Final\_q3

Amber Duevel

4/26/2022

#checking to see if the dice roll is a double  
check\_double<-function(face, double\_count){  
 #does each separate roll match the other  
 if(face[1]==face[2]){  
 #increasing the count by 1 if it is a double  
 double\_count=double\_count+1  
 #if you roll three doubles in a row, the player is sent to jail  
 if(double\_count==3){  
 #11 is position of jail  
 player\_df[player+1, 2] = 11  
 assign("player\_df",player\_df,.GlobalEnv)  
 return(sum(face))  
 }  
 else {  
 #space is the new current space, and updating player\_df with the new position, mono\_board count increases by 1, and mono\_graph\_data adds a row with the new data  
 space = player\_df[player+1, 2] + sum(face)  
 if (space == 40) {  
 space = 40  
 }  
 else {  
 space = space%%40  
 }  
   
 mono\_board[space, 2] = as.numeric(mono\_board[space, 2] + 1)  
 player\_df[player+1, 2] = space  
   
 mono\_graph\_data = rbind(mono\_graph\_data, c(player\_df[player+1, 1], mono\_board[player\_df[player+1, 2], 3], mono\_board[player\_df[player+1, 2], 4]))  
   
 #rolls the dice and gets new face value and calls check\_doubles, which is a recursive call  
 face = sample(1:6,2,replace=TRUE)  
 check\_double(face, double\_count)  
 }  
 }else{  
 #return the sum of the faces and go back to dice function  
 assign("mono\_board", mono\_board, .GlobalEnv)  
 assign("player\_df", player\_df, .GlobalEnv)  
 assign("mono\_graph\_data", mono\_graph\_data, .GlobalEnv)  
 return(sum(face))  
 }  
}  
#calling jail out when a player is in jail and is trying to get out of jail by rolling doubles  
jail\_out<-function(face, jail\_count){  
 if(face[1]==face[2]){  
 movement=sum(face)  
 }else{  
 jail\_count=jail\_count+1  
 #by the third round, if you don't roll doubles, player still leaves jail  
 if(jail\_count==3){  
 movement=sum(face)  
 }  
 return(jail\_count)  
 }  
}  
#calling dice with each turn for each player  
dice <- function(){  
 double\_count = 0  
 jail\_count = 0  
 #rolling the dice  
 face = sample(1:6,2,replace=TRUE)  
 #if the person is in jail we called the jail\_out function  
 if(player\_df[player+1, 2] == 11){  
 while(jail\_count<3){  
 jail\_out(face, jail\_count)  
 break  
 }  
 }  
 #returning the sum of the roll so we can add it to the position  
 sum\_face = check\_double(face, double\_count)  
 print(sum\_face)  
 return(sum\_face)  
}

