Monopoly Board Game Simulation

Laila Ahmed Hamza ~ 900201723

&

Mahmoud Ghandour ~ 900201365

Fundamentals of Simulation – DSCI 3411- 01

Dr. Ali Hadi

Final Project

May 20th, 2023

Question of Interest

Which states in the Monopoly board game have the highest probability that a player will stop at? What conclusions can be drawn from the results to improve one's performance while playing the game?

Introduction

Simulation is a well-known word in today's world. Simulations involve the generation of artificial data or the replication of real-world scenarios using random or deterministic algorithms. By repeatedly running these simulations, we can observe the behavior of a system under different conditions, investigate the impact of different variables, and assess the likelihood of specific outcomes. Moreover, its application in the real world is enormous and undoubtedly essential, as it enables us to address questions that may be challenging to answer analytically or in situations where real-world data is limited, expensive, or difficult to obtain. In our case, we used simulation to learn more about a popular board game, Monopoly. A board game like no other that is a symbol of camaraderie and shared experiences. A board game that always fascinates the players and the audience. Whenever we lost a game, we always blamed our "luck," and we lost because of it. However, we never questioned our decisions and strategy, so now, as we delve into the world of strategy and data analysis, we seek to unravel the secrets of Monopoly and uncover the best strategies to secure victory. Through careful analysis and simulation, we hope to gain insights into the game's dynamics and develop a winning approach that surpasses the limitations of our previous experiences.

Description of the Problem

Monopoly is a game of skill, strategy, and luck, where players aim to accumulate wealth, properties, and monopolies while bankrupting their opponents. The primary challenge in Monopoly is to develop a winning strategy that maximizes income, minimizes expenses, and optimizes the overall financial position. The game involves various strategic decisions, such as property acquisitions, the development of houses and hotels, and the management of cash flow.

Objective

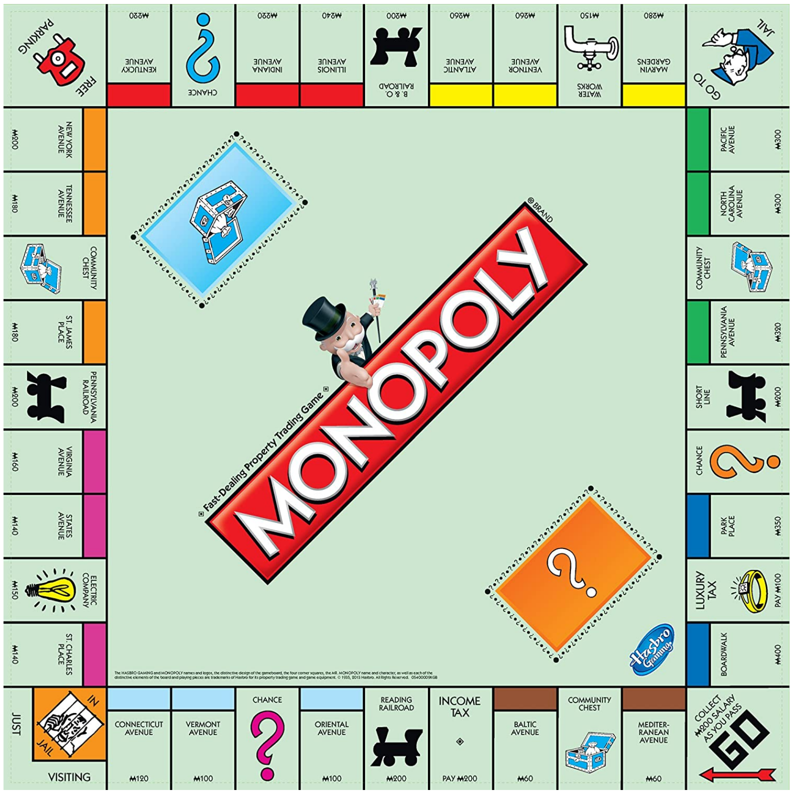
This report aims to utilize Monte Carlo simulation to identify the best states to rent in the popular board game Monopoly, based on the highest probability that players will land in these particular states. Monopoly is a game of chance and strategy, where players move around the board and aim to acquire properties, collect rent, and bankrupt their opponents. One crucial aspect of the game is strategically choosing properties to purchase and develop, with the ultimate goal of generating a steady income stream through rent payments. To make informed decisions about property acquisitions, it is essential to understand the probabilities associated with players landing in different states.

In this report, we will employ Monte Carlo simulation to estimate the likelihood of players landing on each state in Monopoly. By simulating a large number of gameplays, we can generate a distribution of landing probabilities for each state. This simulation-based approach allows us to capture the randomness and uncertainty inherent in the game and provides a more accurate estimation of the probabilities. The primary focus of this analysis is to identify the state with the highest probability of player landings, making it an attractive choice for property rental. We will examine the frequency of player visits to each state and calculate the corresponding probabilities based on the simulation results. By comparing these probabilities, we can determine the most favorable states to rent, with the highest likelihood of generating consistent rental income.

