Analyzing Monopoly via Simulation

Bryce Samwel

6/2/2020

Introduction

Monopoly is one of the most popular household board games of all time. It is also known to be one of the most frustrating games to play as well! However, by simulating dice rolls and following the rules of the game, this analysis will prove that not all color sets were created equal, and hopefully provide you with some useful guidance on which color sets can give you a slight advantage over your opponents!

Simulation Walkthrough

Before we do anything else, we need to first define the spaces on the board, the community chest and chance cards, as well as the location of our player. For the purposes of this model, I didn't feel the need to have multiple players and track their money and purchases. Really, we are only interested in how often each space gets landed on.

```
spaces <- numeric(40) #Vector of all spaces on the board
loc <- 1 #Location of the player
cc <- c(1:16) #Community Chest
ch <- c(1:16) #Chance
consecutive_doubles <- 0
```

I define "GO" as space #1 and Boardwalk as space #40 in this simulation. Using that definition, I can count the number of times each space has been landed on and compile that into the "spaces" vector. Additionally, I define two separate dice rolls in order to track the number of times the player has rolled consecutive doubles. This is important because when a player rolls doubles three times in a row they are sent to jail. It's also worth noting before jumping into the code that I don't just count where the player ends up at the end of their turn. For example, if the player lands on chance and gets sent to jail, I count the player as having visited that chance space as well as jail.

It's difficult to break a loop and talk about it in Markdown, so read the comments in the code for information about what I'm doing!

```
turns <- 1000000


for (t in 1:turns){
  roll1 <- sample(1:6,1,replace = TRUE) #First dice roll
  roll2 <- sample(1:6,1,replace = TRUE) #Second dice roll
  roll <- roll1 + roll2 #Sum of the two dice rolls
```

}

```
  #Check for consecutive doubles
  if (roll1 == roll2){
    consecutive_doubles <- consecutive_doubles + 1
  } else {
    consecutive_doubles <- 0
  }

  #Go to jail if rolling 3 consecutive doubles
  if (consecutive_doubles == 3){
    loc <- 11
  }
  else {
    loc <- loc + roll #Update location
  }

  if (loc > 40){
    loc <- loc - 40 #Make sure "loc" doesn't go above 40
  }

  spaces[loc] <- spaces[loc] + 1 #Update the number of times this spaces has
been landed on


  #Go To Jail
  if (loc == 31){
    loc <- 11
    spaces[loc] <- spaces[loc] + 1
  }


  #Community Chest
  if ((loc == 3)|(loc == 18)|(loc == 34)){
    ccDraw <- sample(cc,1)
    cc <- cc[!cc %in% ccDraw] #Remove ccDraw

    if (ccDraw == 14){ #Go to Jail
      loc <- 11
      spaces[loc] <- spaces[loc] + 1
    }
    else if (ccDraw == 15){ #Advance to Go
      loc <- 1
      spaces[loc] <- spaces[loc] + 1
    }

    if (sum(cc)==0){
      cc <- c(1:16) #Reshuffle the CC cards
    }
  }
```

```r
#Chance
if ((loc == 8)|(loc == 23)|(loc == 37)){
  chDraw <- sample(ch,1)
  ch <- ch[!ch %in% chDraw] #Remove chDraw

  if (chDraw == 1){ #Advance to Go
    loc <- 1
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 2){ #Take a Ride on Reading
    loc <- 6
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 3){ #Go Directly to Jail
    loc <- 11
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 4){ #Advance to St. Charles
    loc <- 12
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 5){ #Advance to Illinois
    loc <- 25
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 6){ #Advance Token to Boardwalk
    loc <- 40
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 7){ #Go Back 3 Spaces
    loc <- loc - 3
    spaces[loc] <- spaces[loc] + 1
  }
  else if (chDraw == 8){ #Nearest Utility
    if ((loc == 8)|(loc == 37)){
      loc <- 13 #Electric Company
      spaces[loc] <- spaces[loc] + 1
    }
    else {
      loc <- 29 #Water Works
      spaces[loc] <- spaces[loc] + 1
    }
  }
  else if ((chDraw == 9)|(chDraw == 10)){ #Nearest Railroad
    if (loc == 8){
      loc <- 16 #Pennsylvania
      spaces[loc] <- spaces[loc] + 1
    }
```

