Stats 102A

Homework 2 - Shall We Play A Game?

June 25, 2019

General Notes

- You will submit a minimum of three files, the core files must conform to the following naming conventions (including capitalization and underscores). 123456789 is a placeholder, please replace these nine digits with your nine-digit UID. The files you must submit are:
    1. `123456789_stats102a_hw2.R` An R script file containing all of the functions you wrote for the homework. The first line of your .Rmd file, after loading libraries, should be sourcing this script.
    2. `123456789_stats102a_hw2.Rmd` Your markdown file which generates the output file of your submission.
    3. `123456789_stats102a_hw2.html/pdf` Your output file, *either* a PDF or an HTML file depending on the output you choose to generate.
    4. `included image files` You may name these what you choose, but you must include all of the image files you generated for your structured flowcharts, otherwise your file will not knit.

    If you fail to submit any of the required core files you will receive **ZERO** points for the assignment.

    If you submit any files which do not conform to the specified naming convention, you will receive (at most) **half credit** for the assignment.
- Your coding should adhere to the tidyverse style guide: https://style.tidyverse.org/
- All flowcharts should be done on separate sheets of paper, but be included, inline as images, in your final markdown document.
- Any functions you write should be included in a separate functions file.
- Your .Rmd file must knit. If your .Rmd file does not knit you will receive (at most) **half credit** for the assignment.

The two most common reasons files fail to knit are because of workspace/directory structure issues and because of missing include files. To remedy the first, ensure all of the file paths in your document are relative paths pointing at the current working directory. To remedy the second, simply make sure you upload any and all files you source or include in your .Rmd file.

NOTE: *Everything* you need to do this assignment is here, in your class notes, or was covered in discussion or lecture.

- **DO NOT** look for solutions online.
- **EVERYTHING** you submit **MUST** be 100% your, original, work product. Any student suspected of plagiarizing, in whole or in part, any portion of this assignment, will be **immediately** referred to the **Dean of Student's office** without warning, and you will get a zero for all parts of the tainted homework.
- **YOU MAY** collaborate on this assignment with a maximum of **TWO** other students.
    - Each person must type up and submit their own files.
    - Each person **MUST** include the name and UID of each collaborator at the top of their output file.
    - You and your collaborators will receive the same grade though only one set of files will be scored. Make sure you trust your collaborators!
    - If you collaborate with another student and do not disclose it, it will be considered plagiarism.

Homework 2 Requirements

For each of the following problems you will:

a) Create a structured flowchart of the game and all functions/subroutines.

b) Write pseudocode sufficiently complete, clear, and concise enough to enable a person to accurately implement the your design in any programming language they are adept with using.

c) You do not need to do complexity analysis of your game.

d) Finally, you must write the functions/subroutines which will allow your game to be played.

Chutes and Ladders

Rules

Chutes and Ladders®

INSTRUCTIONS

For 2 to 4 Players/AGES 3+

This delightful game is simple and easy to play, even for children who can't read. Fun pictures help kids understand the rewards of doing good deeds as they climb up the ladders and the consequences of naughty ones as they slide down the chutes.

CONTENTS

Adult Assembly Required

• Gameboard • Spinner with plastic arrow
• 4 Pawns with plastic stands

OBJECT

Be the first player to reach "Winner" square #100.

THE FIRST TIME YOU PLAY

1. Punch out the 4 pawns from the cardboard sheet. Put each pawn into a plastic stand.

2. Punch out the spinner board from the paper sheet. Discard the waste. Carefully remove the spinner arrow and base from the plastic frame. If needed, use an emery board or sandpaper to remove the excess plastic from the game pieces. Discard the frame after removing all of the game pieces. Then assemble the spinner as shown in Figure 1.

FIGURE 1

SETUP

Position the gameboard so all the players can easily move their pawns from square to square. Everyone chooses a pawn to play. Any extra pawns are out of play. Chosen pawns start off the board near square #1. Now get ready for the fun!