Additionally, we will explore the impact of various factors on player movement across the board. This includes considering the effects of Chance and Community Chest cards, unique board spaces (such as Go, Jail, and Free Parking), and the presence of other players' properties. By incorporating these factors into our simulation, we can obtain a more realistic representation of player behavior and enhance the accuracy of our probability estimates. By identifying the states with the highest probability of player landings, players can strategically position themselves to maximize their rental income and gain a competitive advantage in the game. The Monte Carlo simulation approach ensures a robust and data-driven analysis, considering the game's dynamics and uncertainties.

Input

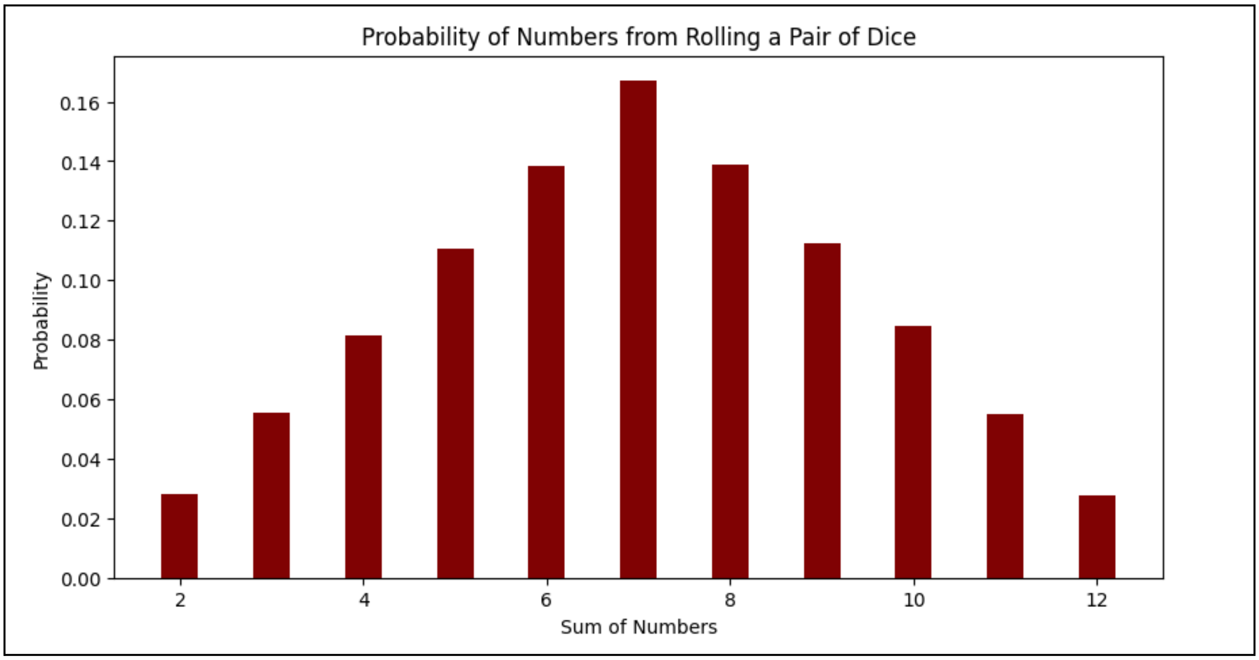
This project was built around Monopoly's original version, so the rules and formatting followed throughout this report adhere to the ones found in that version. We started by declaring our game's necessary states or variables: Board, Community Chest, and Chance cards. There are a total of 40 states on our board. We also considered the cards as they are essential tools that dictate the direction the game follows. The pictures below show the board, community chest cards, and chance cards, respectively.




```
['Advance to GO',
 'Bank error in your favor',
 "Doctor's fees",
 'Get out of Jail free',
 'Go to Jail',
 'Grand Opera Night',
 'Holiday Fund matures',
 'Income Tax refund',
 "It's your birthday",
 'Life insurance matures',
 'Pay hospital fees',
 'Pay school fees',
 'Receive $25 consultancy fee',
 'You are assessed for street repairs',
 'You have won second prize in a beauty contest',
 'You inherit $100',
 'From sale of stock you get $50',
 'Holiday Fund matures']
```

```
['Advance to GO',
 'Advance to Illinois Avenue',
 'Advance to St. Charles Place',
 'Advance to nearest Utility',
 'Advance to nearest Railroad',
 'Bank pays you dividend',
 'Get out of Jail free',
 'Go back 3 spaces',
 'Go to Jail',
 'Make general repairs on all your properties',
 'Pay poor tax',
 'Take a trip to Reading Railroad',
 'Take a walk on the Boardwalk',
 'You have been elected Chairman of the Board',
 'Your building and loan matures',
 'You have won a crossword competition']
```

During each turn, a player tosses two dice simultaneously and moves a number of spaces on the board that is equivalent to the sum of the dice. Accordingly, we defined a function that randomly creates two dice and sums their values. We computed the counts of each possible value (sums from 2 to 12) and used the result to generate a probability distribution function plot. The result was a normal distribution, depicted by the following graph:



From the graph, we can observe that the distribution is centered around the value of 7, making this value both the mean and the median value of the data. We can therefore conclude that the most likely number of steps a Monopoly player will take during his/her turn is seven. In other words, the player will get one of the following combinations: [(1,6),(5,2),(4,3)], where the dice values are interchangeable within each possibility.

