NBA game winning strategy

Information, networks, and markets (5FY126)

Supervisors: Martin Rosvall, Christopher Blocker, Anton Eriksson

Abstract

This report analyzes if game theory can be applied to shooting element in a game of basketball and improve the results for a team using it. To find out if this is true, a program to simulate shots in a basketball game was created using Python. Statistics from a previous season were the parameters for the simulations. Our simulations showed that, when using game theory optimal shooting a statistically worse team could beat a better team which was not using optimal shooting. Based on the results from the simulations, game theory could be applied in a real game of basketball.

September 4, 2021
Umeå Universitet
Department of Physics

Filip Forsgren (`fifo0010@student.umu.se`)
Jakob Storskrubb (`jast0050@student.umu.se`)
Marcus Öström (`maos0049@student.umu.se`)

Contents

1 Introduction

The idea of this study was born while watching the then ongoing NBA finals between Miami Heat and Los Angeles Lakers. The question we asked ourselves were, is there a way we can apply Game Theory (GT) and come up with a better strategy for the selection of shots to shoot? After doing a lot of assumptions we came up with a model which was able to run simulations of a number of games where the teams took turn shooting according to what GT says is the best strategy. We then compared what the outcome would be if the teams were to shoot according to the optimal strategy, derived by GT versus the outcome if the team were to shoot according to how they actually shot through the playoffs of the 2019-2020 season (s19-20).

So why does this matter? Not only is it fun to apply our knowledge to basketball, which is a fun game to watch. National Basket Association (NBA) is also the biggest basket league in the world, where the average value of the NBA franchises increased to $2,12 billion (Badenhausen, 2020). Every win a team can get imply a big amount of money and an optimal game strategy is a key to success, both on the court and on the bank account. If we apply the game into a game theory model, a strategy can then be developed and hopefully it will result in an optimal game winning strategy.

2 Problem Description

The goal with this project is to find a optimal strategy for the attacking team, a strategy that possible will increase the amount of points scored in a basketball game to in turn increase the number games and series won (In NBA Playoffs the teams play 7 games series where the team with the most least wins is eliminated).

2.1 The Game Of Basketball

Basketball is played with five players on each team, the goal is to score by throwing a ball in the other teams hoop as well as defending your own hoop. If a team scores, two or three points are awarded depending on where the shot came from. If the shot was fired outside of the three point line the team is awarded three point, and two points if the shot came from inside the three point line. Each match in the NBA consists of four quarters, each 12 minutes. The team that has scored most points at the end wins the game. Assumptions to simplify the game will be presented in the method section.

2.2 Restrictions

In this project we have chosen to focus on only analyzing statistics from the Playoffs of the 19-20's season. And the two teams we will be analyzing are this years finalists, Los Angeles Lakers and Miami Heat. We are looking for an optimal strategy for the offensive play, where the strategy depends on the combination of shooting three pointers versus two pointers. The optimal strategy in practice depends on a big number of aspects. To make it possible to create a model where we can apply game theory, we have done a lot of assumptions to simplify the game(A list of assumptions can be found under 4.3 Assumptions). NBA.com has is resource with statistics from all teams over many seasons, we have to restrict the analysis to only two teams over one season. Due to the projects time restriction we had to stop at two teams, but the principle on how we integrate GT can be apply in a more extensive model.

2.3 Model

We agreed that the best way to accomplish the goal we set up in a way that create most realistic results is by creating a model in python. The model reads data available on NBA.com and uses this data to simulate a basketball game with a rebounds, turnovers, shot distribution between players, and of course based on the data. Other aspects of the game is simulated by using trivial randomness. And the main aspect, the shot selection is simulated using the data and is constructed to either mimic the distribution between 2- and 3 point shots that each player actually shot through the NBA Playoffs s19-29 or it is constructed by deriving the optimal strategy according to GT for each player.

3 Theory

3.1 Best response and dominant strategy

A best response to a strategy used by the opponent is when your response gives at least as good payoff as any other strategy you could have chosen. We can clarify this with notations, if we have player 1 (P1) with strategy A and Player 2 (P2) with strategy B. Then we have a payoff matrix which can be classified as a pair of strategies (A,B). The payoffs for P1 and P2 can be denoted as:

$$P_1(A,B)$$
$$P_2(A,B)$$

Further, A will now denote a *best response* to a strategy B, if A produces at least as good payoff as any other strategy paired with B, this can be classified as:

$$P_1(S,T) \geq P_1(S',T)$$

In a game with two players, P1 and P2. A dominant strategy for P1 is a strategy that is best response to any of P2‘s strategies (Easley, Kleinberg, 2010, 163-164).

3.2 Nash equilibrium

The definition of Nash equilibrium was presented by John Nash in 1950. A Nash equilibrium occurs in a game with two or more players when none of the players has anything to gain by changing strategy. Suppose player 1 (P1) and player 2 (P2) are playing a game, P1 plays X and P2 plays Y. Then (X,Y) is a Nash equilibrium if X is the optimal strategy against Y and Y is the optimal strategy against X. This holds even if when there are no dominant strategy (Easley, Kleinberg, 2010, 167-168).

3.3 Mixed strategies

If no Nash equilibrium exists in a game, an idea is to play a mix of strategies to throw the opponent off. The mix can be randomized as such, Player 1 plays strategy X with probability p and Y with probability $1 - p$. Player 2 plays strategy X with probability q and Y with probability $1 - q$. This will give payoffs that depends with the probabilities.

Suppose we have the following game with corresponding payoff matrix:

		Player 2	
		X	Y
Player 1	X	$(+2, -2)$	$(-2, +2)$
	Y	$(-2, +2)$	$(+2, -2)$

Table 1: Payoff matrix, payoffs for player 1 and player 2 with different pairs of strategies.

If Player 1 plays strategy X the expected payoff will be:

$$2q - 2 * (1 - q)$$

If Player 1 plays strategy Y the expected payoff will be:

$$2 * (1 - q) - 2q$$

An equilibrium for mixed strategy occurs when a pair of probabilities that is best response to each other is found.

4 Methodology

4.1 Data Acquisition

The data was extracted from the official NBA website NBA.com which has a great database of statistics from players and teams. The data is extracted to Excel documents, one for the player stats (see appendix 9.1) and one for the team stats (see appendix 9.2).

4.2 Python code

In Python we built a model that simulates a set number of basketball games. The first step is to read in data from Excel. This data is then processed to the point where it can be used in the model, for the most part getting distributions and percentages for a specific element of the game for a player or team. The processed data is then stored in arrays so they can be called upon when needed in the simulation. For a detailed look how this simulation works and uses the data to simulate a game until the time for a shot and after (see appendix 9.1 - 9.3). The shot which is the interesting part is simulated using one of two Python functions Shooting_opt or Shooting_Real both whom takes a player index as an parameter and further works accordingly (to save time when reading, both functions are identical with the exception of step 1):

Shooting_Opt:

1. solving for optimal strategy.

   given the following Payoff-matrix:

Pa_{3D}	Pa_{2O}
Pa_{3O}	Pa_{2D}

   where Pa_{ij} = (Given by data: the shooting players probability of a successful shot with strategy $i + j$) $\times$ -*i*

   ,

   and i represents the strategies 2- and 3 point shot and j represents the defenders impact which is -0.1 *if defending the right shot* (D) *and* 0.1 *if giving an open shot*(O).

