "Lebron James");
Player twoTeam2 = new Player(.537,.258,.725,.291,.697,.044,.131,.127, "Dwayne Wade");
Player threeTeam2 = new Player(.557,.284,.798,.228,.713,.063,.11,.114, "Chris Bosh");
Player fourTeam2 = new Player(.456,.409,.795,.16,.327,.432,.171,.071, "Mario Chalmers");
Player fiveTeam2 = new Player(.512,0,.711,.107,.795,0,.143,.063, "Udonis Haslem");
```

```
Team3 team2 = new Team3("Heat", oneTeam2, twoTeam2, threeTeam2, fourTeam2, fiveTeam2,
94.0, .2697, .7426);

// generates a possession count that will be the same for each team based on a normal
distribution
// with the mean as the average of both teams statistical average possessions per game
from the
// 2012-2013 season and with a standard deviation of 4.
Random r = new Random();
GaussianGenerator n = new GaussianGenerator((team1.avgPoss + team2.avgPoss)/2, 4.0, r);
int numPoss = (int) Math.round(n.nextValue());

double team1Rebound = (team2.defensiveReb + (1-team1.offensiveReb))/2;
for(int i = 0; i < numPoss; i++) {
teamOneScore += possession(team1, team1Rebound, 0);
}

double team2Rebound = (team1.defensiveReb + (1-team2.offensiveReb))/2;
for(int i = 0; i < numPoss; i++) {
teamTwoScore += possession(team2, team2Rebound, 0);
}

System.out.println(team1.teamName + " : " + teamOneScore);
System.out.println(team2.teamName + "   : " + teamTwoScore);
System.out.println();

if(teamOneScore > teamTwoScore) {
System.out.println(team1.teamName + " win!");
} else {
System.out.println(team2.teamName + " win!");
}
System.out.println();

// Lakers stats
System.out.println(team1.teamName + " Statistics:");
System.out.printf("%-17s|%8s%8s%8s%8s\n", "PLAYERS", "PTS", "FG", "3P", "FT");

System.out.printf("%-17s|%8d%8s%8s%8s\n", team1.pOne.playerName, team1.pOne.points,
team1.pOne.FGMade + "/" + team1.pOne.FGAttempts,
team1.pOne.threeMade + "/" + team1.pOne.threeAttempts, team1.pOne.freeThrowMade + "/" +
team1.pOne.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team1.pTwo.playerName, team1.pTwo.points,
team1.pTwo.FGMade + "/" + team1.pTwo.FGAttempts,  team1.pTwo.threeMade + "/" +
team1.pTwo.threeAttempts, team1.pTwo.freeThrowMade + "/" +  team1.pTwo.freeThrowAtt);
```

```
System.out.printf("%-17s|%8d%8s%8s%8s\n", team1.pThree.playerName, team1.pThree.points,
team1.pThree.FGMade + "/" + team1.pThree.FGAttempts,
team1.pThree.threeMade + "/" + team1.pThree.threeAttempts,
team1.pThree.freeThrowMade + "/" + team1.pThree.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team1.pFour.playerName, team1.pFour.points,
team1.pFour.FGMade + "/" + team1.pFour.FGAttempts,
team1.pFour.threeMade + "/" + team1.pFour.threeAttempts,
team1.pFour.freeThrowMade + "/" + team1.pFour.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n\n", team1.pFive.playerName, team1.pFive.points,
team1.pFive.FGMade + "/" + team1.pFive.FGAttempts,
team1.pFive.threeMade + "/" + team1.pFive.threeAttempts,
team1.pFive.freeThrowMade + "/" + team1.pFive.freeThrowAtt);

// Heat stats
System.out.println(team2.teamName + " Statistics:");
System.out.printf("%-17s|%8s%8s%8s%8s\n", "PLAYERS", "PTS", "FG", "3P", "FT");