The following step was computing the transition probabilities of moving from one state to another (one property tile on the board to another). This was done by first defining the different "states" that exist on the standard Monopoly board. A 10,000-iteration Monte Carlo simulation was made to run on these defined states in order to generate the list of possible positions a player could land on. This provides the probability of a player landing on a state given that he or she is in a random state X, which is the PDF of the dice between 2 steps & 12 steps, with zero probability of landing in any other state. A For loop was made to simulate the rolling of the dice and the consequent 'moving' of the player. The code also includes specific modifications for special cases like Go to Jail, Community Chest, and Chance Cards, which could change the player's position. Thus, the probability of visiting each of these possible positions was calculated, showing that the 'Jail' tile was the most commonly landed on tile in the game with a probability of 0.05898. Since 'Jail' and 'Go' represent exceptional cases that were high on the list of states with the highest probabilities, they were removed to allow for a more beneficial analysis. Below is the resulting list:

	Probability
Place	
Illinois Avenue	0.03292
New York Avenue	0.03062
Reading Railroad	0.02908
Tennessee Avenue	0.02907

Fig 2

Next, we analyzed the cost corresponding to each of the states available on the board by considering the relationship between the cost of a state and its transition probability. This cost represents the monetary amount in Monopoly Dollars that a player must pay in order to purchase a particular state. The expectation, going in, was that an approximately normal distribution would emerge from this relationship, with the majority of states being of average price and most frequently reached by players, leaving the cheapest and most expensive states being those with lower transition probabilities. This assumption was based on the theory that the easiest and hardest 'levels' of a game are usually less frequent than those of average difficulty.

However, this theory was proved false since the two variables did not seem to share any meaningful relationship (verified by an extremely low Pearson correlation coefficient of 0.141). Below is a scatterplot supporting this result. Visually, the data can seem approximately uniform with the exception of a few outliers with a cost equal to zero (non-purchasable).