   If we let q be the probability that the shooter chooses strategy 3 point shot. We have that the expected payoff for shooting a 3 point shot is $Pa_{3D} * q + Pa_{3O} * (1 - q)$ and the expected payoff for a 2 point shot is $Pa_{2O} * q + Pa_{2D} * (1 - q)$. The optimal strategy for the offensive player is that the defending player has the same payoff no matter their choice of strategy.

   $$Pa_{3D} * q + Pa_{3O} * (1 - q) - (Pa_{2O} * q + Pa_{2D} * (1 - q)) = 0 \tag{1}$$

   So given by GT, by solving equation 1 for q we get the optimal distribution for the strategies, where the offensive player should shoot 3 point shots with probability q and 2 point shot with probability $(1 - q)$.

   (step 1. is performed outside the function and q's are stored in an array and are called upon inside the function. This is to optimizing the run time of the simulation.)

2. The Shooter is assumed to play the optimal strategy and therefor shoot according to step 1.

   - Generate a standard uniformly distributed random variable x.
   - If $x < q_p$ the shooter chooses to shoot the 3 point shot and if $x \geq q_p$ the shooter chooses to shoot the 2 point shot. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)

3. The defender is assumed to know/learn how the offensive player distributes his shot and defend the 2- and 3 point shot with the corresponding probability. To simulate what the defending player chooses to defend we:
    - Generate a standard uniformly distributed random variable y.
    - If $y < q_p$ the defender chooses to defend the 3 point shot and if $y \geq q_p$ the defender chooses to defend the 2 point shot. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)
4. Given the following matrix for the probability of the player making the shot:

P_{3D}	P_{2O}
P_{3O}	P_{2D}

where P_{ij} = Given by data: the shooting players probability of a successful shot with strategy $i + j$, *and i represents the strategies 2- and 3 point shot and j represents the defenders impact which is* -0.1 *if defending the right shot* (D) *and* 0.1 *if giving an open shot*(O).

- Generate a standard uniformly distributed random variable z.
- If $z < q_p$ the shot goes in and the offensive team is awarded the number of points according to what shot they took and if $z \geq q_p$ the team is awarded zero points. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)

Shooting_Real:

1. Real life mimicking (RLM) of distribution of shots taken. In the data gathered from NBA.com we can see how many shots each player attempted of each strategy, 2- and 3 point shots in the playoffs s19-20.

$$q = \frac{3\,point\ attempts}{3\,point\ attempts + 2\,point\ attempts}$$

to mimic the real distribution of 2 and 3 point shots the offensive player should shoot 3 point shots with probability q and 2 point shot with probability $(1 - q)$.

2. The Shooter is assumed to play according to the RLM distribution and therefor shoot according to step 1.
    - Generate a standard uniformly distributed random variable x.
    - If $x < q_p$ the shooter chooses to shoot the 3 point shot and if $x \geq q_p$ the shooter chooses to shoot the 2 point shot. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)
3. The defender is assumed to know/learn how the offensive player distributes his shot and defend the 2- and 3 point shot with the corresponding probability. To simulate what the defending player chooses to defend we:
    - Generate a standard uniformly distributed random variable y.
    - If $y < q_p$ the defender chooses to defend the 3 point shot and if $y \geq q_p$ the defender chooses to defend the 2 point shot. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)
4. Given the following matrix for the probability of the player making the shot:

P_{3D}	P_{2O}
P_{3O}	P_{2D}

where P_{ij} = Given by data: the shooting players probability of a successful shot with strategy $i + j$, *and i represents the strategies 2- and 3 point shot and j represents the defenders impact which is* -0.1 *if defending the right shot* (D) *and* 0.1 *if giving an open shot*(O).

- Generate a standard uniformly distributed random variable z.
- If $z < q_p$ the shot goes in and the offensive team is awarded the number of points according to what shot they took and if $z \geq q_p$ the team is awarded zero points. (where q_p is the probability that the player p shoots a 3 point shot as given from step 1.)

After the shot outcome has been simulated the rest of the simulations continuous until the game is finished. To get a result that is not too volatile we iterated the game-simulation 10 000 times for each combination of shot strategy, that is:

- Heat using GT optimal strategy, Los Angeles Lakers using RLM strategy.
- Heat using RLM strategy, Los Angeles Lakers using GT optimal strategy.
- Both teams using GT optimal strategy.
- Both teams using RLM strategy.

From the 40 000 iterations we gathered information on how many 7 game series won, games won and point made per game by each team. The outcome is presented under Results (Section 5).

4.3 Assumption

To make this project manageable we had to do some assumptions, here is a list of some key assumptions:

- The defender is assumed to know/learn how the offensive player distributes his shot and defend the 2- and 3 point shot with the corresponding probability.
- The game only consist of three pointers and two pointers, so free throws, penalties and fouls are excluded.
- The game lasts for 48 minutes (2880) seconds, no quarter breaks, halftime or timeouts are included in the simulation.
- In case of a tie the game will allow 5 more minutes of play until there is a winner.
- The time for possession will be a uniformly randomized number between 0 and 24 seconds.
- No substitutes, the five players in the starting line will play through the whole game series.
- The defender affects the shooter with -0.1 probability of the original probability to make the shot if defending the correct shot. If defending the wrong shot the defender affects the shooter with +0.1 of the original probability. The original probability is extracted from the data.

5 Results

	Miami Heat	Los Angeles Lakers
Games won[%]	47.54	52.46
Series won[%]	43.63	56.37
Points scored	99.4	100.3

Table 2: Miami Heat Playing GTO, 10 000 simulations.

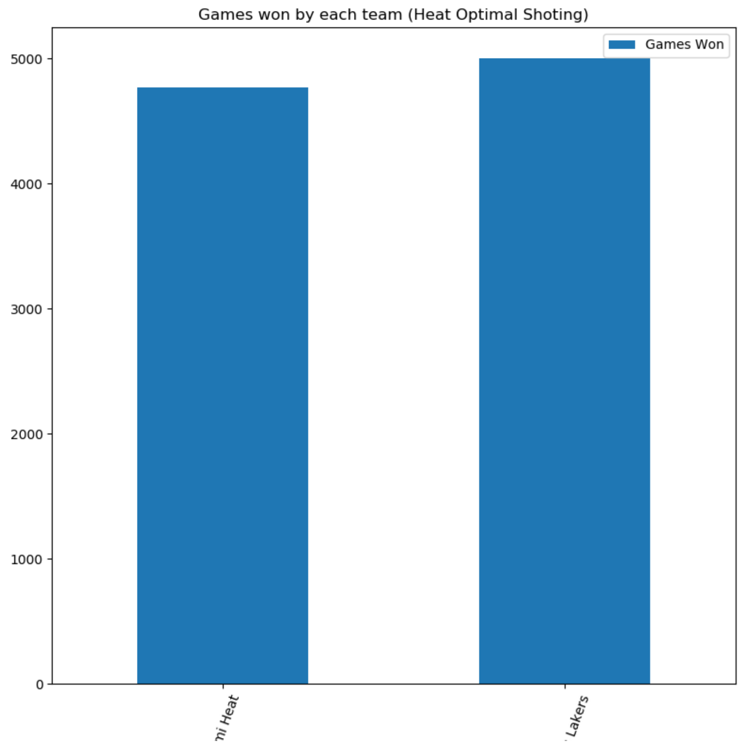


Figure 1: *Games won per team, Miami Heat Playing GTO strategy, Lakers playing RLM strategy*