ALL ABOUT THE SQUARES:

Take a peek at the gameboard. The squares are numbered from 1 to 100. Players' pawns will move back and forth across the board, following the numbers upward – starting at square #1 and moving right toward square #10, then up to square #11 and left toward square #20, etc.

Of course, you can also move up by climbing ladders and sometimes go down, too, by sliding down chutes. More about that later.

HOW TO PLAY

Everyone spins the spinner. The player with the highest number goes first. Play proceeds to the left.

WHAT TO DO ON YOUR TURN:

On your turn, spin the spinner and move your pawn, square by square, the number shown on the spinner. For example, on your first turn, if you spin a 5, move to square #5 on the board. Once you move your pawn, your turn is over. NOTE: Two or more pawns may be on the same space at the same time.

GOING UP A LADDER OR DOWN A CHUTE

LADDERS: Any time a pawn ends its move on a picture square at the bottom of a ladder, that pawn must climb up to the picture square at the top of the ladder. For example, if you end your move on square #9, you can immediately move up to square #31.

Notice that the pictures on these two squares are related. The little boy who mows the lawn is rewarded for his job with a trip to the circus.

Climb Up With A Good Deed!

Naughty Deeds Slide You Back!

CHUTES: Any time a pawn ends its move on a picture square at the top of a chute, that pawn must slide down the chute to the picture square at the bottom of the chute.

For example, if you end your move on square #49, you must immediately move down to square #11. Again, the pictures are related. Eating too many cookies can give you a tummy-ache!

If your pawn ends its turn on any of the following spaces, your turn is over:

- a square with no picture
- a square with no picture and just an arrow
- a square that a ladder or chute just passes through
- a picture square at the top of a ladder
- a picture square at the bottom of a chute

WINNING THE GAME

The first player to reach the "Winner" square #100 wins the game. You can get there 2 ways:

1. Land there by exact count. If your spin would take you past square #100, don't move. Try again on your next turn.

2. Climb there by ending your move on ladder square #80.

100 WINNER

Pet Show

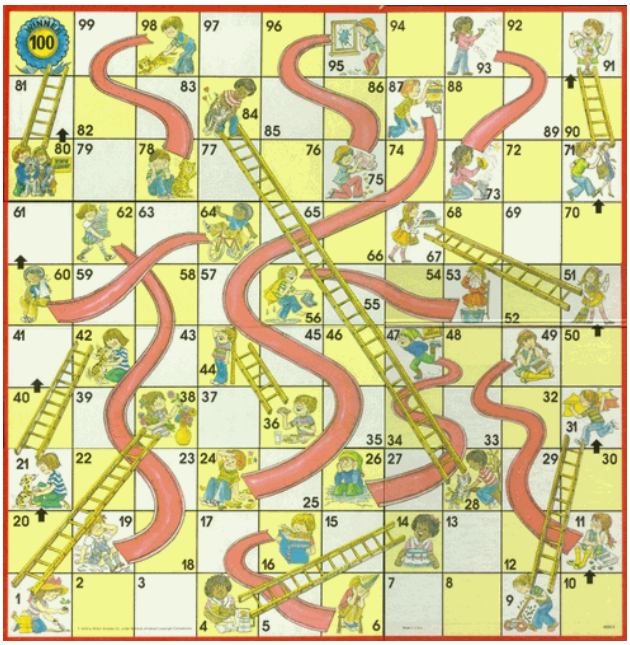
1144555MBU

Not suitable for children under 3 years because of small parts – choking hazard.

GAME PARTS STORED BELOW

NOTE: The spinner has values 1-6.

(1) Create a representation of the game board.



Create an object in R to represent the default Chutes and Ladders game board, in whatever way you choose.

(2) Create a function to show the game board.

Your function should be named `show_board()` and take one argument `board` which is the internal representation of the game board you have decided on.