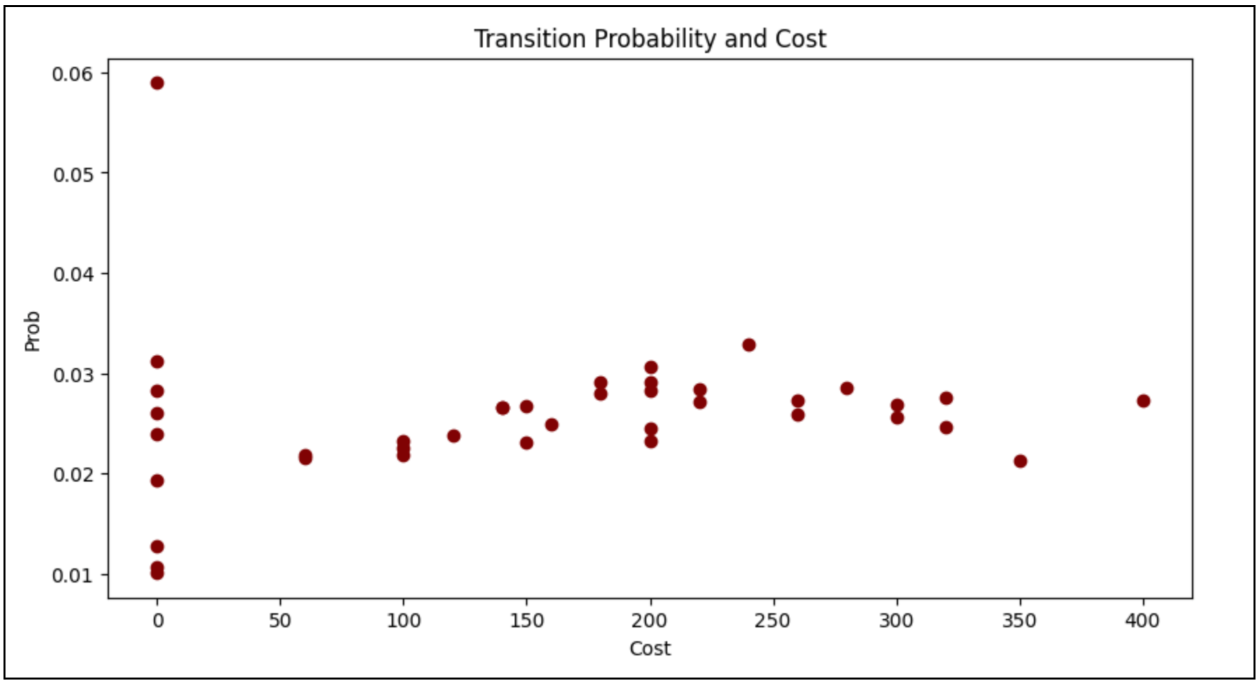


Fig 3

Once a player purchases a state, he or she can gather rent from other players should they land on this purchased property. The rent can increase based on certain conditions (having full sets of the same color, owning houses or hotels on that property, etc.) However, for the purposes of this project, we chose to overlook the nuances in rent amounts and only consider the base rent values. Again, we wanted to see whether there was a relationship between the transition probabilities of the states and their base rents. Again, the results indicated little correlation between the two variables (with a Pearson Coefficient that's slightly lower than the previous pair: 0.138). Therefore, the probability of landing in a state appears to be unrelated to the state's cost and base rent. Below is a figure depicting the base rent plotted against the transition probability.

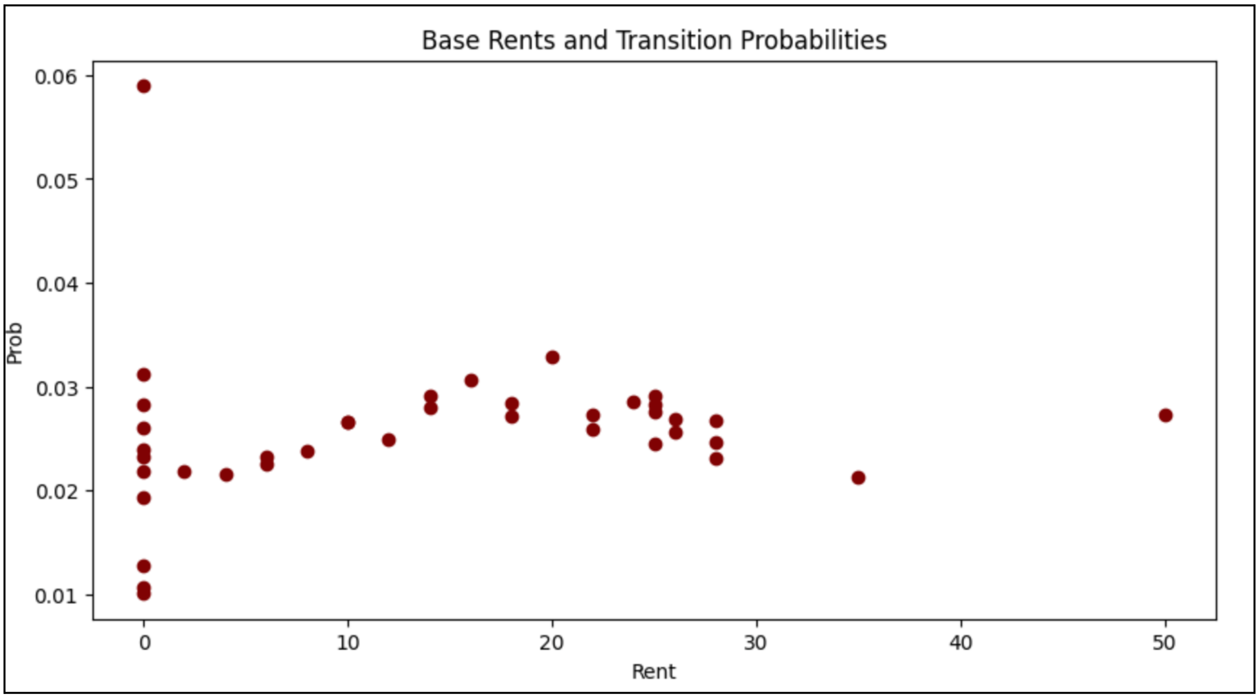


Fig 4

Again, we can see that apart from values falling on the zero rent line (perceived outliers to the pdf function), most states of different rents fall within the 0.02 and 0.03 transition probability range. In addition, it can be seen that two outliers exist in terms of the rent values (rent equal to approx. 35 and 50).

Next, we assessed the base rent-to-cost ratio since maximizing this ratio would be optimal for players. This is because, ideally, a player would maximize the inflow of profit sourced from rent relative to the cost he or she paid for that property state in the first place. Below is the resulting list.

Place	Probability	Cost	Rent	R2C
Water Works	0.02673	150	28	0.186667
Electric Company	0.02311	150	28	0.186667
B. & O. Railroad	0.02833	200	25	0.125000
Boardwalk	0.02725	400	50	0.125000
Reading Railroad	0.02908	200	25	0.125000
Short Line	0.02442	200	25	0.125000
Park Place	0.02122	350	35	0.100000
Pennsylvania Avenue	0.02457	320	28	0.087500
North Carolina Avenue	0.02561	300	26	0.086667
Pacific Avenue	0.02681	300	26	0.086667

Fig 5

From this list, we can identify Reading Railroad as one of the 4 states on the list of highest transition probabilities. Therefore, considering the cost, the base rent, and the probability of landing in the specific state, Reading Railroad appears to be the optimal option. Investing in this state would be the best use of a player's money since it generates high rent, and players will have a higher chance of landing on it (thus having to pay the aforementioned rent).