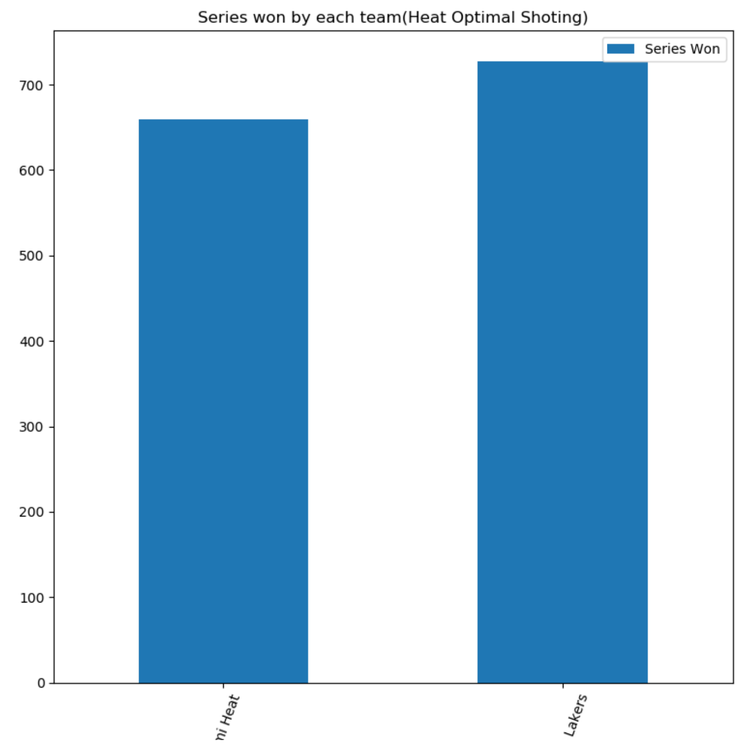


Figure 2: *7 game series won per team, Miami Heat Playing GTO strategy, Lakers playing RLM strategy*

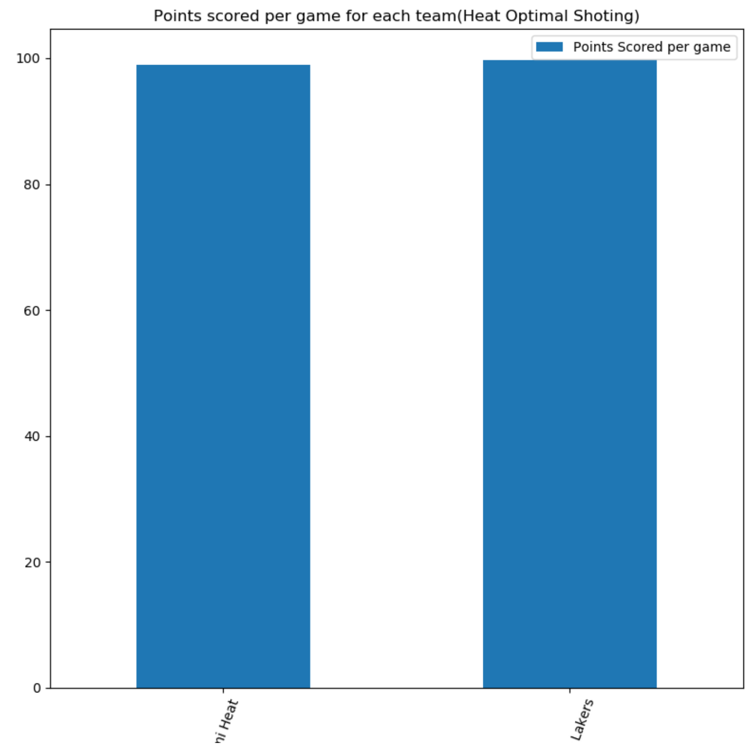


Figure 3: *Average points per game per team, Miami Heat Playing GTO strategy, Lakers playing RLM strategy*

	Miami Heat	Los angeles Lakers
Games won[%]	30.45	69.55
Series won[%]	8.30	91.70
Points scored	95.8	104.6

Table 3: Lakers Playing GTO, 10 000 simulations.

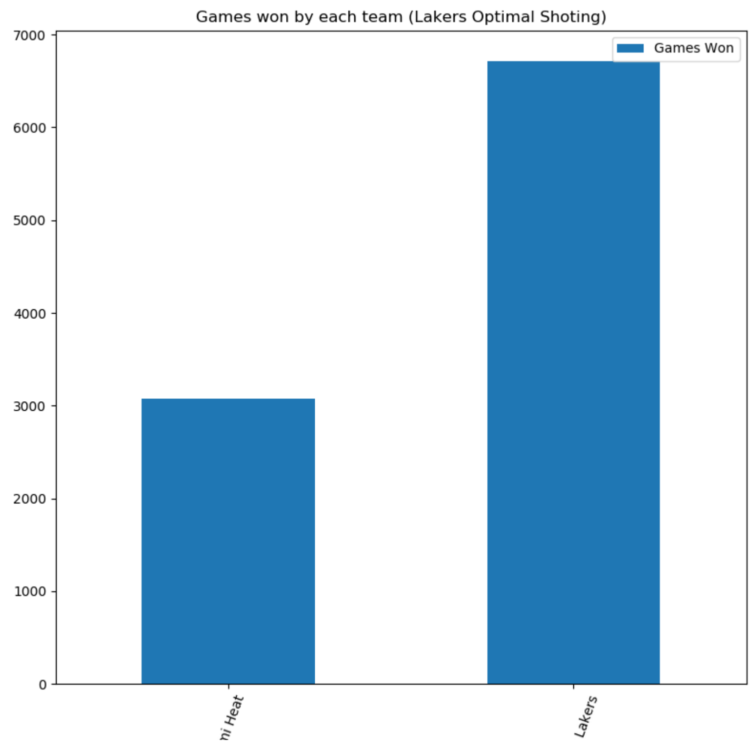


Figure 4: *Games won per team, Miami Heat Playing RLM strategy, Lakers playing GTO strategy*

