# Stats 102A - Homework 4 Instructions: Monopoly



## Homework 4 Requirements

You will submit three files.

The files you submit will be:

1. `102a_hw_04_script_First_Last.R` Your R script file containing the functions you write for the homework assignment. Write comments as necessary. Use the starter_code file as a starting point for the script. Upload the R script to Canvas.
2. `102a_hw_04_output_First_Last.Rmd` Take the provided R Markdown file and make the necessary edits so that it generates the requested output. The first line of your .Rmd file should be to source the R script file you wrote. Upload the Rmd file to Canvas.
3. `102a_hw_04_output_First_Last.pdf` Your output PDF file. This is the primary file that will be graded. Make sure all requested output is visible in the output file. Upload the PDF to Gradescope.

### Academic Integrity

At the top of your R markdown file, be sure to include the academic integrity statement.

### Recommended Reading:

a. S3 (not necessary for this HW): https://adv-r.hadley.nz/s3.html
b. R6: https://adv-r.hadley.nz/r6.html

## Monopoly Board game simulation

## Read through all of the instructions before getting started.

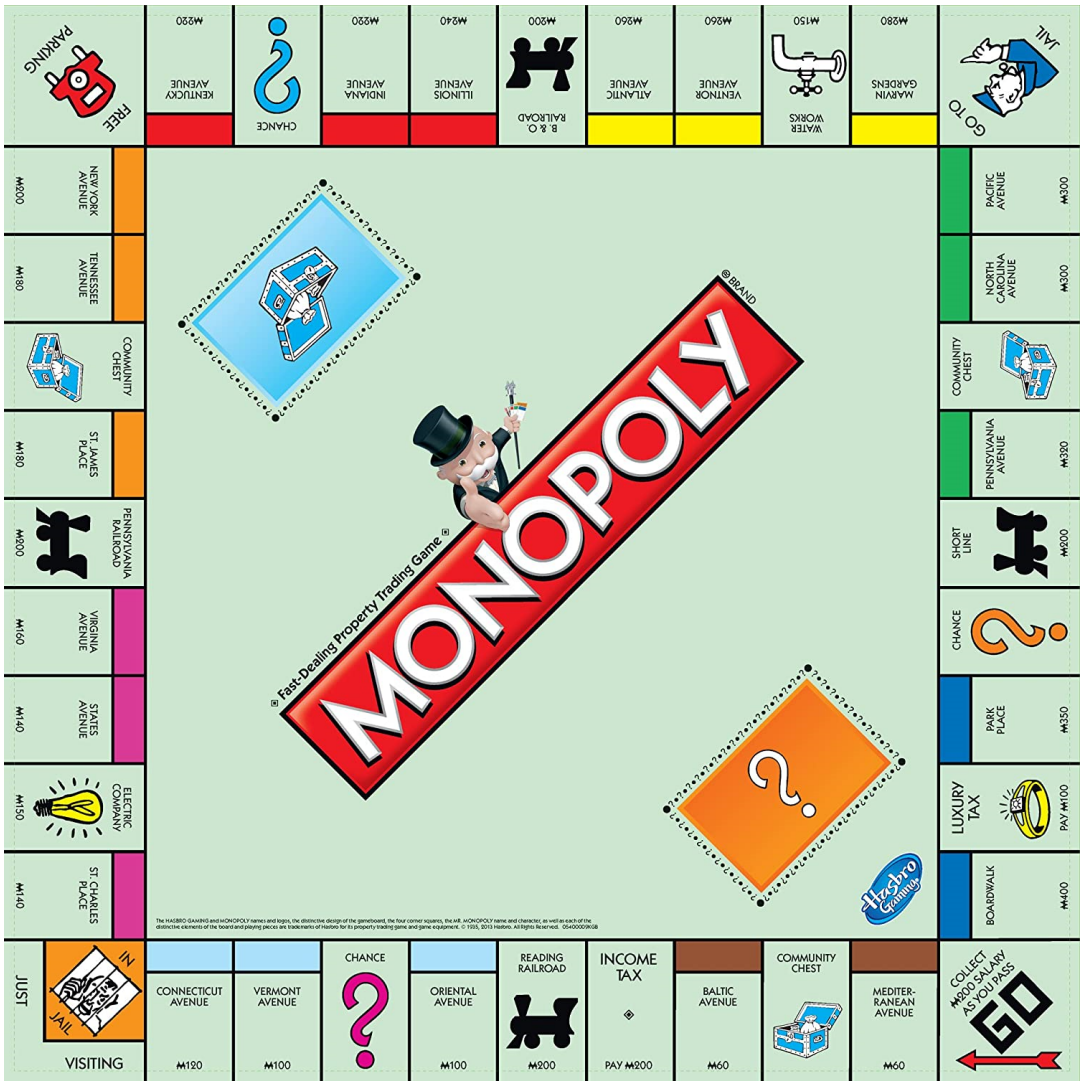
### For this homework assignment, you will create a simulation of the classic board game, Monopoly. The goal is to find out which spaces on the board get landed on the most.