Recommendations

Diving deeper into our Monte Carlo simulation conducted to identify the best states to rent in Monopoly, it is evident that the orange and red states demonstrate a higher probability of players landing on them. This provides valuable insights and suggests that these states may present favorable rental opportunities. Therefore, it is recommended to prioritize investments in the orange and red properties when playing Monopoly. These properties include St. James Place (orange), Tennessee Avenue (orange), New York Avenue (orange), Kentucky Avenue (red), Indiana Avenue (red), and Illinois Avenue (red). These states are strategically positioned on the board, attracting significant player traffic due to their location. Especially orange states, as they have the optimal distance and the highest chance of rolls (6, 7, 8) away from the most visited place (Jail). Red states are almost two rolls away from Jail. Consequently, focusing on the orange and red properties as key rental opportunities in Monopoly can enhance your chances of generating consistent income and gaining a competitive advantage. However, it is essential to note that while the orange and red states have a higher probability of player landings, there is still an element of chance involved in Monopoly. Other factors, such as dice rolls, the actions of other players, and card draws, can impact the outcomes. Therefore, it is crucial to consider the identified states as potential opportunities rather than definitive guarantees. And at the end, a recommendation is just a recommendation. **“You can lead a horse to water, but you can't make it drink.”**

Conclusion

By conducting extensive simulations and analyzing the probabilities associated with player movements, we were able to make valuable conclusions. Through our Monte Carlo simulation, we observed that the probability of players landing on specific states varied significantly. Certain states, such as those near the Jail state, were more likely to be landed on by players due to the fact that the Jail state is the state with the highest probability that a player will stop at and then move from. Knowing this, the distribution of dice rolls empirically proves it, as these states have the highest probability to stop at, when starting from Jail, as they range from 7 to 14 positions away from Jail, so maybe 1 or 2 rolls away. Our analysis also revealed that properties with higher rent values and those located near popular locations, such as railroads or highly desirable properties, tended to generate higher income over time. These properties offered a competitive advantage and contributed to a player's overall financial success. Moreover, these states presented strategic opportunities for rental income. Finally, by leveraging the insights gained from the Monte Carlo simulation, players can strategically invest in and build properties more likely to be landed on, increasing their chances of generating rental income and gaining a competitive edge.

References

Morgan, B. J. T. (1995). *Elements of Simulation*. Chapman and Hall.

Koehrsen, W. (2018, April 5). *Markov chain Monte Carlo in python*. Medium. https://towardsdatascience.com/markov-chain-monte-carlo-in-python-44f7e609be98

Rashed, F. (2020, December 3). *Oh, the places you'll go in Monopoly*. Medium. https://towardsdatascience.com/oh-the-places-youll-go-in-monopoly-96abf70cdbd7

Appendix

Figure 1:

1. Import Necessary Libraries

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import random
    from random import sample
    from scipy.stats import pearsonr
```

Figure 2:

2. Run Dice Summing Simulation

[2]
```
def dice(n):
    return [random.randint(1, 6) + random.randint(1, 6) for _ in range(n)]
```

[3]
```
sim_dice = dice(100000) #1000 simulation iterations of summing random dice values
counts = pd.value_counts(sim_dice)
```

[4]
```
dice_df = pd.DataFrame(counts)
dice_df.columns = ["counts"]
dice_df.head(10)
```

	counts
7	16708
8	13877
6	13859
9	11252
5	11072
10	8464
4	8156
3	5532
11	5499
2	2825

```
fig1 = plt.figure(figsize = (10, 5))
# creating the bar plot
plt.bar(dice_df.index, dice_df['counts']/100000, color ='maroon',
        width = 0.4)

plt.xlabel("Sum of Numbers")
plt.ylabel("Probability")
plt.title("Probability of Numbers from Rolling a Pair of Dice")
plt.show()
```

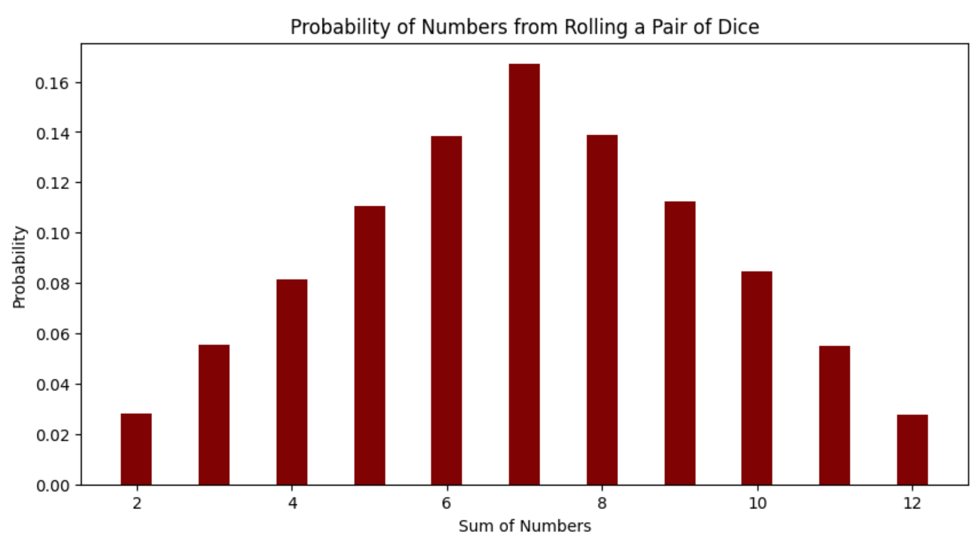


Figure 3A:

3. Transition Probabilities:

```
positions = list(range(40))
visits = [0] * 40

board = [
    "GO", "Mediterranean Avenue", "Community Chest # 1", "Baltic Avenue", "Income Tax",
    "Reading Railroad", "Oriental Avenue", "Chance # 1", "Vermont Avenue", "Connecticut Avenue",
    "Jail", "St. Charles Place", "Electric Company", "States Avenue", "Virginia Avenue",
    "Pennsylvania Railroad", "St. James Place", "Community Chest # 2", "Tennessee Avenue", "New York Avenue",
    "Free Parking", "Kentucky Avenue", "Chance # 2", "Indiana Avenue", "Illinois Avenue",
    "B. & O. Railroad", "Atlantic Avenue", "Ventnor Avenue", "Water Works", "Marvin Gardens",
    "Go To Jail", "Pacific Avenue", "North Carolina Avenue", "Community Chest # 3", "Pennsylvania Avenue",
    "Short Line", "Chance # 3", "Park Place", "Luxury Tax", "Boardwalk"
]

community_chest_cards = [
    "Advance to GO", "Bank error in your favor", "Doctor's fees", "Get out of Jail free",
    "Go to Jail", "Grand Opera Night", "Holiday Fund matures", "Income Tax refund",
    "It's your birthday", "Life insurance matures", "Pay hospital fees", "Pay school fees",
    "Receive $25 consultancy fee", "You are assessed for street repairs", "You have won second prize in a beauty contest",
    "You inherit $100", "From sale of stock you get $50", "Holiday Fund matures"
]

chance_cards = [
    "Advance to GO", "Advance to Illinois Avenue", "Advance to St. Charles Place",
    "Advance to nearest Utility", "Advance to nearest Railroad", "Bank pays you dividend",
    "Get out of Jail free", "Go back 3 spaces", "Go to Jail", "Make general repairs on all your properties",
    "Pay poor tax", "Take a trip to Reading Railroad", "Take a walk on the Boardwalk",
    "You have been elected Chairman of the Board", "Your building and loan matures",
    "You have won a crossword competition"
]

# Run Monte Carlo simulation
current_position = 0 # Begin from GO

num_iterations = 100000

for _ in range(num_iterations):
    visits[current_position] += 1

    # Roll dice
    dice_roll = random.randint(1, 6) + random.randint(1, 6)

    # Move the player
    current_position = (current_position + dice_roll) % 40

    # Handle special positions
    if current_position == 30:  # Go To Jail
        current_position = 10
    elif current_position == 2 or current_position == 17 or current_position == 33:  # Community Chest
        # Draw a Community Chest card
        community_chest_card = random.choice(community_chest_cards)

        if community_chest_card == "Advance to GO":
            current_position = 0
        elif community_chest_card == "Go to Jail":
            current_position = 10
```

Figure 3B:

```
    elif current_position == 7 or current_position == 22 or current_position == 36:  # Chance
        # Draw a Chance card
        chance_card = random.choice(chance_cards)

        if chance_card == "Advance to GO":
            current_position = 0
        elif chance_card == "Advance to Illinois Avenue":
            current_position = 24
        elif chance_card == "Advance to St. Charles Place":
            current_position = 11
        elif chance_card == "Advance to nearest Utility":
            if current_position < 13:
                current_position = 13
            elif current_position > 29:
                current_position = 13
            elif current_position < 29:
                current_position = 29
        elif chance_card == "Advance to nearest Railroad":
            if current_position < 5:
                current_position = 5
            elif current_position > 35:
                current_position = 5
            elif current_position < 15:
                current_position = 15
            elif current_position < 25:
                current_position = 25
            elif current_position < 35:
                current_position = 35
        elif chance_card == "Go back 3 spaces":
            current_position = current_position - 3
        elif chance_card == "Go to Jail":
            current_position = 10
        elif chance_card == "Take a trip to Reading Railroad":
            current_position = 5
        elif chance_card == "Take a walk on the Boardwalk":
            current_position = 39
```

```
[ ] # Calculate visit probabilities
    visit_probabilities = [visit_count / num_iterations for visit_count in visits]