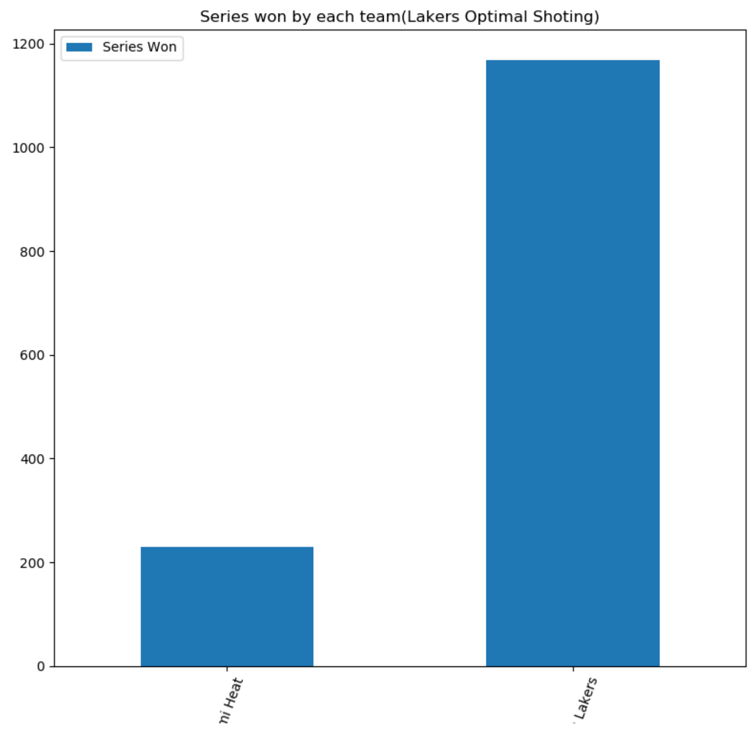


Figure 5: *7 games series won per team, Miami Heat Playing RLM strategy, Lakers playing GTO strategy*

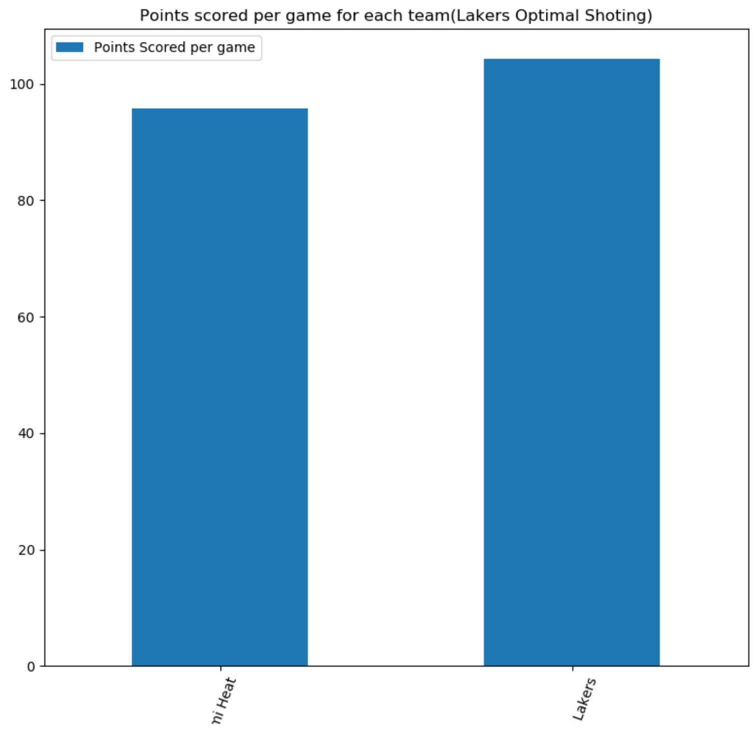


Figure 6: *Average points per game per team, Miami Heat Playing RLM strategy, Lakers playing GTO strategy*

	Miami Heat	Los angeles Lakers
Games won[%]	39.81	60.19
Series won[%]	28.15	71.85
Points scored	98.9	103.9

Table 4: Both teams playing GTO, 10 000 simulations.

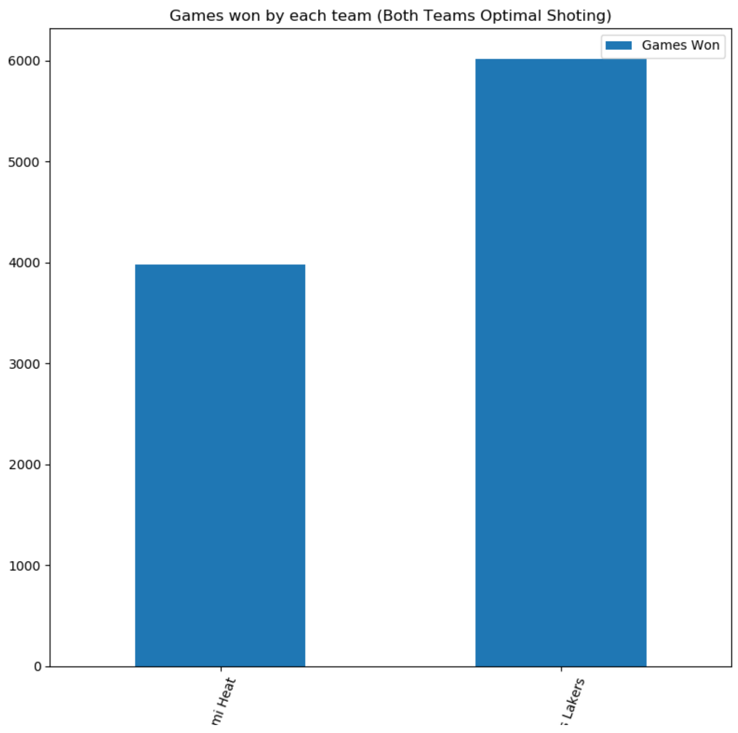


Figure 7: *Games won per team, Miami Heat Playing GTO strategy, Lakers playing GTO strategy*

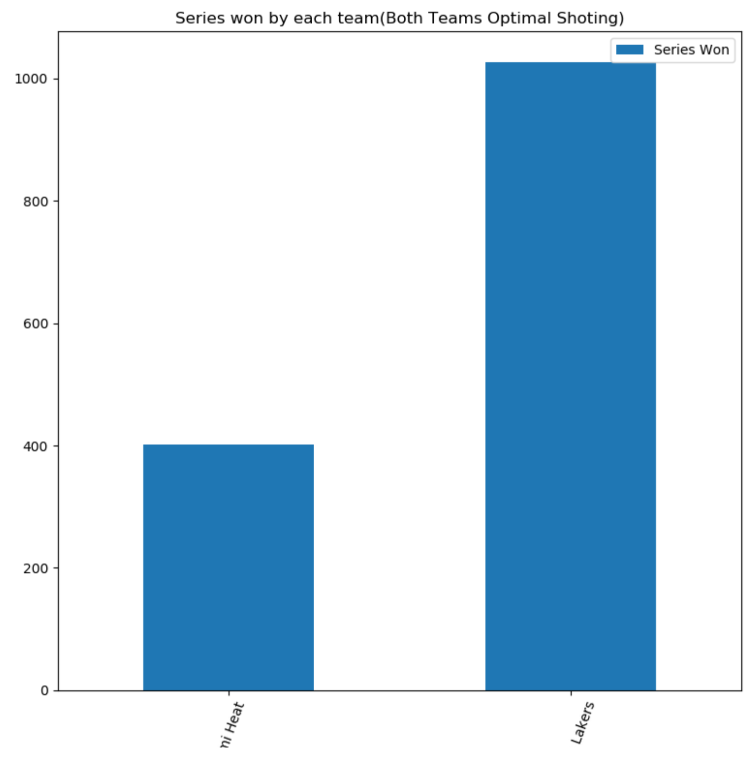


Figure 8: *7 games series won per team, Miami Heat Playing GTO strategy, Lakers playing GTO strategy*