You will not simulate the entire game. You will simulate **only the movement** of pieces, and will keep track of which squares the pieces land on.

Official rules https://www.hasbro.com/common/instruct/monins.pdf

Familiarize yourself with the game board. (Taken from Amazon's product page.)

```
include_graphics('board.png')
```




# Rules for movement

The Monopoly Board is effectively a circle with 40 spaces on which a player can land. Players move from space to space around the board in a circle (square).

The number of spaces a player moves is determined by the roll of 2 dice. Most often, the player will roll the dice, move the number of spaces shown on the dice, land on a space, and end their turn there. If this were the entire game, there would be an equal probability of landing on all 40 spaces after many turns - and the distribution of space frequency would be uniform.

There are, however, several rules which create variation in the frequency of landing on different spaces.

## Go to Jail

One space, "Go to Jail" sends players directly to jail (there is a jail space on the board). This space never counts as having been landed upon. As soon as the player 'lands' here, he is immediately sent to jail, and the jail space gets counted as landed upon. This is the only space on the game board that moves a player's piece. The count of how often this space is landed on will always be 0.

## Rolling Doubles

If a player rolls doubles (two of the same number), the player moves his piece, and then gets to roll the dice again for another move. However, if a player rolls doubles three times in a row, he is sent directly to jail. (The third space

that the player would have 'landed on' does not count, but the jail space gets counted as landed on.)

## Card Decks: Chance and Community Chest

A player can land on a "Chance" or "Community Chest" space. When a player lands on these spaces, he draws a card from the respective deck and follows its instructions. The instructions will sometimes give money to or take money from the player with no change in the player's position on the board. Other times, the card will instruct the player to move to another space on the board. The list of cards that can be drawn from each deck is provided.

There are nine cards in the Chance deck that move the player's token. There are two cards in the Community Chest deck that move the player's token. All other cards do not move the player's token. For the sake of this simulation, you only need to program actions for the cards that move the tokens. There is no need to do anything for 'get out of jail' or any of the other cards.

A card may say 'move to the nearest railroad' or 'move to the nearest utility' or even 'go to property . . . '. In these cases, the player always moves forward. So if a player is on 'Oriental Avenue,' the nearest railroad is 'Pennsylvania Railroad' and NOT 'Reading Railroad.'

For the sake of this simulation, the Chance and Community Chest **get counted as landed on** when the player lands on the Chance or Community Chest space. The player may also generate another count if the card moves the player to another space on the board. In those cases, a tally is counted for the Chance/Community Chest space, the token is moved, and then a tally is counted for the space where the player ends his turn.

## Jail

Jail is the most complicated aspect of this simulation.

