Monopoly Simulation

Mizuki Kio

2016/11/21

Monopoly Board game simulation

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 be simulating the entire game. You will be simulating only the movement of pieces, and will keep track of which squares the pieces land on. If you have never played Monopoly before, I recommend watching a few videos on the topic. https://www.youtube.com/watch?v=4Hfe97Q5kuI (https://www.youtube.com/watch?v=4Hfe97Q5kuI)

You can also familiarize yourself with the game board. (Taken from Amazon's product page.)

http://ecx.images-amazon.com/images/I/81oC5pYhh2L._SL1500_.jpg (http://ecx.images-amazon.com/images/I/81oC5pYhh2L._SL1500_.jpg)

Official rules http://www.hasbro.com/common/instruct/00009.pdf (http://www.hasbro.com/common/instruct/00009.pdf)

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, land on a space, and end his turn there. If this were the entire game, the spaces would have a uniform distribution.

There are, however, several exceptions which provide the primary source of variation in space landing

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 below.

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 worry about the cards that move the tokens.

A card may say ‘move to the nearest railroad’ or ‘move to the nearest utility’ or even ‘go to property xxx’. 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.’

The Chance and Community Chest spaces always get counted as “landed on” even if the card drawn moves the player to another space or sends him to jail. 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), he is not in Jail. He is ‘just visiting.’ His play continues on as normal.

A player can be placed in 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.

When in jail, the player has the option to pay a fee to ‘get out,’ or he can choose not to pay the fee. If he pays the fee, he is out of jail, and his play continues normally as before. If he chooses not to pay the fee, he rolls 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 his third turn in jail, he rolls the dice and moves the number of spaces the dice show no matter what. If they are doubles, he moves those spaces for free. If he does not roll doubles, he moves those spaces, but must also pay a fee.

Play then continues as normal.

More rules on jail: http://monopoly.wikia.com/wiki/Jail (http://monopoly.wikia.com/wiki/Jail)

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

We will simulate a ‘long stay’ strategy for Jail. This effectively means that the player will never choose to pay the fee to get out jail unless forced to do so. Effectively, this means that he will roll the dice and only leave jail if he gets doubles or it is his third turn in jail.

Your Simulation

Your task is to run 1,000 simulations of a two-player game that lasts 150 turns. This is a total of over 6 hundred thousand dice rolls - 1000 games x 150 turns x 2 players x 2 dice + 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 should be a table of the spaces on the board, how many times the space was landed upon, and the relative frequency of landing on that space. Arrange the table in descending order of frequency of landing.

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

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

```
gameboard <- data.frame(space = 1:40, title = c("Go" , "Mediterranean Avenue" , "Community Ches
t" , "Baltic Avenue" , "Income Tax" , "Reading Railroad" , "Oriental Avenue" , "Chance" , "Vermo
nt Avenue" , "Connecticut Avenue" , "Jail" , "St. Charles Place" , "Electric Company" , "States
 Avenue" , "Virginia Avenue" , "Pennsylvania Railroad" , "St. James Place" , "Community Chest" ,
 "Tennessee Avenue" , "New York Avenue" , "Free Parking" , "Kentucky Avenue" , "Chance" , "India
na Avenue" , "Illinois Avenue" , "B & O Railroad" , "Atlantic Avenue" , "Ventnor Avenue" , "Wate
r Works" , "Marvin Gardens" , "Go to jail" , "Pacific Avenue" , "North Carolina Avenue" , "Commu
nity Chest" , "Pennsylvania Avenue" , "Short Line Railroad" , "Chance" , "Park Place" , "Luxury
 Tax" , "Boardwalk"))

chancedeck <- data.frame(index = 1:15, card = c("Advance to Go" , "Advance to Illinois Ave." ,
"Advance to St. Charles Place" , "Advance token to nearest Utility" , "Advance token to the near
est Railroad" , "Take a ride on the Reading Railroad" , "Take a walk on the Boardwalk" , "Go to
 Jail" , "Go Back 3 Spaces" , "Bank pays you dividend of $50" , "Get out of Jail Free" , "Make g
eneral repairs on all your property" , "Pay poor tax of $15" , "You have been elected Chairman o
f the Board" , "Your building loan matures"))

communitydeck <- data.frame(index = 1:16, card = c("Advance to Go" , "Go to Jail" , "Bank error
 in your favor. Collect $200" , "Doctor's fees Pay $50" , "From sale of stock you get $45" , "Ge
t Out of Jail Free" , "Grand Opera Night Opening" , "Xmas Fund matures" , "Income tax refund" ,
"Life insurance matures. Collect $100" , "Pay hospital fees of $100" , "Pay school tax of $150"
 , "Receive for services $25" , "You are assessed for street repairs" , "You have won second pri
ze in a beauty contest" , "You inherit $100"))
```