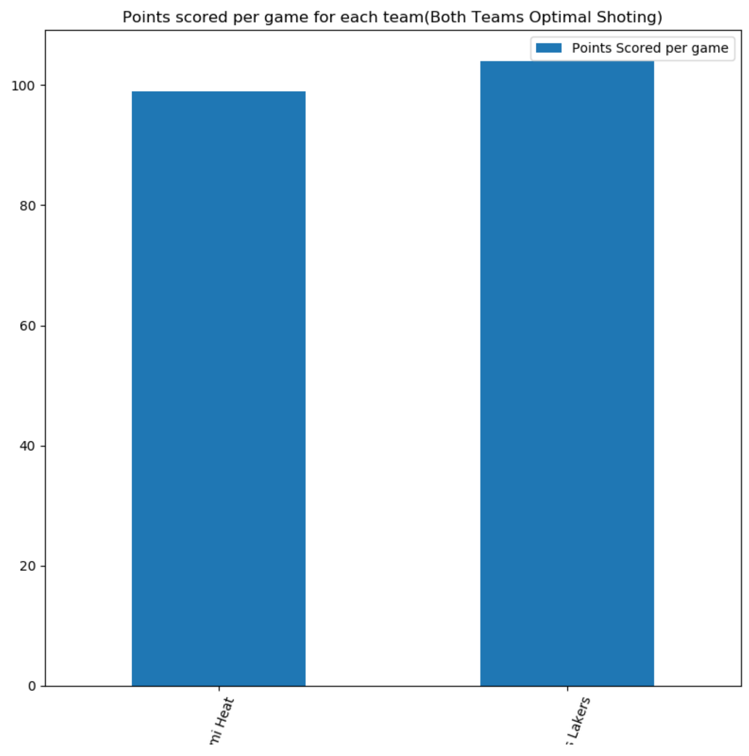


Figure 9: *Average points per game per team, Miami Heat Playing GTO strategy, Lakers playing GTO strategy*

	Miami Heat	Los angeles Lakers
Games won[%]	40.03	59.97
Series won[%]	29.83	70.17
Points scored	96.3	100.4

Table 5: No team playing GTO, 10 000 simulations.

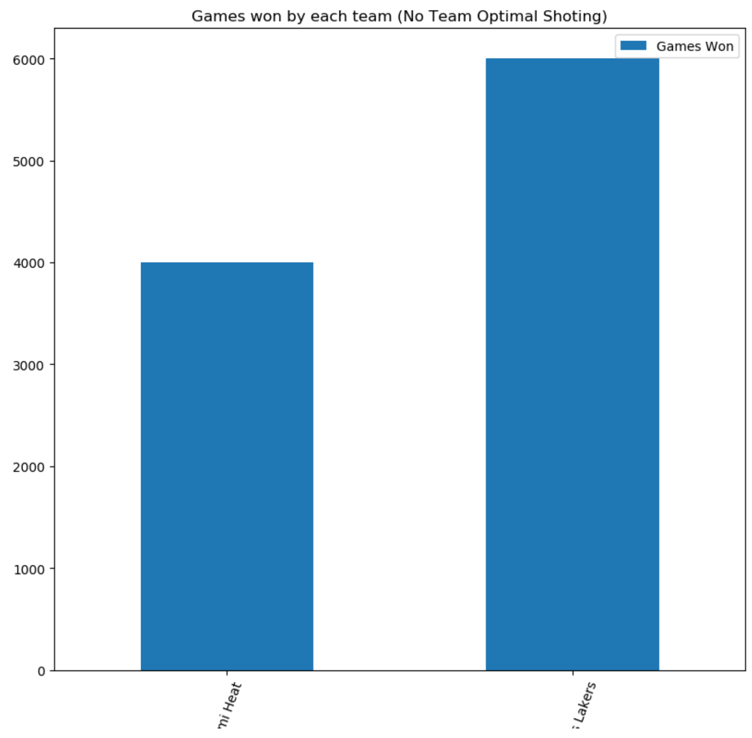


Figure 10: *Games won per team, Miami Heat Playing RLM strategy, Lakers playing RLM strategy*

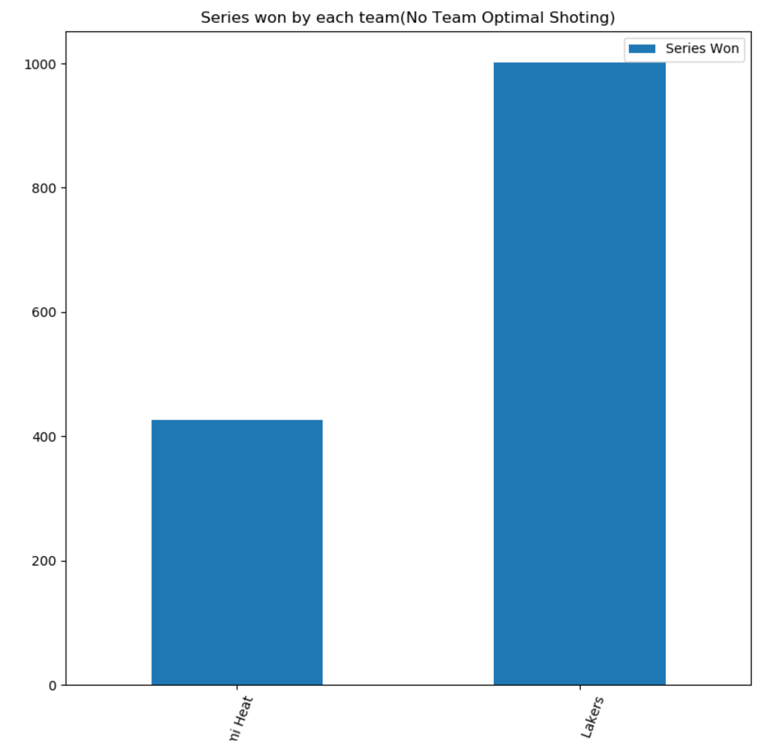


Figure 11: *7 games series won per team, Miami Heat Playing RLM strategy, Lakers playing RLM strategy*

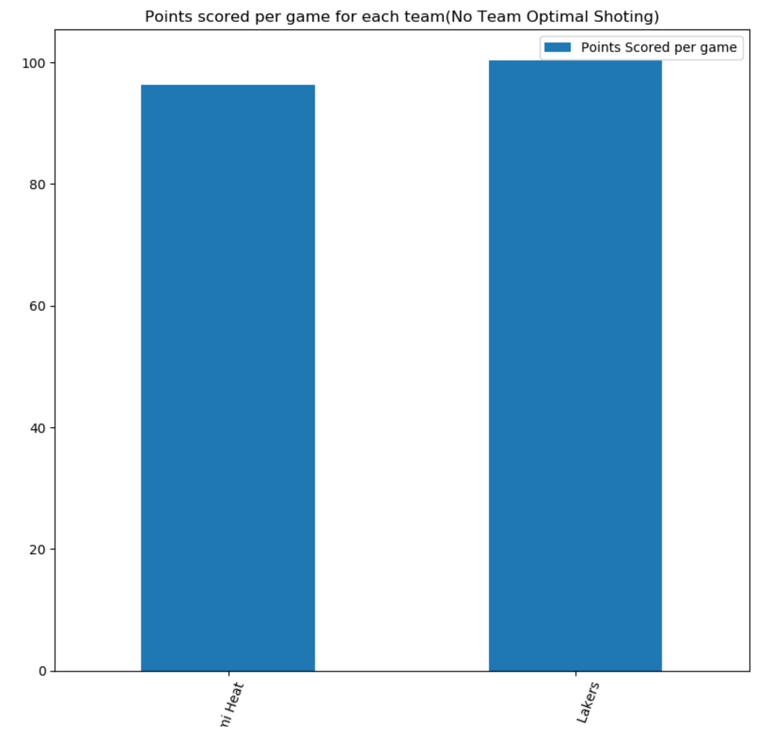


Figure 12: *Average points per game per team, Miami Heat Playing RLM strategy, Lakers playing RLM strategy*