If a player lands *on* space 11 (Jail) simply from rolling the dice, he is not in Jail. He is 'just visiting' jail. He generates a tally for landing *on* jail, and his play continues on as normal.

A player can be *sent to jail* in several ways:

- he rolls doubles three times in a row;
- he lands on the "go to jail" space;
- he draws a card that sends hims to jail.

As soon as the player is *sent to jail*, his token moves to jail (space 11), he generates a count for landing on jail, and his turn ends immediately.

On the next turn, the player begins in jail and the player will roll the dice. If he rolls doubles on the dice, he gets out of jail and moves the number of spaces the dice show. However, even though he rolled doubles, he does NOT roll again. He takes his move out of jail and his turn ends. If he does not roll doubles, he stays in jail.

A player cannot stay in jail for more than three turns. On the third turn he begins in jail, he rolls the dice and moves the number of spaces the dice show no matter what. If he rolls doubles, he exits but does not roll again. If he does not roll doubles, he still exits and does not roll again.

Play then continues as normal.

For this simulation, each time a player ends his turn in Jail, a tally will be counted as having been 'landed upon.'

There are more rules on jail that include paying a fee to get out early, or using a get out of jail free card. We will not implement those rules. We will simply simulate a 'long stay' strategy for Jail. This means that the player will never pay the fee to get out jail early. He will roll the dice and only leave jail if he gets doubles or it is his third turn in jail.

## Space Definitions:

- Utilities are Electric Company and Water Works
- Railroads are: Reading Railroad, Pennsylvania Railroad, B & O Railroad, Shortline Railroad

# The Assignment

Your assignment is to implement the rules of Monopoly movement.

You must use R6 to create an object class `Player` which will be used to keep track of a player.

## Part 1

You will first demonstrate that you have coded the rules by showing the output of several test cases using **preset** dice. Your output for this section should match the published results exactly.

The output should be very verbose. It should announce the player roles, where the player moves, what spaces get tallies, if they rolled doubles, etc. Your text output should match as closely as possible.

## Part 2

The next part is to run 1,000 simulations of a two-player game that lasts 150 turns. This is a total of over 3 hundred thousand tosses of the dice - 1000 games x 150 turns x 2 players + additional rolls if the player gets doubles.

Your task is to keep track of where the players land. We ultimately want to build a distribution showing which spaces are most likely to be landed upon. Advance the tokens around the board according to the rules. Keep in mind the special situations involving the cards, jail, and rolling doubles. After 150 turns, reset the game and start over. Simulate 1000 games.

Your final output will be two tables of the spaces on the board and their frequencies. Each table will show the space name, how many times the space was landed upon, and the relative frequency of landing on that space.

The first table is arranged in descending order of frequency of landing. (Jail should be #1, Go to jail should be last.)

The second table is arrange in the order of the spaces on the board. (Go will be #1, Boardwalk will be last.)

Also print a bar graph showing the frequency of how often each space is landed on.

You do not have to simulate or track money at all in this simulation.

## Starter Code

For your convenience, I have created the necessary data frames for the game board, and the two decks of cards.

I have also created several helper R6 classes for you:

- `PresetDice`: You will use this for Part 1: the test cases
- `RandomDice`: You will use this for Part 2: the simulation of 1000 games.
- `CardDeck`: For creating the Chance and Community Chest decks. This reference class 'shuffles' the deck, and each time a player draws a card, it shows the next card drawn. When all the cards in the deck have been used, it 'shuffles' the deck again.
- `SpaceTracker`: For keeping track of all the spaces you land on.

Please take the time to read through these R6 class definitions. Understanding the definitions and what the methods do will be necessary for completing the assignment.

### Tips

At first blush, the task may seem overwhelming.