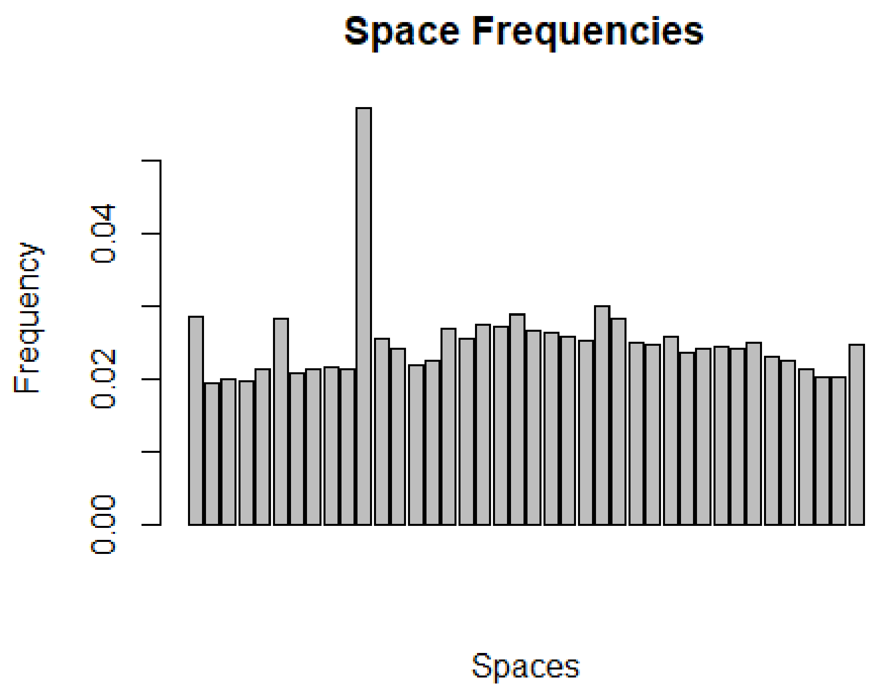
```
      else  if (loc == 23){
        loc <- 26 #B. & O.
        spaces[loc] <- spaces[loc] + 1
      }
      else {
        loc <- 6 #Reading
        spaces[loc] <- spaces[loc] + 1
      }
    }


    if (sum(ch)==0){
      ch <- c(1:16) #Reshuffle the Ch cards
    }
  }
}
```

The main thing worth talking about in this simulation that was tricky for me to figure out was how to account for the player drawing community chest and chance cards. What I chose to do was create vectors numbered one through sixteen. I would then randomly sample a single number from that vector to determine which card was drawn. Then, I would remove that number from the vector. Once the vector was empty, I would redifine it as 1:16 again. So, instead of shuffling the numbers and drawing from the top, it's as if players are picking from a pile of cards at random instead. This process is effectively the same as shuffling the cards.