6 Discussion

When analyzing at the result we consider this project a success. We ware able to reach our goal to find a strategy based on GT that would increase the amount of points scored, games won and series won. Granted that our simulation does not accurately represents a NBA game in all aspects due to the many assumptions and simplifications. But if it did the team that played according to the GT optimal strategy would have a greater chance of winning the NBA championship if the opponent ware to play according to their RLM strategy.
For someone following NBA and American sports in general know that the Americans are obsessed with stats surrounding players and teams. A popular trend in American sports lately is that coaches uses analytics to plan out their strategy. In our case the analytics we made, GTO suggests that the strategy 3 point shot should be selected more often than the current distribution between strategies. This aligns with the current climate of the NBA where the trend is to shoot more 3 point shots (Goldsberry, 2020). We interpret this as a good sign that our result is realistic even tough many assumptions and simplifications has been made.

To make the Model even better there is no limit for what parameters can be included to make the simulations more realistic. But we feel as the next step could be to introduce a ”Defender rating” where some defenders affects the shooter more or less and also depends on the shot the player is taking instead of having them as constants(-0.1 and +0.1). Other improvements could be Player match ups, substitutions, non uniform randomness for the time of possession, fouls, free throws and more. The Model could also be improved so that the user can choose what teams they want to simulate a game between.

7 Conclusion

We are happy with the results and feel as the we could successfully build a model that use data from NBA.com and apply GT to come up with an optimal strategy when it comes to shooting in a basketball game. The GTO strategy also made it so that the team utilizing it scored more point then when using RLM strategy when running the simulation

8 References

1. Badenhausen, Kurt. 2020. *NBA Team Values 2020: Lakers And Warriors Join Knicks In Rarefied $4 Billion Club*. Forbes. https://www.forbes.com/sites/kurtbadenhausen/2020/02/11/nba-team-values-2020-lakers-and-warriors-join-knicks-in-rarefied-4-billion-club/409512162032 (Accessed 2020-10-15)

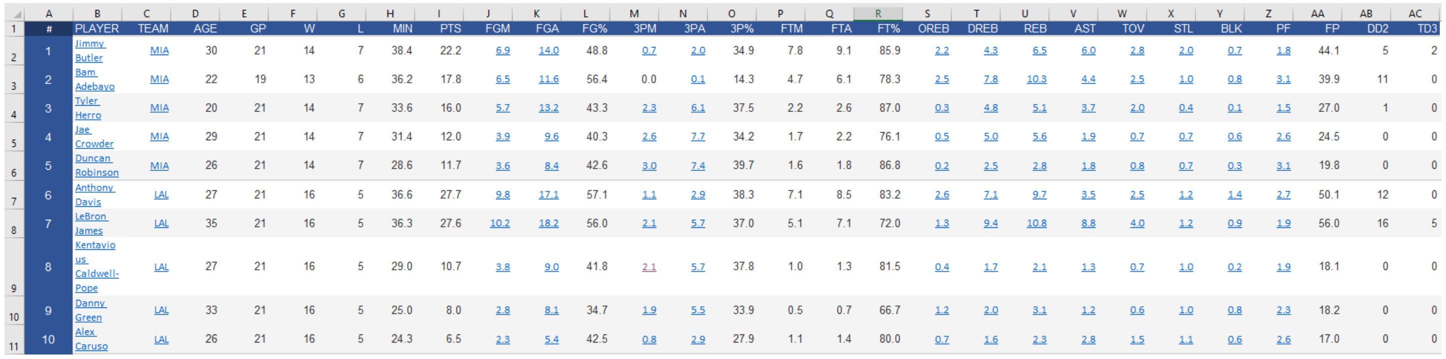
2. Easley, David; Kleinberg, Jon. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World* . Cambridge University Press,

3. NBA. 2020. *NBA Advanced Stats*. https://stats.nba.com/ (Accessed 2020-10-12)

4. Goldsberry, Kirk. 2019. *The NBA is obsessed with 3s, so let's finally fix the thing*. ESPN. https://www.espn.com/nba/story/_/id/26633540/the-nba-obsessed-3s-let-fix-thing (Accessed 2020-10-20)