    # Determine best strategy based on visit probabilities
    ind = np.argpartition(visit_probabilities, -39)[-39:]
    visit_probabilities = np.array(visit_probabilities)
    highest_probability = visit_probabilities[ind]
    board = np.array(board)
```

```
[ ] print("Best Strategy:")
    print("Focus on the following positions:")
    df = pd.DataFrame([board[ind] , visit_probabilities[ind]]).transpose()
    df = df.sort_values(by = [1],ascending = False)
    df.columns =['Place', 'Probability']
    df = df.set_index('Place')
    df.head(5)
```

```
Best Strategy:
Focus on the following positions:
```

Place	Probability
Jail	0.05898
Illinois Avenue	0.03292
GO	0.0312
New York Avenue	0.03062
Reading Railroad	0.02908

Figure 4:

4a. Measures of Centrality & Dispersion

4b. Assessment

```
print('Mean = ',np.mean(df.Probability))
print('Median = ', np.median(df.Probability))
print('Standard Deviation = ',np.std(df.Probability))
print('Quartiles = ',np.quantile(df.Probability, [0.25,0.5,0.75,0.85]))
```

```
Mean =  0.02564102564102564
Median =  0.026
Standard Deviation =  0.00728648161547642
Quartiles =  [0.02318 0.026 0.028124999999999997 0.028748]
```

```
quartiles = np.quantile(df.Probability, [0.25,0.5,0.75,0.85])
cutoff = quartiles[-1]
cutoff
```

```
0.028748
```

```
df_best = df[df > cutoff]
df_best = df_best.dropna()
df_best
```

Place	Probability
Jail	0.05898
Illinois Avenue	0.03292
GO	0.0312
New York Avenue	0.03062
Reading Railroad	0.02908
Tennessee Avenue	0.02907

```
df_best = df_best.drop(["Jail","GO"],axis = 0)
df_best
```

Place	Probability
Illinois Avenue	0.03292
New York Avenue	0.03062
Reading Railroad	0.02908
Tennessee Avenue	0.02907

Figure 5A, 5B:

5. Considering Cost:

```
[ ] d = { 'GO':0, "Mediterranean Avenue":60, "Community Chest # 1":0, "Baltic Avenue":60, "Income Tax":200,
    "Reading Railroad":200, "Oriental Avenue":100, "Chance # 1":0, "Vermont Avenue":100, "Connecticut Avenue":120,
    "Jail":0, "St. Charles Place":140, "Electric Company":150, "States Avenue":140, "Virginia Avenue":160,
    "Pennsylvania Railroad":320, "St. James Place":180, "Community Chest # 2":0, "Tennessee Avenue":180, "New York Avenue":200,
    "Free Parking":0, "Kentucky Avenue":220, "Chance # 2":0, "Indiana Avenue":220, "Illinois Avenue": 240,
    "B. & O. Railroad": 200, "Atlantic Avenue": 260, "Ventnor Avenue":260 , "Water Works": 150, "Marvin Gardens":280,
    "Go To Jail": 0, "Pacific Avenue": 300, "North Carolina Avenue":300, "Community Chest # 3":0, "Pennsylvania Avenue":320,
    "Short Line": 200, "Chance # 3": 0, "Park Place": 350, "Luxury Tax": 100, "Boardwalk": 400}
```

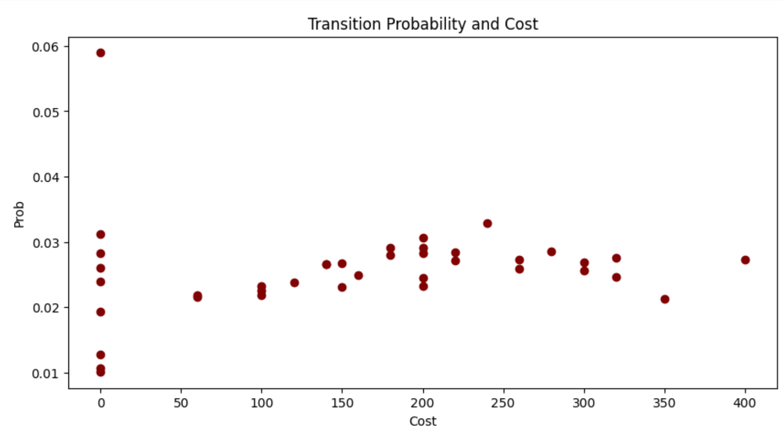
```
[ ] df["Cost"] = pd.Series(d)
```

```
df.head(10)
```

Place	Probability	Cost
Jail	0.05898	0
Illinois Avenue	0.03292	240
GO	0.0312	0
New York Avenue	0.03062	200
Reading Railroad	0.02908	200
Tennessee Avenue	0.02907	180
Marvin Gardens	0.02861	280
Kentucky Avenue	0.02835	220
B. & O. Railroad	0.02833	200
Free Parking	0.0283	0