The output of the function should be an image (i.e., a graphical representation of the game board) showing the board with arrows representing the chutes and ladders. In keeping with the spirit of the original game, make red arrows to represent the chutes and yellow arrows to represent the ladders.

(3) Write a function to simulate one game of Chutes and Ladders.

Your function should be named `play_cl()` and take the following arguments:

`n_players` The number of players. The original rules stipulate this is a game for 2-4 players, but your function should accept an arbitrary integer value greater than zero. The default value should be 1.

`spinner` The number of values on the spinner or the individual values on the spinner. If `spinner` is a single integer value (example: 6), your game will be played with a spinner that has 6 values on it: 1, 2, 3, 4, 5, 6. If, however, the value in `spinner` is a vector of integer values (example: `c(1, 1, 3, 5, 7)`), your game will be played with a spinner with exactly those values.

The output of your function should be a list containing *at least* the following information:

`winner` An integer representing which player won the game.

`turns` A length-n_players integer vector containing the number of turns taken by each player.

`chutes` A length-n_players integer vector containing the number of times each player slid down a chute.

`ladders` A length-n_players integer vector containing the number of times each player climbed a ladder.

For some of the questions you will be answering, you may need to include additional information in the list you return. Read through *all* of the question you will be asked to answer *before* you start programming.

Questions

A) Standard Game

1. Using the `play_cl()` function, simulate $10,000$ one-player games of Chutes and Ladders, immediately before you simulate your data, set your random seed to your UID.
2. Using the `play_cl()` function, simulate $10,000$ two-player games of Chutes and Ladders, immediately before you simulate your data, set your random seed to your UID with the digits in reverse order.
3. Using the `play_cl()` function, simulate $10,000$ three-player games of Chutes and Ladders, immediately before you simulate your data, set your random seed to your UID then set the random seed to `sample(1e4, 1)`.

Answer the following questions for each of your simulations above:

a) What is the minimum number of turns needed to win the game? **NOTE:** This is the number of turns *all* players took before the game ended.

b) What proportion of games are won by each player?

c) What proportion of games ended in the minimum number possible?

d) What proportion of games were "close?" For the purpose of this question, we will define a close game as one where a player other than the winner would be capable of winning on their next turn.

e) What is the maximum number of turns needed to finish the game.

f) If a player has completed at least least 6 turns, what is the most likely square for that player to be on at the end of their 6^{th} turn?

Also answer the following questions:

4. Plot the distribution of turns needed to finish each game setup (one, two, and three-player) on one set of axes.
5. What proportion of three-player games ended in fewer total turns than the observed maximum number of turns for one-player games

B) Custom Game

Write a function `make_random_board()` which, perhaps unsurprisingly, generates a random Chutes and Ladders game board to play on. This function should take *at least* the following arguments:

`n_rows` The number of rows. The default value should be 10.

`n_cols` The number of columns. The default value should be 10.

`n_chutes` The number of chutes. The default value should be 10.

`n_ladders` The number of columns. The default value should be 9.

You are free to decide how the chutes and ladders are placed on your board, with the following caveats:

- It must be possible to complete the game.
- No chute or ladder may end or begin on the same square another chute or ladder begins or ends on.

You are free to add any additional *optional* arguments you want, though you are not required to do so. Some ideas might include:

`longest_chute` Maybe you don't want super long chutes which send a player from square 99 back to square 1.

`shortest_ladder` Maybe you don't want pointless ladders which let a player move up only one square.

`min_spacing` Maybe you want to make sure you don't have the starts of two chutes or the starts of two ladders right next to each other. Otherwise, you *could* generate an impossible game (imagine a game with six chutes in a row and no ladder bypassing them).

Once you have the function `make_random_board()` written, generate a 11×13 board with 13 chutes and 11 ladders. Show the board, and run the three simulations again with a spinner with the values 1, 2, 3, 5, 7, 11, and 13 (with the same seed setting procedures) then answer all of the above questions again.