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.
...
```

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.

- Add something so landing on “Go to jail” sends the player to jail.
- Add functions for the Chance and Community Chest decks. Keep in mind that some cards have no effect on player movement, while other cards do.
- Add something to allow players to move again after doubles.
- Finally implement the 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.

Requirements for grading

0. Do NOT print the verbose version for all of 1000 games.
1. Your final output should be a table of the spaces on the board, how many times the space was landed upon, and the relative frequency of landing on that space. Print this table two times. First, arrange the table in descending order of frequency of landing. Second, arrange the table in the order of the spaces on the board.
    a. While we do not expect your results to match perfectly with ours, we will check for certain outcomes.
    b. For example, “Go to Jail” should have a frequency of 0.
    c. “Jail” should be the most frequently landed on space.
    d. You will be graded on whether your results table matches the established trends of the board.
2. You will also be graded on the functions that you have implemented in your code.
    a. In your code, create sections to clearly show the functionality that you are implementing.
    b. You can achieve this in R studio by using the menu Code > Insert Section
    c. If you are implementing multiple functionalities in one function of the code, that is fine, but be sure to clearly indicate the different sections that are implemented in the code.

```
# dice ----------------------------------------------------------------

dice <- function()...

# drawing chance card -------------------------------------------------

etc. etc.
```

3. We will check for the following sections. You can have more, but proper implementation of these sections is required for full credit. Because your code will be quite long, it is *VERY IMPORTANT* you *CLEARLY* mark these sections for the grader.
    a. drawing chance card
    b. drawing community chest card
    c. landing on “go to jail”
    d. roll again for rolling doubles
    e. going to jail for rolling three doubles
    f. jail functionality
4. To see if you fully implemented the code correctly, I am giving a set of rolls that I want everyone to use for 20 turns of one player. If you follow, the rules, where you land should be deterministic. On the 18th turn, you will land on a chance space. This will be your first random event (with my seed and implementation, my first chance card sends me to jail). We’ll check to see if your code follows the monopoly rules for these 20 turns. Be sure to print out your `space_tracking$tally` vector for verification.

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.

You are encouraged to work with other students currently enrolled in Stats 102A. You are not allowed to seek out past solutions from previous classes.

Best wishes!

```
## Write your code here

#Switch for verbose & manual_mode
verbose_switch <- TRUE #turn this off before running simulation
manual_switch <- TRUE #turn this off before running simulation

# Setting Reference Classes  ----------------------------------------------------------------
-
##player class
player <- setRefClass("player",
            fields=list(id="character",
                        pos="numeric",
                        verbose="logical",
                        n_jail="numeric"),
            methods=list(
              #Method to move by n spaces
              move_n = function(n) {
                if(verbose) {cat("Player",id,"at:", pos, ",", as.character(gameboard$title[pos])
,"\n")}
                if(verbose) cat("Player",id,"moves:", n, "\n")
                pos <<- pos + n
                if(pos > 40) pos <<- pos - 40
                if(verbose) cat("Player",id,"now at:", pos, ",", as.character(gameboard$title[po
s]),".\n")
              },
              #Method to move to space n
              go_2_space_n = function(n){
                if(verbose) cat("Player",id,"at:", pos, ",", as.character(gameboard$title[pos]),
".\n")
                pos <<- n
                if(verbose) cat("Player",id,"now at:", pos, ",", as.character(gameboard$title[po
s]),".\n")
              },
              #Method to get out of jail
              out_jail = function() {n_jail <<- 0}
            )
)

player1 <- player$new(id="A", pos=1, verbose=verbose_switch, n_jail=0)
player2 <- player$new(id="B", pos=1, verbose=verbose_switch, n_jail=0)

