Simulation Portfolio Assignment

Quest for the Charms!

Aaradhana Bharill,CM Box 3202

## Introduction

In this report, a simulation of the game “Quest for the Charms” (found behind the “Lucky Charms” cereal box“) is made. The simulation is used to answer the following questions for a 4-player game: 1. How likely is it that player 3 wins the game? 2. On average, how many rounds will a game last? 3. How likely is it that at least one player lands on a forward slide?

## Methods

First a matrix of the game board was made, as shown below. The matrix has 7 rows and 8 columns.

Then a function was written, defining what to do when it is a player’s turn. The function takes the input valuesof the charm the player picked, and the existing row and column position of the player in the matrix. The function is broken into smaller chunks by if statements for each row. There are 6 different functions for the 7 rows (the last row doesn’t need a function because the player can never be in that row because of the backward slide.) The turn function returns the new row and column for the player, along with a boolean value for whether the player slided forward in that turn or not.

The following assumptions were made when deciding how the player moves: 1. It was assumed that to win, a player would have to pick the same charm again, that they were already positioned on so for e.g. if the player was on a rainbow, then they would have to be positioned on the last rainbow in the game and pick up a rainbow again to win. 2. On the heart at the end of the game, the player moves backward and does not miss a turn. (The game board was ambiguous in instructions).

# Load Necessary Packages  
library(dplyr)  
library(base)  
  
#setup game  
game = matrix(   
 c("c1","c2","c3","c4","c5","c6","c7","c8", #if c4 move to c8 in this row  
 "c1","c2","c3","c4","c5","c6","c7","c8", #if c6, move to c3 in this row  
 "c1","c2","c3","c4","c5","c6","c1","c8",  
 "c7","c2","c3","c4","c5","c6","c7","c8", #if the first c7, then move to the second c7 in this row--rainbow bridge  
 "c1","c2","c3","c4","c5","c6","c7","c8",  
 "c1","c2","c5","c4","c5","c6","c7","c8", #if c2, move to c5 in this row  
 "c1", "win" ,"", "", "", "", "", ""), #if on c1 here, go back 3 i.e. to c6 (arrow on game drawing in pdf)  
nrow=7, ncol=8, byrow = TRUE)  
  
slide=FALSE  
#needed functions  
prow1 <- function(prow,pcol,charm) {  
 if(charm==1){  
 prow=2  
 pcol=1  
 }  
 else if(charm==4&pcol<4){  
 pcol=8  
 slide=TRUE  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 else{  
 prow=2  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
prow2 <- function(prow,pcol,charm) {  
 if(charm==6&pcol<6){  
 prow=3  
 }  
 else if(charm==pcol){  
 if(charm==7){  
 prow=4  
 pcol=7#takes rainbow bridge from col1 to col7  
 slide=TRUE  
 }  
 else{  
 prow=3  
 }  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 else{  
 prow=3  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
prow3 <- function(prow,pcol,charm) {  
 if(charm==1){  
 if(pcol==1){  
 pcol=7  
 }  
 else{#pcol==7  
 prow=5  
 pcol=1  
 }  
 }  
 else if(charm==pcol){  
 prow=4  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 else{  
 prow=4  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
prow4 <- function(prow,pcol,charm) {  
 if(charm==pcol){  
 prow=5  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 else{  
 prow=5  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
prow5 <- function(prow,pcol,charm) {  
 if(charm==2){  
 prow=6  
 pcol=5  
 slide=TRUE  
 }  
 else if(charm==3){#win!  
 prow=7  
 pcol=2  
 }  
 else if(charm==5){  
 prow=6  
 pcol=3  
 }  
 else if(charm==pcol){  
 prow=6  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 else{  
 prow=6  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
prow6 <- function(prow,pcol,charm) {  
 if(charm==1){  
 prow=6  
 pcol=6  
 }  
 else if(charm==5){  
 if(pcol==3){  
 pcol=5  
 }  
 else{#pcol=5#win!!   
 prow=7  
 pcol=2  
 }  
 }  
 else if(charm==pcol){#win!  
 prow=7  
 pcol=2  
 }  
 else if(charm>pcol){  
 pcol=charm  
 }  
 return(c(prow,pcol,slide))  
}  
  