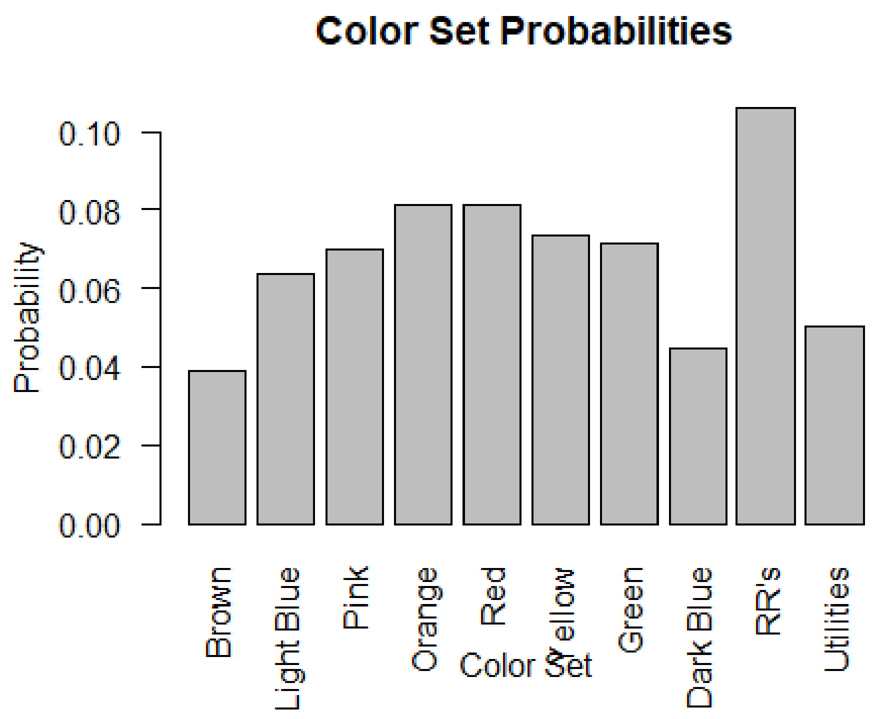
Results and Analysis

The first thing we are interested in knowing is how often each space gets landed on. So, we normalize the number of times each space is landed on in order to get frequencies and then plot them.



Without even referencing the board, can you guess what that big spike is? Yep, that's jail. There is not only a space on the board that sends you to jail, but there are also two cards, and the consecutive doubles rule. It's no surprise that this is the most visited space. The other two spaces that stand out compared to their neighbors are GO and Reading Railroad. These are also likely a result of the community chest and chance cards. It's also interesting to note that the frequencies are slightly higher toward the "center" of the board, near the orange and red spaces. My assumption is that these spaces are visited more frequently because of how often people are sent to jail and their proximity to it.

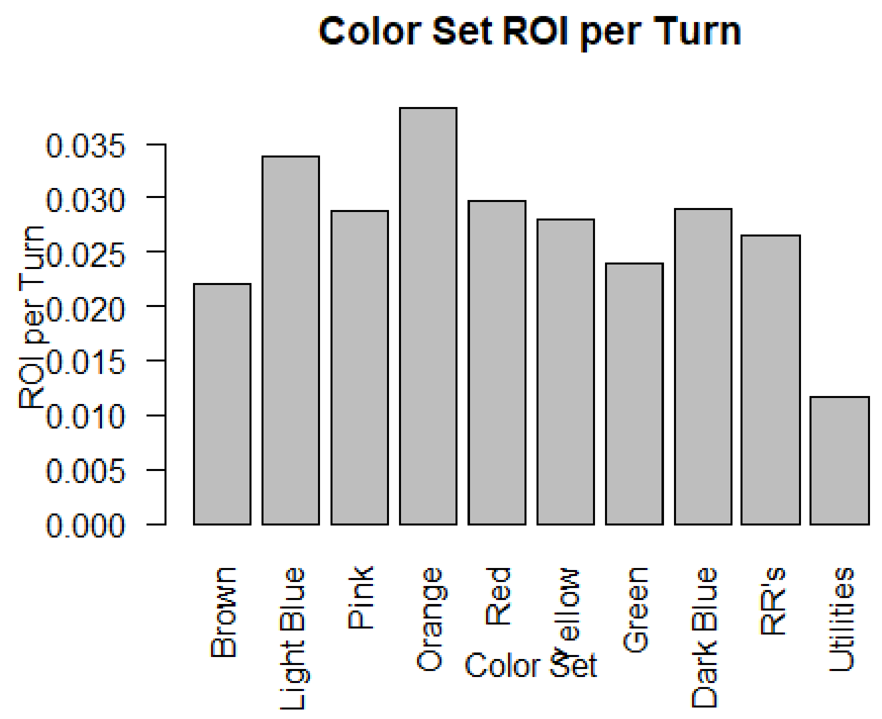
Nontheless, it appears as though there aren't any specific spaces that are very significantly better than the others, and it also isn't as useful to look at individual spaces. Everybody knows that in order to win the game, you need to make monopolies. Therefore, it's helpful to look at the color sets instead of each space separately. So, adding the frequencies of the spaces that make up each set (including the railroads and utilities) we get the following graph:



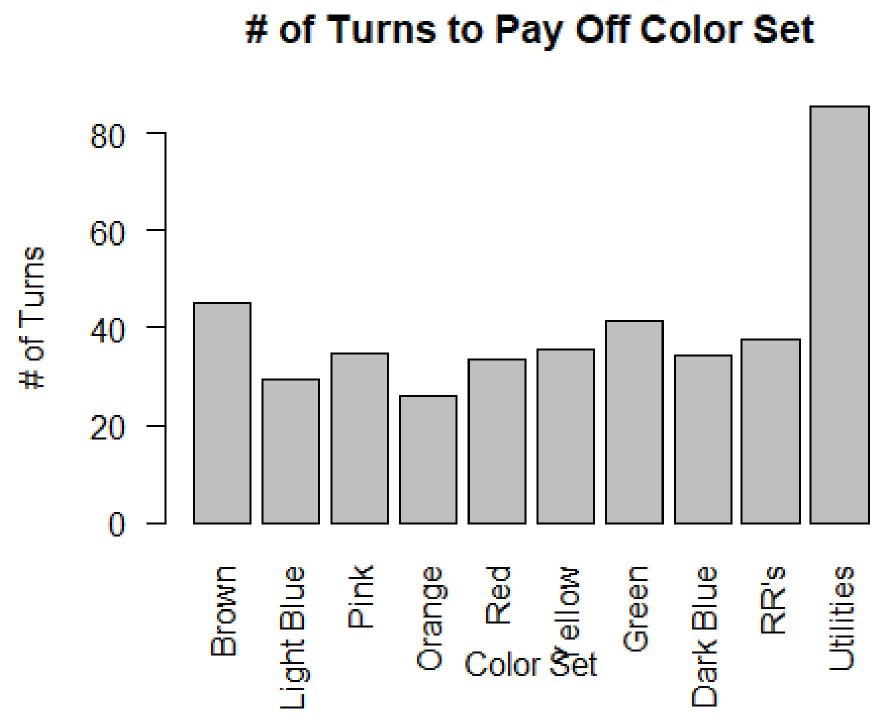
It appears as though the railroads benefit from being the only set with four separate spaces that make them up. Additionally, we continue to see that the central color sets, Orange and Red, are landed on with the highest frequency. Also, due to their limited number of spaces, Brown, Dark Blue, and Utilities are on the lower end. However, this doesn't exactly mean that these spaces are useless! In order to truly understand how valuable a space is, it's important to know how often it is landed on as well as how much money you can make each time it is. Doing this isn't very straight forward as it costs varying amounts of money to place houses and hotels and the player gets differing returns based on how many they have. To plot a set's ability to make money, I chose to measure the return on investment for sets with all hotels. It's true that the player will likely make some money on their set from rent before they make it to having hotels, but this measure is still useful when considering end-game strategy. So, ROI per turn for each set will be calculated as