##tracking class
tracking <- setRefClass("tracking",
  fields = list(tally = "numeric", verbose="logical"),
  methods = list(
    increase_count = function(n){
      if(verbose){cat("Tally at",n,":",as.character(gameboard$title[n]),".\n")}
      tally[n] <<- tally[n] + 1
    }
  )
)

space_tracking <- tracking$new(tally = rep(0,40), verbose=verbose_switch)
```

```
#reset function: reset player positions and tracking info
reset_game <- function(reset_player=TRUE,reset_tally=FALSE) {
  if(reset_player) {
    player1$pos <- 1
    player1$n_jail <- 0
    player2$pos <- 1
    player2$n_jail <- 0
  }
  if(reset_tally) {space_tracking$tally <- rep(0,40)}
}

# Manual Dice ------------------------------------------------------------
Dice = setRefClass("Dice",
                   fields = list(
                     rolls = "numeric",
                     pos = "numeric",
                     verbose = "logical"
                   ),
                   methods = list(
                     roll = function() {
                       faces = rolls[pos + seq_len(2)]
                       pos <<- pos + 2
                       if(faces[1] == faces[2]) doubles = TRUE
                       else doubles = FALSE
                       movement = sum(faces)
                       if(verbose) cat("Rolled:", faces[1], faces[2], "\n")
                       return(list(faces=faces, doubles=doubles, movement=movement))
                     }
                   )
)

#Deterministic manual dice
setdice <- Dice$new(rolls = c(6, 4, 5, 3, 3, 5, 6, 2, 5, 4, 4, 1, 2, 6, 4, 4, 4, 4, 2, 2, 4, 3,
4, 4, 1, 4, 3, 4, 1, 2, 3, 6, 5, 4, 5, 5, 1, 2, 5, 4, 3, 3, 1, 1, 2, 1, 1, 3), pos = 0, verbose
= verbose_switch)
manual_dice <- function() setdice$roll()


# Random Dice ------------------------------------------------------------

dice <- function(verbose=verbose_switch){
  faces <- sample(1:6, 2, replace=TRUE)
  if(faces[1] == faces[2]) doubles = TRUE
  else doubles = FALSE
  movement = sum(faces)
  if(verbose) cat("Rolled:", faces[1], faces[2], "\n")
  return(list(faces=faces, doubles=doubles, movement=movement))
}

# go2Jail function -------------------------------------------------------

go2jail <- function(player,tracking=space_tracking,verbose=verbose_switch) {
  if(verbose) {cat("Going to Jail.\n")}
```

```
  player$go_2_space_n(11)
  player$n_jail=1
  tracking$increase_count(player$pos)
}

# drawing chance card ------------------------------------------------

chance <- function(player,tracking=space_tracking,verbose=verbose_switch){
  if(verbose) {cat("Draw chance card.\n")}
  k <- sample(1:15,1)
  if(verbose) {cat("Card Drawn:", as.character(chancedeck$card[k]),"\n")}
  if(k==1) {player$go_2_space_n(1)} #Advance to Go
  if(k==2) {player$go_2_space_n(25)} #Advance to Illinois Ave.
  if(k==3) {player$go_2_space_n(12)} #Advance to St. Charles Place
  if(k==4) { #Advance to nearest Utility
    if(player$pos %in% c(8,37)) {player$go_2_space_n(13)} #Go to electric company
    if(player$pos==23) {player$go_2_space_n(29)} #Go to water works
  }
  if(k==5) { #Advance to nearest Railroad
    if(player$pos==8) {player$go_2_space_n(16)} #Go to Pennsylvania Railroad
    if(player$pos==23) {player$go_2_space_n(26)} #Go to B & O Railroad
    if(player$pos==37) {player$go_2_space_n(6)} #Go to Reading Railroad
  }
  if(k==6) {player$go_2_space_n(6)} #Take a ride on the Reading Railroad
  if(k==7) {player$go_2_space_n(40)} #Take a walk on the Broadwalk
  if(k==8) {go2jail(player)} #Go to Jail
  if(k==9) {player$move_n(-3)} #Move back 3 spaces
  #if(k %in% 10:15) DO NOTHING

  if(k %in% c(1:7,9)) {tracking$increase_count(player$pos)} #tally at updated position.
  #Do not tally if k=8 because go2jail function handles tally
  #Do not tally if k %in% 10:15 because there is no movement