System.out.printf("%-17s|%8d%8s%8s%8s\n", team2.pOne.playerName, team2.pOne.points,
  team2.pOne.FGMade + "/" + team2.pOne.FGAttempts,
team2.pOne.threeMade + "/" + team2.pOne.threeAttempts,
team2.pOne.freeThrowMade + "/" + team2.pOne.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team2.pTwo.playerName, team2.pTwo.points,
team2.pTwo.FGMade + "/" + team2.pTwo.FGAttempts,
team2.pTwo.threeMade + "/" + team2.pTwo.threeAttempts,
team2.pTwo.freeThrowMade + "/" + team2.pTwo.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team2.pThree.playerName, team2.pThree.points,
team2.pThree.FGMade + "/" + team2.pThree.FGAttempts,
team2.pThree.threeMade + "/" + team2.pThree.threeAttempts,
team2.pThree.freeThrowMade + "/" + team2.pThree.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team2.pFour.playerName, team2.pFour.points,
team2.pFour.FGMade + "/" + team2.pFour.FGAttempts,
team2.pFour.threeMade + "/" + team2.pFour.threeAttempts,
team2.pFour.freeThrowMade + "/" + team2.pFour.freeThrowAtt);

System.out.printf("%-17s|%8d%8s%8s%8s\n", team2.pFive.playerName, team2.pFive.points,
team2.pFive.FGMade + "/" + team2.pFive.FGAttempts,
team2.pFive.threeMade + "/" + team2.pFive.threeAttempts,
team2.pFive.freeThrowMade + "/" + team2.pFive.freeThrowAtt);

}
```

```
public static int possession (Team3 teamOffense, double rebound, int score) {
double randPoss = Math.random();
if (randPoss <= teamOffense.one) {
// Player One
score = possessionHelp(teamOffense.pOne, teamOffense, rebound, score);
} else if (randPoss > teamOffense.one && randPoss <= teamOffense.two) {
// Player Two
score = possessionHelp(teamOffense.pTwo, teamOffense, rebound, score);
} else if (randPoss > teamOffense.two && randPoss <= teamOffense.three) {
// Player Three
score = possessionHelp(teamOffense.pThree, teamOffense, rebound, score);
} else if(randPoss > teamOffense.three && randPoss <= teamOffense.four) {
// Player Four
score = possessionHelp(teamOffense.pFour, teamOffense, rebound, score);
} else {
// Player Five
score = possessionHelp(teamOffense.pFive, teamOffense, rebound, score);
}
return score;
}

public static int possessionHelp (Player p, Team3 teamOffense, double rebound, int score) {
double randPoss = Math.random();
if (randPoss > p.turnovers && randPoss <= p.splitOne) {
// 2pt shot
return shot(p, teamOffense, rebound, score, 2);
} else if (randPoss > p.splitOne && randPoss <= p.splitTwo){
// 3pt shot
return shot(p, teamOffense, rebound, score, 3);
} else if (randPoss > p.splitTwo) {
// Free Throws
p.freeThrowAtt += 2;
for (int i = 0; i < 2; i++) {
double r = Math.random();
if (i == 1 && r > p.freeThrowPerc) {
offenseRebound(teamOffense, rebound, score);
} else if(r <= p.freeThrowPerc) {
p.points++;
p.freeThrowMade++;
score++;
}
}
return score;
}
// Turnover
return 0;
```

```
}

public static int shot (Player p, Team3 teamOffense, double rebound, int score, int n) {
double r = Math.random();
p.FGAttempts++;
if (n == 3) {
p.threeAttempts++;
if (r <= p.threePointPerc) {
p.points += 3;
p.threeMade++;
p.FGMade++;
return score + 3;
}
} else if (n == 2 && r<= p.twoPointPerc) {
p.points += 2;
p.FGMade++;
return score + 2;
} else {
return offenseRebound(teamOffense, rebound, score);
}
return score;
}

public static int offenseRebound (Team3 teamOffense, double rebound, int score) {
double randReb = Math.random();
if(randReb > rebound) {
return possession(teamOffense, rebound, score);
}
return score;
}
}

\end{tiny}
```