```
fig2 = plt.figure(figsize = (10, 5))
# creating the bar plot
plt.scatter(df['Cost'], df['Probability'], color='maroon')

plt.xlabel("Cost")
plt.ylabel("Prob")
plt.title("Transition Probability and Cost")
plt.show()
```




```
[ ] # calculate Pearson's correlation
    corr, _ = pearsonr(df['Cost'], df['Probability'])
    print('Pearsons correlation: %.3f' % corr)

    Pearsons correlation: 0.141
```

Figure 6A, 6B:

6. Considering Base Rent:

```
d2 = { 'GO':0, "Mediterranean Avenue":2, "Community Chest # 1":0, "Baltic Avenue":4, "Income Tax":0,
     "Reading Railroad":25, "Oriental Avenue":6, "Chance # 1":0, "Vermont Avenue":6, "Connecticut Avenue":8,
     "Jail":0, "St. Charles Place":10, "Electric Company":28, "States Avenue":10, "Virginia Avenue":12,
     "Pennsylvania Railroad":25, "St. James Place":14, "Community Chest # 2":0, "Tennessee Avenue":14, "New York Avenue":16,
     "Free Parking":0, "Kentucky Avenue":18, "Chance # 2":0, "Indiana Avenue":18, "Illinois Avenue": 20,
     "B. & O. Railroad": 25, "Atlantic Avenue": 22, "Ventnor Avenue":22 , "Water Works": 28, "Marvin Gardens":24,
     "Go To Jail": 0, "Pacific Avenue": 26, "North Carolina Avenue":26, "Community Chest # 3":0, "Pennsylvania Avenue":28,
     "Short Line": 25, "Chance # 3": 0, "Park Place": 35, "Luxury Tax": 0, "Boardwalk": 50}
```

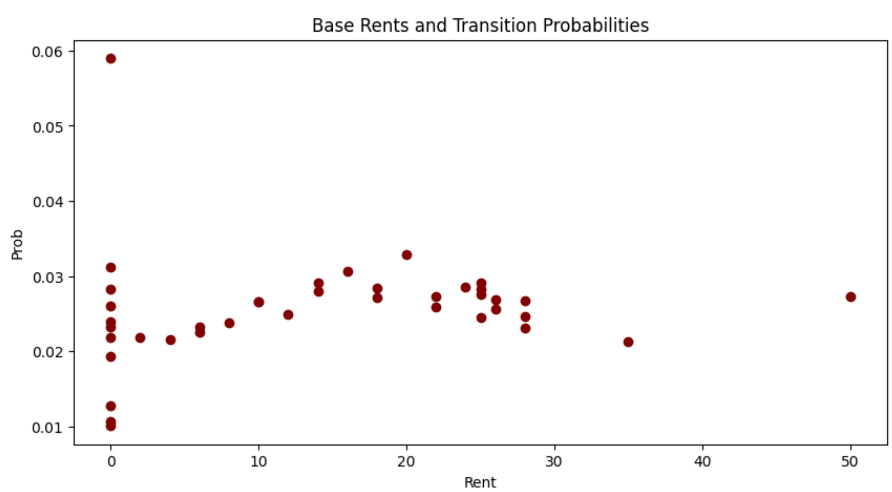
```
df["Rent"] = pd.Series(d2)
```

```
df.head(10)
```

Place	Probability	Cost	Rent
Jail	0.05898	0	0
Illinois Avenue	0.03292	240	20
GO	0.0312	0	0
New York Avenue	0.03062	200	16
Reading Railroad	0.02908	200	25
Tennessee Avenue	0.02907	180	14
Marvin Gardens	0.02861	280	24
Kentucky Avenue	0.02835	220	18
B. & O. Railroad	0.02833	200	25
Free Parking	0.0283	0	0

```
fig3 = plt.figure(figsize = (10, 5))
# creating the bar plot
plt.scatter(df['Rent'], df['Probability'], color='maroon')

plt.xlabel("Rent")
plt.ylabel("Prob")
plt.title("Base Rents and Transition Probabilities")
plt.show()
```




```
# calculate Pearson's correlation
corr, _ = pearsonr(df['Rent'], df['Probability'])
print('Pearsons correlation: %.3f' % corr)
```

Pearsons correlation: 0.138

Figure 7:

Seeking the Maximum Base Rent to Cost Ratio

```
[ ] df["R2C"] = (df["Rent"]/df["Cost"]).replace(np.nan, 0)
    df = df.sort_values(by=['R2C'], ascending=False)

    df.head(10)
```

Place	Probability	Cost	Rent	R2C
Water Works	0.02673	150	28	0.186667
Electric Company	0.02311	150	28	0.186667
B. & O. Railroad	0.02833	200	25	0.125000
Boardwalk	0.02725	400	50	0.125000
Reading Railroad	0.02908	200	25	0.125000
Short Line	0.02442	200	25	0.125000
Park Place	0.02122	350	35	0.100000
Pennsylvania Avenue	0.02457	320	28	0.087500
North Carolina Avenue	0.02561	300	26	0.086667
Pacific Avenue	0.02681	300	26	0.086667