- Break the task into smaller manageable parts.
- Start with a simulation that moves pieces around the board and keeps track of where they land. (I've done this part for you in my example code.)
- Then add complexity one part at a time. Each time you add something, thoroughly test it to make sure it behaves the way you want it to.
- The PresetDice reference class will be very helpful in your testing. For example, you can intentionally create some dice rolls to test certain situations. Want to test if Chance is working correctly? Set your dice to cause the player to land on Chance. Want to test if doubles is working correctly? Set the PresetDice to produce

doubles. If you had purely random dice, you might have to wait for the computer to produce many many random outcomes before you see three doubles in a row.

My recommendation in terms of adding complexity:

- Add code so landing on "Go to jail" sends the player to jail.
- Add code that draws from the Chance and Community Chest decks and moves the player accordingly. Keep in mind that some cards have no effect on player movement, while other cards do.
- Add code to allow players to roll again after doubles but goes to jail if rolling doubles three times in a row.
- Add code to implement the rules for Jail. You'll need to keep track of whether the player is actually in jail or not, how many turns the player has been in jail, and the rules for getting out.

## Grading

1. Do NOT print the verbose version for all of 1000 games.
2. We will not run your code. The simulation will take time to run, and we do not have the luxury of running the entire simulation for all students.
3. We will check the output of the test cases.

These are the point values of each test case.

- Test Case 1: 5 points.
  - Turn 1: Player must be sent to jail.
  - Tally at 11 only.
- Test Case 2: 15 points.
  - Turn 1: Player ends at Go.
  - Turn 2: Player ends at Reading Railroad.
  - Turn 3: Player goes to Pennsylvania Railroad, goes again, player ends at Water Works.
  - Turn 4: Player ends on Chance. No movement.
  - Tally at: 1, 6, 8 (3), 16, 23, 29, 37
  - If your Chance cards are not matching mine, make sure your R is up to date and that you are not loading a saved work space on start up.
- Test Case 3: 10 points.
  - Turn 1: Player rolls the dice a total of three times and ends at States Ave.
  - Turn 2: Player lands on Community Chest. Does not move.
  - Tally at: 7, 11, 14, 18
- Test Case 4: 10 points.
  - Turn 1: Player ends up in Jail.
  - Turn 2: Player stays in Jail
  - Turn 3: Player stays in Jail
  - Turn 4: Player exits Jail and lands on Kentucky ave.
  - Tally at: 7, 11 (3), 13, 22
- Test Case 5: 10 points.
  - Turn 1: Player ends in Jail. Player does not roll again, despite having rolled doubles.
  - Turn 2: Player stays in Jail.
  - Turn 3: Player rolls doubles and exits Jail. Player does not roll again, despite having rolled doubles.
  - Tally at: 11 (2), 17
- Test Case 6: 10 points.
  - Output must match the example output

That is 60 points.

4. We will check the final output of counts in the results tables.

Things we are looking for in the results tables (5 points each):

- Jail should be the most frequent space landed on.
- Jail should have a frequency between 10 and 13%.
- Spots 2 and 3 should be Illinois Ave and Go (possibly switched)
- New York Ave. and Tennessee Ave. should be in the top 10.

- Reading Railroad and B&O Railroad should be in the top 10.
- Mediterranean Ave and Baltic Ave should be in the bottom 5.
- Park Place should be very infrequent. Bottom 10.
- Go to jail should be landed on 0 times.

5. If your test cases and table output matches what we are looking for, you will get full credit.

You can 'hard code' the functions that handle the decks. In other words, you can write something along the lines of

```
## for chance deck
...
    if(carddrawn == 1)
        code changes player position to space 1  # advance to go
    if(carddrawn == 2)
        code changes player position to space 25 # advance to Illinois avenue
    # etc.
...
```

**Good luck!**

I know this is a tough assignment. Like everything else in life, sometimes you have to prioritize other things (eg. health, sleep, sanity, etc.) over the task at hand.

If you are unable to implement all parts of the solution, that is also okay. I cannot give you full credit, but please indicate what you were able to implement and what you were not able to implement. You will be graded on what you were able to complete.