} #End of chance function

# drawing community chest card ---------------------------------------

chest <- function(player, tracking=space_tracking, verbose=verbose_switch) {
  if(verbose) {cat("Draw community chest card.\n")}
  k <- sample(1:16,1)
  if(verbose) {cat("Card Drawn:", as.character(communitydeck$card[k]),"\n")}
  if(k==1) { #Advance to Go
    player$go_2_space_n(1)
    tracking$increase_count(player$pos)}
  if(k==2) {go2jail(player)} #Go to Jail. go2jail will handle tally.
  #for all other cards, do nothing
}

# taking turn ------------------------------------------------------------

n_double <- 0

taketurn <- function(player, tracking=space_tracking, manual_mode=manual_switch) {
  ###Check if player is in jail###
```

```
if(player$n_jail>0) {
  # jail functionality ------------------------------------------------
  if(verbose_switch) {cat("Number of turns in jail:",player$n_jail,"\n")}
  #roll dice
  if(manual_mode) {roll <- manual_dice()}
  else {roll <- dice(verbose=verbose_switch)}
  #check jail_count
  if(player$n_jail==3) {
    #3rd turn in jail, get out no matter what
    if(verbose_switch) {cat("Third turn in jail. Get out of jail.\n")}
    player$out_jail()
    player$move_n(roll$movement)
    tracking$increase_count(player$pos)
  }
  else {
    #1st or 2nd turn in jail, check for doubles
    if(roll$doubles) {
      #rolled double, get out of jail
      if(verbose_switch) {cat("Rolled doubles. Get out of jail.\n")}
      player$out_jail()
      player$move_n(roll$movement)
      tracking$increase_count(player$pos)
    }
    else {
      #did not roll double, stay in jail
      if(verbose_switch) {cat("Stay in jail.\n")}
      player$n_jail = player$n_jail + 1
      tracking$increase_count(player$pos)
    }
  }
} # End of jail functionality
else{
  # everything below: if(NOT in jail)
    #roll dice first
    if(manual_mode) {roll <- manual_dice()}
    else {roll <- dice(verbose=verbose_switch)}
    #if roll is a double
    if(roll$doubles) {
      n_double <<- n_double+1
      if(verbose_switch) {cat("Double count is:",n_double,".\n")}
      # going to jail for rolling 3 doubles ----------------------------
      if(n_double==3) {
        n_double <<- 0
        go2jail(player,tracking,verbose_switch)
        return() #force turn to end
      }
      #if n_double<3, execute the "regular" move first
    } #end of if(roll$doubles)

  #the "regular" movement:

    #1.update position
    player$move_n(roll$movement)
```

```
      #2.check "Go to jail"
      # Landing on "Going to Jail" -----------------------------------
      if(player$pos==31) {
        n_double <<- 0
        go2jail(player,tracking,verbose_switch)
        return() #force turn to end
      }

      #any space besides "Go to jail" can get a tally
      tracking$increase_count(player$pos)

      #3.chance or community chest
      if(player$pos %in% c(8,23,37)) {
        chance(player,tracking,verbose_switch)
      }
      if(player$pos %in% c(3,18,34)) {
        chest(player,tracking,verbose_switch)
      }

    # roll again for rolling doubles -----------------------------------
      if(roll$doubles & player$n_jail==0) { #the second condition checks whether the player is s
ent to jail because of chance card or community chest
        if(verbose_switch) {cat("Roll dice again.\n")}
        taketurn(player,tracking,manual_switch)
        }
  ###End of turn. Reset n_double to 0.###
n_double <<- 0
  } ###End of if(NOT in jail)

} #End of taketurn function


####Function to turn all verbose in RefClass on/off###
#verbose_switch in global environment cannot be turned on/off using this function
turn_verbose_on <- function(on) {
  if(on) {
    player1$verbose=TRUE
    player2$verbose=TRUE
    space_tracking$verbose=TRUE
    setdice$verbose=TRUE
  }
  else {
    player1$verbose=FALSE
    player2$verbose=FALSE
    space_tracking$verbose=FALSE
    setdice$verbose=FALSE
  }
}
```

Use set 20 turns to test code

```
## Set 20 turns --------------------
verbose_switch <- TRUE
turn_verbose_on(TRUE)
manual_switch <- TRUE
reset_game(reset_player = TRUE, reset_tally = TRUE)

set.seed(10)
testplayer <- player$new(id="T", pos = 1, verbose = TRUE, n_jail=0)  # new players for each game
for(i in 1:20){ # 100 turns for each game
  cat("\n## Turn", i,"\n")
  taketurn(testplayer, space_tracking)
}
```