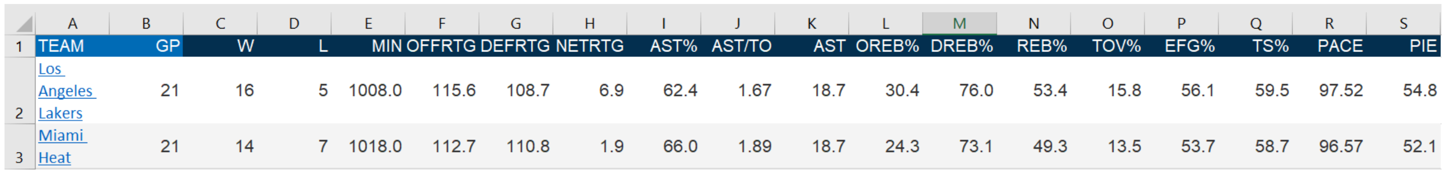
9 Appendix

9.1 Excel Player Stats



#	PLAYER	TEAM	AGE	GP	W	L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	TOV	STL	BLK	PF	FP	DD2	TD3
1	Jimmy Butler	MIA	30	21	14	7	38.4	22.2	6.9	14.0	48.8	0.7	2.0	34.9	7.8	9.1	85.9	2.2	4.3	6.5	6.0	2.8	2.0	0.7	1.8	44.1	5	2
2	Bam Adebayo	MIA	22	19	13	6	36.2	17.8	6.5	11.6	56.4	0.0	0.1	14.3	4.7	6.1	78.3	2.5	7.8	10.3	4.4	2.5	1.0	0.8	3.1	39.9	11	0
3	Tyler Herro	MIA	20	21	14	7	33.6	16.0	5.7	13.2	43.3	2.3	6.1	37.5	2.2	2.6	87.0	0.3	4.8	5.1	3.7	2.0	0.4	0.1	1.5	27.0	1	0
4	Jae Crowder	MIA	29	21	14	7	31.4	12.0	3.9	9.6	40.3	2.6	7.7	34.2	1.7	2.2	76.1	0.5	5.0	5.6	1.9	0.7	0.7	0.6	2.6	24.5	0	0
5	Duncan Robinson	MIA	26	21	14	7	28.6	11.7	3.6	8.4	42.6	3.0	7.4	39.7	1.6	1.8	86.8	0.2	2.5	2.8	1.8	0.8	0.7	0.3	3.1	19.8	0	0
6	Anthony Davis	LAL	27	21	16	5	36.6	27.7	9.8	17.1	57.1	1.1	2.9	38.3	7.1	8.5	83.2	2.6	7.1	9.7	3.5	2.5	1.2	1.4	2.7	50.1	12	0
7	LeBron James	LAL	35	21	16	5	36.3	27.6	10.2	18.2	56.0	2.1	5.7	37.0	5.1	7.1	72.0	1.3	9.4	10.8	8.8	4.0	1.2	0.9	1.9	56.0	16	5
8	Kentavious Caldwell-Pope	LAL	27	21	16	5	29.0	10.7	3.8	9.0	41.8	2.1	5.7	37.8	1.0	1.3	81.5	0.4	1.7	2.1	1.3	0.7	1.0	0.2	1.9	18.1	0	0
9	Danny Green	LAL	33	21	16	5	25.0	8.0	2.8	8.1	34.7	1.9	5.5	33.9	0.5	0.7	66.7	1.2	2.0	3.1	1.2	0.6	1.0	0.8	2.3	18.2	0	0
10	Alex Caruso	LAL	26	21	16	5	24.3	6.5	2.3	5.4	42.5	0.8	2.9	27.9	1.1	1.4	80.0	0.7	1.6	2.3	2.8	1.5	1.1	0.6	2.6	17.0	0	0

9.2 Excel Team Stats



TEAM	GP	W	L	MIN	OFFRTG	DEFRTG	NETRTG	AST%	AST/TO	AST	OREB%	DREB%	REB%	TOV%	EFG%	TS%	PACE	PIE
Los Angeles Lakers	21	16	5	1008.0	115.6	108.7	6.9	62.4	1.67	18.7	30.4	76.0	53.4	15.8	56.1	59.5	97.52	54.8
Miami Heat	21	14	7	1018.0	112.7	110.8	1.9	66.0	1.89	18.7	24.3	73.1	49.3	13.5	53.7	58.7	96.57	52.1