#a player's turn  
turn<-function(charm,prow,pcol){  
 if(prow==1){  
 vec<-prow1(prow,pcol,charm)  
 }  
 else if(prow==2){  
 vec<-prow2(prow,pcol,charm)  
 }  
 else if(prow==3){  
 vec<-prow3(prow,pcol,charm)  
 }  
 else if(prow==4){  
 vec<-prow4(prow,pcol,charm)  
 }  
 else if(prow==5){  
 vec<-prow5(prow,pcol,charm)  
 }  
 else{#(prow==6)  
 vec<-prow6(prow,pcol,charm)  
 }  
 #prow<-vec[1]  
 #pcol<-vec[2]  
 #slide<-vec[3]  
 return(c(vec[1],vec[2],vec[3]))  
}

Then the play function is defined which plays a game, and returns a vector with the number of the player who won, the number of rounds that the game lasted, and a boolean value for whether any player slided in a game or not. The function takes an input variable for the number of players playing the game. In the case of this report, it is 4 players.

First, the function generates all the players for the game and puts them at the start row and column positions of the game. Then, it executes a for loop for all the players. The for loop is inside an infinite while loop that returns when someone wins. Inside the for loop, first a random charm is picked for the player. Then, using the turn function, the new player positions, are decided.

#play game  
play<-function(numPlayers){  
 slide=FALSE  
 for(i in 1:numPlayers){  
 players <- cbind(row=rep(1,numPlayers),col=rep(1,numPlayers))  
 rownames(players)<-paste0("player",1:numPlayers)  
 }  
 while(TRUE){  
 for(i in 1:numPlayers){  
 charm<-sample(hand, size=1, prob=c(1/8,1/8,1/8,1/8,1/8,1/8,1/8,1/8))  
 prow<-players[i,1]  
 pcol<-players[i,2]  
 newstate<-turn(charm,prow,pcol)  
 newprow<-newstate[1]  
 newpcol<-newstate[2]  
 if(!slide&newstate[3]){  
 slide=TRUE  
 }  
 if(game[newprow,newpcol]=="win"){  
 winner<-i  
 return(c(winner,round,slide))   
 }  
 players[i,1]<-newprow  
 players[i,2]<-newpcol  
 }  
 round=round+1#round is counted when round ends  
 }  
}

Finally the simulation is run where the play function is executed 10000 times and the results are saved in 3 differnt vectors: winners, rounds, and slides (boolean forwards slides).

#set start parameters  
set.seed(20181019)  
hand<-c(1,2,3,4,5,6,7,8)#8charmss  
prow<-1  
pcol<-1  
numPlayers<-4  
winner<--1  
round=0  
  
winners<-vector()  
rounds<-vector()  
slides<-vector()  
  
m<-10000  
for(i in 1:m){  
 played<-play(numPlayers)  
 winners<-c(winners, played[1])  
 rounds<-c(rounds,played[2])  
 slides<-c(slides,played[3])  
}

## Results

The code below shows the results obtained and how they were obtained. Looking at the entire winners, table, it makes sense that the first player would have hte best chances of winning, and the chances of winning would consequently fall for each player after. It seems that on average, the game lasts 8 rounds. That makes sense too, considering that there are 6 filled rows with 8 charms each, and there’s generally a good chance for each player to move about 1 row in each turn and struggle to make it to the win position. Finally there is 71.7% chance of a player sliding forward in the game.

prop.table(table(winners))

winners  
 1 2 3 4   
0.2898 0.2622 0.2402 0.2078

#How likely is it that player 3 wins the game?  
prop.table(table(winners))[[3]]

[1] 0.2402

#On average, how many rounds will a game (with all four of us playing) last?  
mean(rounds)

[1] 7.5596

#How likely is it that at least one of us lands on such a   
#forwards slide (not backward) during a game?  
prop.table(table(slides))[[2]]

[1] 0.717