```
##
## ## Turn 1
## Rolled: 6 4
## Player T at: 1 , Go
## Player T moves: 10
## Player T now at: 11 , Jail .
## Tally at 11 : Jail .
##
## ## Turn 2
## Rolled: 5 3
## Player T at: 11 , Jail
## Player T moves: 8
## Player T now at: 19 , Tennessee Avenue .
## Tally at 19 : Tennessee Avenue .
##
## ## Turn 3
## Rolled: 3 5
## Player T at: 19 , Tennessee Avenue
## Player T moves: 8
## Player T now at: 27 , Atlantic Avenue .
## Tally at 27 : Atlantic Avenue .
##
## ## Turn 4
## Rolled: 6 2
## Player T at: 27 , Atlantic Avenue
## Player T moves: 8
## Player T now at: 35 , Pennsylvania Avenue .
## Tally at 35 : Pennsylvania Avenue .
##
## ## Turn 5
## Rolled: 5 4
## Player T at: 35 , Pennsylvania Avenue
## Player T moves: 9
## Player T now at: 4 , Baltic Avenue .
## Tally at 4 : Baltic Avenue .
##
## ## Turn 6
## Rolled: 4 1
## Player T at: 4 , Baltic Avenue
## Player T moves: 5
## Player T now at: 9 , Vermont Avenue .
## Tally at 9 : Vermont Avenue .
##
## ## Turn 7
## Rolled: 2 6
## Player T at: 9 , Vermont Avenue
## Player T moves: 8
## Player T now at: 17 , St. James Place .
## Tally at 17 : St. James Place .
##
## ## Turn 8
## Rolled: 4 4
## Double count is: 1 .
```

```
## Player T at: 17 , St. James Place
## Player T moves: 8
## Player T now at: 25 , Illinois Avenue .
## Tally at 25 : Illinois Avenue .
## Roll dice again.
## Rolled: 4 4
## Double count is: 2 .
## Player T at: 25 , Illinois Avenue
## Player T moves: 8
## Player T now at: 33 , North Carolina Avenue .
## Tally at 33 : North Carolina Avenue .
## Roll dice again.
## Rolled: 2 2
## Double count is: 3 .
## Going to Jail.
## Player T at: 33 , North Carolina Avenue .
## Player T now at: 11 , Jail .
## Tally at 11 : Jail .
##
## ## Turn 9
## Number of turns in jail: 1
## Rolled: 4 3
## Stay in jail.
## Tally at 11 : Jail .
##
## ## Turn 10
## Number of turns in jail: 2
## Rolled: 4 4
## Rolled doubles. Get out of jail.
## Player T at: 11 , Jail
## Player T moves: 8
## Player T now at: 19 , Tennessee Avenue .
## Tally at 19 : Tennessee Avenue .
##
## ## Turn 11
## Rolled: 1 4
## Player T at: 19 , Tennessee Avenue
## Player T moves: 5
## Player T now at: 24 , Indiana Avenue .
## Tally at 24 : Indiana Avenue .
##
## ## Turn 12
## Rolled: 3 4
## Player T at: 24 , Indiana Avenue
## Player T moves: 7
## Player T now at: 31 , Go to jail .
## Going to Jail.
## Player T at: 31 , Go to jail .
## Player T now at: 11 , Jail .
## Tally at 11 : Jail .
##
## ## Turn 13
## Number of turns in jail: 1
## Rolled: 1 2
```