9.3 Python Code

```python
### import ###
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import random
from sympy import symbols, Eq, solve
import time

#choose one mode, choosing both will be messy
SimMode = False       #set to true to simulate n number of games
n = 10000
Watchmode = True    #set to true to "watch" a game,
T = 1 #set T to how many minutes you want a game to last


#choose what shot-strategy each team will have, True = GTO, False = RLM
MIHOptOff = True
LALOptOff = True



#Shooting Functions:
def Shooting_Real(Player):
    ''' this function simuletes What shot
    the offencive player chooses to shoot
    based on data gathered from NBA.com
    (plyoffs 19-20). (defencive player
    is assumed to adapt and defend 2pt and 3pt
    poportinatly to the offencive player'''

    ###Shotig Percentages
    ShootingPrc = np.array([[(float(ThreePtPRc[Player])*0.01-DefDifThree),...
                             (float(TwoPtPrc[Player])*0.01+DefDifTwo)],...
                            [(float(ThreePtPRc[Player])*0.01+DefDifThree),...
                             (float(TwoPtPrc[Player])*0.01-DefDifTwo)]])

    ### Defence Choice:
    rand = np.random.random()
    if rand < ThreePtQuota[Player]:
        Def = 0                         #defend 3 pt
    else:
        Def = 1                         #defend 2 pt

    ###Offencive Choice:
    rand = np.random.random()
    if rand < ThreePtQuota[Player]:
        shot = 0                         #shoot 3 pt
    else:
        shot = 1                         #shoot 2 pt

    #does the ball go in?
    pts = 0
    rand = np.random.random()
    if Def == 0 and shot == 0 :
        if rand < ShootingPrc[0,0]:
            pts = 3
    elif Def == 1 and shot == 0 :
        if rand < ShootingPrc[1,0]:
            pts = 3
    elif Def == 0 and shot == 1 :
        if rand < ShootingPrc[0,1]:
            pts = 2
    elif Def == 1 and shot == 1 :
        if rand < ShootingPrc[1,1]:
            pts = 2

    return pts


def Shooting_Opt(Player):
    ''' this function simuletes What shot
    the offencive player chooses to shoot
    based on optimal strategy. (defencive player
    is assumed to adapt and defend 2pt and 3pt
    poportinatly to the offencive player'''

    ###Shotig Percentages
    ShootingPrc = np.array([[(float(ThreePtPRc[Player])*0.01-DefDifThree),...
                             (float(TwoPtPrc[Player])*0.01+DefDifTwo)],...
                            [(float(ThreePtPRc[Player])*0.01+DefDifThree),...
                             (float(TwoPtPrc[Player])*0.01-DefDifTwo)]])

    ### Defence Choice:
    rand = np.random.random()
    if rand < OptOffPlay[Player]:
        Def = 0                     #defend 3 pt
    else:
        Def = 1                     #defend 2 pt

    ###Offencive Choice:
    rand = np.random.random()
    if rand < OptOffPlay[Player]:
        shot = 0                     #shoot 3 pt
    else:
        shot = 1                     #shoot 2 pt

    #does the ball go in?
    pts = 0
    rand = np.random.random()
    if Def == 0 and shot == 0 :
        if rand < ShootingPrc[0,0]:
            pts = 3
    elif Def == 1 and shot == 0 :
        if rand < ShootingPrc[1,0]:
            pts = 3
    elif Def == 0 and shot == 1 :
        if rand < ShootingPrc[0,1]:
            pts = 2
    elif Def == 1 and shot == 1 :
        if rand < ShootingPrc[1,1]:
            pts = 2

    return pts


####importing NBA Player- and Team stats
#(stats in excell file is from playoffs ss19-20
#except 3pt% for adebayo wich is from reg ss 19-20)

###Team Data
TeamStat = pd.read_excel('teamstat.xlsx', header=None)
TeamStat = TeamStat.values
TovPrc = np.array([float(TeamStat[1,14])*0.01,float(TeamStat[2,14])*0.01])
NormDReb = np.array([(float(TeamStat[1,12])*0.01) / (float(TeamStat[1,12])*0.01.
            + float(TeamStat[2,11])*0.01),(float(TeamStat[2,12])*0.01) /...
            (float(TeamStat[2,12])*0.01 + float(TeamStat[1,11])*0.01)])

###PLayer Data
PlayerStat = pd.read_excel('playerstat.xlsx', header=None)
PlayerStat = PlayerStat.values
Players = PlayerStat[1:11,1]
TwoPtPrc = PlayerStat[1:11,11]
ThreePtPRc = PlayerStat[1:11,14]

ThreePtQuota =np.zeros(10)
for i in range(10):
    ThreePtQuota[i] = float(PlayerStat[i+1,13]) / (float(PlayerStat[i+1,10])....
                                                   + float(PlayerStat[i+1,13]))

SPLakers = np.zeros(5)
SPHeat = np.zeros(5)
for i in range(5):
    SPHeat[i] = float(PlayerStat[i+1,10]) + float(PlayerStat[i+1,13])
    SPLakers[i] = float(PlayerStat[i+6,10]) + float(PlayerStat[i+6,13])
TotAtHeat = np.sum(SPHeat)
TotAtLakers = np.sum(SPLakers)

for i in range(5):
    if i == 0:
        SPHeat[i] = SPHeat[i] / TotAtHeat
        SPLakers[i] = SPLakers[i] / TotAtLakers
    else:
        SPHeat[i] = (SPHeat[i] / TotAtHeat) + SPHeat[i-1]
        SPLakers[i] = (SPLakers[i] / TotAtLakers) + SPLakers[i-1]

DefDifThree = 0.1
DefDifTwo = 0.1
OptOffPlay = np.zeros(10)
for i in range(10):
    DPayoffM = np.array([[-(float(ThreePtPRc[i])*0.01-DefDifThree)*3, ...
                          -(float(TwoPtPrc[i])*0.01+DefDifTwo)*2],...
                         [-(float(ThreePtPRc[i])*0.01+DefDifThree)*3,...
                          -(float(TwoPtPrc[i])*0.01-DefDifTwo)*2]])
    p = symbols('p') # probability p that the the offencive player shoots 3pt sho
    eq = Eq( p*DPayoffM[0,0] + (1-p)*DPayoffM[1,0] - ...
             (p*DPayoffM[0,1] + (1-p)*DPayoffM[1,1]),0)
    sol = solve(eq)
    OptOffPlay[i] = sol[0]


if Watchmode:
    n = 1
    t = T/48
```

```
############### Game starts ################

HPtsAllowed = np.zeros(n)
LPtsAllowed = np.zeros(n)
HWins = np.zeros(n)
LWins = np.zeros(n)
HSwins = 0
LSwins = 0
for i in range(n):
    ###jump ball
    GameTime = 0
    Jump = np.random.random()
    if Jump<0.5:
        pos = 0          #Lakers ball
    else:
        pos = 1          #Heat Ball
    GameTime = 0
    Lpoints = 0
    Hpoints = 0
    Halftime = True
    Q1 = True
    Q3 = True
    end = False
    while GameTime < 2880 and not end  : # 1 loop = 1 game
        TimePos = np.random.random()*24       #time of possetion
        GameTime = GameTime + TimePos          #gameclock
        MinTime = math.floor(GameTime/60)
        SecTime = math.floor(GameTime%60)
        Clock = str(MinTime)+':'+str(SecTime)

        turnover = 0                           #turnover?
        rand = np.random.random()
        if rand < TovPrc[pos]:
            Turnover = 1
            if pos == 0:
                pos = 1
            else:
                pos = 0

        elif pos == 0 and turnover == 0:    #Lakers shooting
            rand = np.random.random()       #who is shooting
            if rand < SPLakers[0]:
                Player = 5
            elif SPLakers[0] < rand < SPLakers[1]:
                Player = 6
            elif SPLakers[1] < rand < SPLakers[2]:
                Player = 7
            elif SPLakers[2] < rand < SPLakers[3]:
```

```
            Player = 8
        elif SPLakers[3] < rand < SPLakers[4]:
            Player = 9

        if LALOptOff:                          #what strategy?
            Pts = Shooting_Opt(Player)
        else:
            Pts = Shooting_Real(Player)

        Lpoints = Lpoints + Pts

        if Pts == 0:                          #rebound
            rand = np.random.random()
            if rand < NormDReb[pos+1]:
                pos = 1
            else:
                pos = 0
        else:
            pos = 1
            if Watchmode:
                print(Players[Player], 'Scores', Pts,...
                      'At time', Clock,'To make it, Miami Heat',...
                      Hpoints, '-', Lpoints, 'Los Angeles Lakers')
                time.sleep(t*TimePos)

    else:                                   #Heat shooting
        rand = np.random.random()           #who is shooting
        if rand < SPHeat[0]:
            Player = 0
        elif SPHeat[0] < rand < SPHeat[1]:
            Player = 1
        elif SPHeat[1] < rand < SPHeat[2]:
            Player = 2
        elif SPHeat[2] < rand < SPHeat[3]:
            Player = 3
        elif SPHeat[3] < rand < SPHeat[4]:
            Player = 4

        if MIHOptOff:                          #what strategy?
            Pts = Shooting_Opt(Player)
        else:
            Pts = Shooting_Real(Player)

        Hpoints = Hpoints + Pts

        if Pts == 0:                        #rebound
            rand = np.random.random()
            if rand < NormDReb[pos-1]:
                pos = 0
            else:
                    pos = 1

            else:
                pos = 0
                if Watchmode:
                    print(Players[Player], 'Scores', Pts,...
                    'At time', Clock,'To make it, Miami Heat',...
                    Hpoints, '-', Lpoints, 'Los Angeles Lakers')
                    time.sleep(t*TimePos)

        if Watchmode and GameTime > 1440 and Halftime:
            print('At halftime the score is:', 'Miami Heat',...
                  Hpoints, '-', Lpoints, 'Los Angeles Lakers')
            Halftime = False
        if Watchmode and GameTime > 720 and Q1:
            print('After quarter 1 the score is:', 'Miami Heat',...
                  Hpoints, '-', Lpoints, 'Los Angeles Lakers')
            Q1= False
        if Watchmode and GameTime > 2160 and Q3:
            print('After quarter 3 the score is:', 'Miami Heat',...
                  Hpoints, '-', Lpoints, 'Los Angeles Lakers')
            Q3 = False

        if GameTime > 2880  and Hpoints == Lpoints:   #overtime
            GameTime = 2580

    HPtsAllowed[i] = Lpoints
    LPtsAllowed[i] = Hpoints
    if Hpoints < Lpoints:
        LWins[i] = 1
        end = True
    else:
        HWins[i] = 1
        end = True

    if Watchmode:
        print('Game ends: Miami Heat', Hpoints, '-', Lpoints, ...
              'Los Angeles Lakers')

    if end:
        if (i+1)%7 == 0:
            HWS = sum(HWins[i-6:i+1])
            LWS = sum(LWins[i-6:i+1])
            if HWS < LWS:
                LSwins = LSwins + 1
            else:
                HSwins = HSwins + 1


if SimMode:
```

```
HWins = sum(HWins)
LWins = sum(LWins)
HPtsAllowedPG = sum(HPtsAllowed) / n
LPtsAllowedPG = sum(LPtsAllowed) / n
print('Games',HWins,LWins)
print('Series', HSwins, LSwins)
print('Pts Scored', LPtsAllowedPG, HPtsAllowedPG)

data = {"Team":["Miami Heat", "Los Angeles Lakers"],...
        "Series Won":[HSwins,LSwins]}
dataFrame = pd.DataFrame(data=data)
dataFrame.plot.bar(x="Team", y="Series Won", rot=70,...
                   title="Series won by each team")
plt.show()

data = {"Team":["Miami Heat", "Los Angeles Lakers"],...
        "Games Won":[HWins,LWins]}
dataFrame = pd.DataFrame(data=data)
dataFrame.plot.bar(x="Team", y="Games Won", rot=70,...
                   title="Games won by each team")
plt.show()

data = {"Team":["Miami Heat", "Los Angeles Lakers"],...
        "Points Scored per game":[LPtsAllowedPG,HPtsAllowedPG]}
dataFrame = pd.DataFrame(data=data)
dataFrame.plot.bar(x="Team", y="Points Scored per game",...
        rot=70, title="Points scored per game for each team")
plt.show()
```