$$\sum_{n=1}^{N} \frac{HotelRent(n)}{TotalBuildingCost} Freq(n)$$

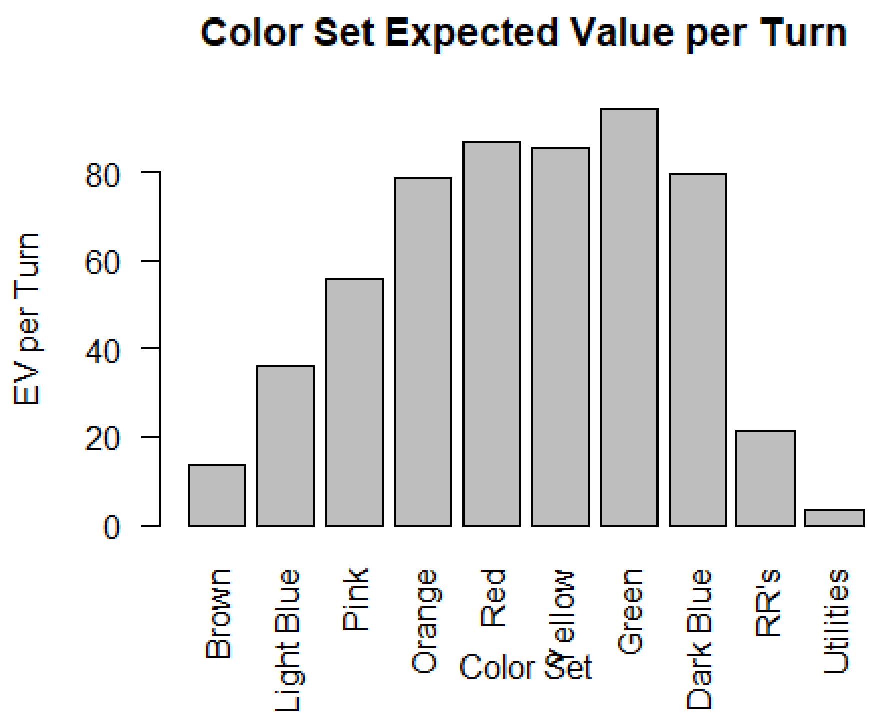
where N is the number of properties in a color set, HotelRent is the rent for a given property with a hotel, TotalBuildingCost is the cost to buy all of the properties and pay for each of the houses and hotel,and Freq is the frequency of a given property being landed on. This is essentially an expected value for the return each turn divided by the investment.



Based on this measuremnt, it appears as though Orange would be the first property to pay itself off, with Light Blue coming in second. To make this more clear, we can also calculate the number of turns required to pay off a color set:



There is one downside to this though. Time to repay doesn't cosider how much could be made into the future. It would be smart to also look at the expected value of a given color set per turn without dividing it by the cost to build everything.



Looking at it this way, we can see that the Green spaces provide nearly 100 monopoly dollars per turn, whereas the Utilities are making below ten!

Discussion and Conclusion

Based on the analysis of our simulation, it would appear as though there isn't a major difference between color sets, but the Orange and Light Blue pay themselves off the fastest, the Green and Red have the highest expected value per turn and the Utilities SUCK. In terms of strategy, succeeding in the early game would depend on your ability to spend your money wisely. Therefore color sets with higher ROI's are good for the early game. However, in the late game when most of the people left playing have hotels on their properties, the color sets with higher expected values are much better. Based on those assumptions, it would seem as though Red and Orange are the ideal color sets because they pay themselves off fairly quickly and continue to pay nicely throughout the rest of the game. If you can get it early on, Light Blues are good to shoot for. If you want to win though, it's important to have some late game winners like Green, Red, Yellow, Orange, and Dark Blue. It's also safe to say that it's not worth going out of your way to get the Brown or Utility sets.

As I mentioned before, this isn't a perfect analysis as we didn't look into how the ROI evolves over the number of houses on eahc property, but performing an analysis based on the returns with Hotels helps to give an idea of what each color set will provide toward the later ends of the game. I hope this was interesting and that you can take these recommendations into a real game and see how well it works for you!