```
## Stay in jail.
## Tally at 11 : Jail .
##
## ## Turn 14
## Number of turns in jail: 2
## Rolled: 3 6
## Stay in jail.
## Tally at 11 : Jail .
##
## ## Turn 15
## Number of turns in jail: 3
## Rolled: 5 4
## Third turn in jail. Get out of jail.
## Player T at: 11 , Jail
## Player T moves: 9
## Player T now at: 20 , New York Avenue .
## Tally at 20 : New York Avenue .
##
## ## Turn 16
## Rolled: 5 5
## Double count is: 1 .
## Player T at: 20 , New York Avenue
## Player T moves: 10
## Player T now at: 30 , Marvin Gardens .
## Tally at 30 : Marvin Gardens .
## Roll dice again.
## Rolled: 1 2
## Player T at: 30 , Marvin Gardens
## Player T moves: 3
## Player T now at: 33 , North Carolina Avenue .
## Tally at 33 : North Carolina Avenue .
##
## ## Turn 17
## Rolled: 5 4
## Player T at: 33 , North Carolina Avenue
## Player T moves: 9
## Player T now at: 2 , Mediterranean Avenue .
## Tally at 2 : Mediterranean Avenue .
##
## ## Turn 18
## Rolled: 3 3
## Double count is: 1 .
## Player T at: 2 , Mediterranean Avenue
## Player T moves: 6
## Player T now at: 8 , Chance .
## Tally at 8 : Chance .
## Draw chance card.
## Card Drawn: Go to Jail
## Going to Jail.
## Player T at: 8 , Chance .
## Player T now at: 11 , Jail .
## Tally at 11 : Jail .
##
## ## Turn 19
```

```
## Number of turns in jail: 1
## Rolled: 1 1
## Rolled doubles. Get out of jail.
## Player T at: 11 , Jail
## Player T moves: 2
## Player T now at: 13 , Electric Company .
## Tally at 13 : Electric Company .
##
## ## Turn 20
## Rolled: 2 1
## Player T at: 13 , Electric Company
## Player T moves: 3
## Player T now at: 16 , Pennsylvania Railroad .
## Tally at 16 : Pennsylvania Railroad .
```

```
space_tracking$tally
```

```
##  [1] 0 1 0 1 0 0 0 1 1 0 7 0 1 0 0 1 1 0 2 1 0 0 0 1 1 0 1 0 0 1 0 0 2 0 1
## [36] 0 0 0 0 0
```

```
##
```

Running 1000 simulation

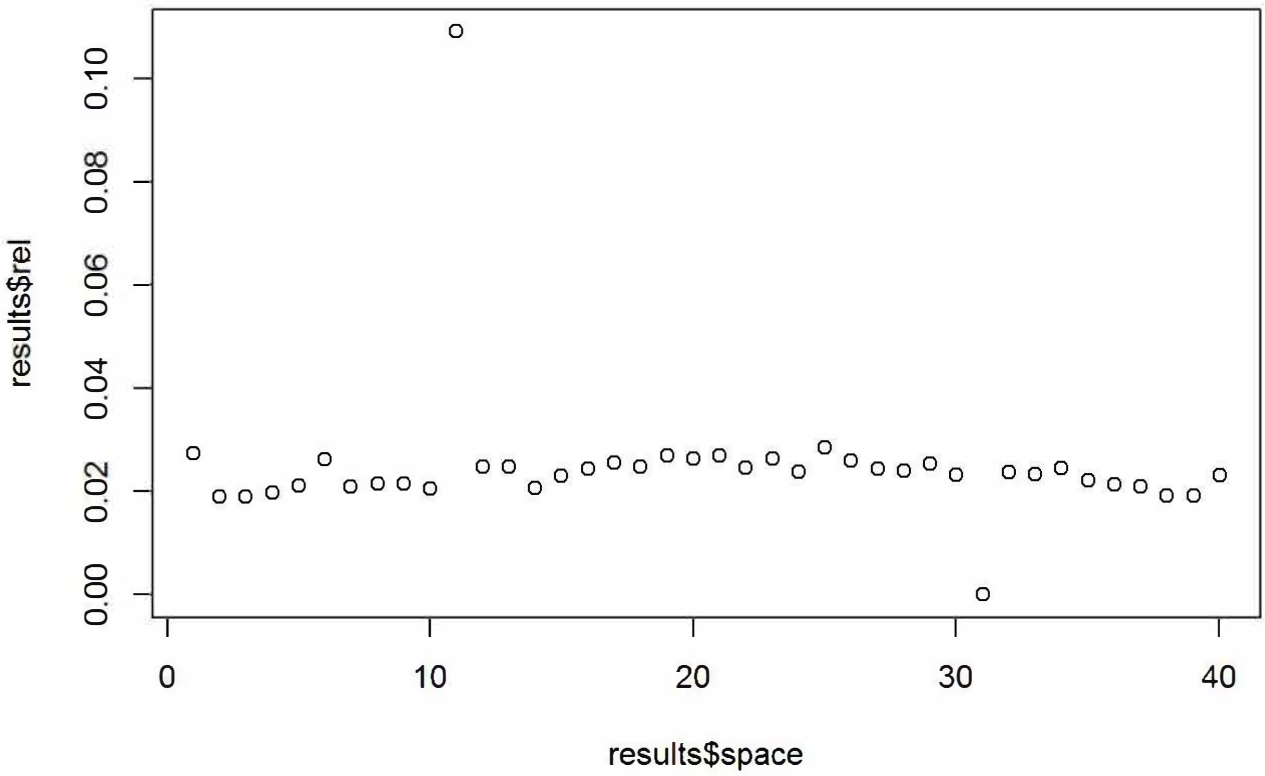
```
## Running 1000 simulation ---------------------------------
reset_game(reset_player = TRUE, reset_tally = TRUE)
manual_switch <- FALSE
turn_verbose_on(FALSE)
verbose_switch <- FALSE

set.seed(1)

for(i in 1:1000){ # simulate 1000 games
  # cat("#### NEW GAME",i,"##### \n")
  reset_game(reset_player = TRUE, reset_tally = FALSE)
  for(i in 1:150){ # 150 turns for each game
    if(player1$verbose) cat("Player 1 turn\n")
    taketurn(player1, space_tracking)
    if(player2$verbose) cat("Player 2 turn\n")
    taketurn(player2, space_tracking)
  }
}

results <- cbind(gameboard, tally = space_tracking$tally)
results <- cbind(results, rel = results$tally/sum(results$tally))
results <- cbind(rank=1:40,results[order(results$rel,decreasing=TRUE),])
results
```

```
##    rank space                title tally        rel
## 11    1    11                 Jail 39983 0.10900937
## 25    2    25       Illinois Avenue 10457 0.02850989
## 1     3     1                   Go 10035 0.02735935
## 21    4    21          Free Parking  9928 0.02706763
## 19    5    19      Tennessee Avenue  9887 0.02695585
## 23    6    23               Chance  9709 0.02647055
## 20    7    20       New York Avenue  9694 0.02642965
## 6     8     6      Reading Railroad  9593 0.02615429
## 26    9    26        B & O Railroad  9537 0.02600161
## 17   10    17       St. James Place  9409 0.02565263
## 29   11    29           Water Works  9324 0.02542089
## 18   12    18       Community Chest  9147 0.02493832
## 12   13    12     St. Charles Place  9134 0.02490287
## 13   14    13      Electric Company  9125 0.02487833
## 22   15    22       Kentucky Avenue  9082 0.02476110
## 34   16    34       Community Chest  8964 0.02443939
## 16   17    16 Pennsylvania Railroad  8951 0.02440394
## 27   18    27       Atlantic Avenue  8950 0.02440122
## 28   19    28        Ventnor Avenue  8871 0.02418583
## 24   20    24        Indiana Avenue  8791 0.02396772
## 32   21    32        Pacific Avenue  8661 0.02361329
## 33   22    33 North Carolina Avenue  8580 0.02339245
## 30   23    30        Marvin Gardens  8543 0.02329157
## 15   24    15       Virginia Avenue  8465 0.02307892
## 40   25    40             Boardwalk  8458 0.02305983
## 35   26    35   Pennsylvania Avenue  8135 0.02217921
## 9    27     9        Vermont Avenue  7906 0.02155486
## 36   28    36   Short Line Railroad  7848 0.02139673
## 8    29     8               Chance  7811 0.02129585
## 5    30     5           Income Tax  7737 0.02109410
## 7    31     7       Oriental Avenue  7717 0.02103957
## 37   32    37               Chance  7714 0.02103139
## 14   33    14         States Avenue  7647 0.02084873
## 10   34    10    Connecticut Avenue  7548 0.02057881
## 4    35     4         Baltic Avenue  7245 0.01975272
## 38   36    38            Park Place  7088 0.01932467
## 39   37    39            Luxury Tax  7079 0.01930013
## 3    38     3       Community Chest  7061 0.01925106
## 2    39     2  Mediterranean Avenue  6971 0.01900568
## 31   40    31            Go to jail     0 0.00000000
```

```
sum(results$tally)
```

```
## [1] 366785
```

```
plot(results$space,results$rel)
```


results$rel
0.00
0.02
0.04
0.06
0.08
0.10
0
10
20
30
40
results$space