#establishing the monopoly board with positions  
mono\_board = data.frame()  
# All of the spaces on the board  
go = c("Go", 0, 200, 0, "None")  
mediterranean = c("Mediterranean Avenue", 0, 172, 0, "Purple")  
community1 = c("Community Chest1", 0, 155, 0, "Cards")  
baltic = c("Baltic Avenue", 0, 136, 0, "Purple")  
income = c("Income Tax", 0, 118, 0, "None")  
reading = c("Reading Railroad", 0, 100, 0, "Railroad")  
oriental = c("Oriental Avenue", 0, 82, 0, "Light Blue")  
chance1 = c("Chance1", 0, 64, 0, "Cards")  
vermont = c("Vermont Avenue", 0, 46, 0, "Light Blue")  
connecticut = c("Connecticut Avenue", 0, 28, 0, "Light Blue")  
just\_visiting\_jail = c("Just Visiting/Jail", 0, 0, 0, "None")  
charles = c("St. Charles Place", 0, 0, 28, "Pink")  
electric = c("Electric Company", 0, 0, 46, "Utilities")  
states = c("States Avenue", 0, 0, 64, "Pink")  
virginia = c("Virginia Avenue", 0, 0, 82, "Pink")  
penn\_railroad = c("Pennsylvania Railroad", 0, 0, 100, "Railroad")  
james = c("St. James Place", 0, 0, 118, "Orange")  
community2 = c("Community Chest2", 0, 0, 136, "Cards")  
tennessee = c("Tennessee Avenue", 0, 0, 154, "Orange")  
new\_york = c("New York Avenue", 0, 0, 172, "Orange")  
parking = c("Free Parking", 0, 0, 200, "None")  
kentucky = c("Kentucky Avenue", 0, 28, 200, "Red")  
chance2 = c("Chance2", 0, 46, 200, "Cards")  
indiana = c("Indiana Avenue", 0, 64, 200, "Red")  
illinois = c("Illinois Avenue", 0, 82, 200, "Red")  
b\_o = c("B&O Railroad", 0, 100, 200, "Railroad")  
atlantic = c("Atlantic Avenue", 0, 118, 200, "Yellow")  
ventnor = c("Ventnor Avenue", 0, 136, 200, "Yellow")  
water = c("Water Works", 0, 155, 200, "Utilities")  
marvin = c("Marvin Gardens", 0, 172, 200, "Yellow")  
jail = c("Go to Jail", 0, 200, 200, "None")  
pacific = c("Pacific Avenue", 0, 200, 172, "Green")  
carolina = c("North Carolina Avenue", 0, 200, 154, "Green")  
community3 = c("Community Chest3", 0, 200, 136, "Cards")  
penn\_avenue = c("Pennsylvania Avenue", 0, 200, 118, "Green")  
short = c("Short Line", 0, 200, 100, "Railroad")  
chance3 = c("Chance3", 0, 200, 82, "Cards")  
park = c("Park Place", 0, 200, 64, "Dark Blue")  
luxury = c("Luxury Tax", 0, 200, 46, "None")  
boardwalk = c("Boardwalk", 0, 200, 28, "Dark Blue")  
mono\_board = rbind(mono\_board, go, mediterranean, community1, baltic, income, reading, oriental, chance1, vermont, connecticut, just\_visiting\_jail, charles, electric, states, virginia, penn\_railroad, james, community2, tennessee, new\_york, parking, kentucky, chance2, indiana, illinois, b\_o, atlantic, ventnor, water, marvin, jail, pacific, carolina, community3, penn\_avenue, short, chance3, park, luxury, boardwalk)  
colnames(mono\_board) = c('space\_name', 'count', 'x\_dimension', 'y\_dimension', 'space\_color')

#making it so we can display the board as a graph  
mono\_board = transform(mono\_board, x\_dimension = as.numeric(x\_dimension))  
mono\_board = transform(mono\_board, y\_dimension = as.numeric(y\_dimension))  
mono\_board = transform(mono\_board, count = as.numeric(count))

#establishing the players and putting them into a data frame  
#each player holds what number player, position, and how much money they have  
player\_df = data.frame()  
player1 = c("player1", 1, 1500)  
player2 = c("player2", 1, 1500)  
player3 = c("player3", 1, 1500)  
player4 = c("player4", 1, 1500)  
player\_df = rbind(player\_df, player1, player2, player3, player4)  
colnames(player\_df) = c("name", "position", "money")  
player\_df = transform(player\_df, position = as.numeric(position))  
num\_players = nrow(player\_df)  
#mono\_graph\_data is a dataframe which stores the history of each players moves around the board  
mono\_graph\_data = data.frame()  
mono\_graph\_data = rbind(c(player\_df[1, 1], mono\_board[player\_df[1, 2], 3], mono\_board[player\_df[1, 2], 4]),   
 c(player\_df[2, 1], mono\_board[player\_df[2, 2], 3], mono\_board[player\_df[2, 2], 4]),   
 c(player\_df[3, 1], mono\_board[player\_df[3, 2], 3], mono\_board[player\_df[3, 2], 4]),   
 c(player\_df[4, 1], mono\_board[player\_df[4, 2], 3], mono\_board[player\_df[4, 2], 4]))  
colnames(mono\_graph\_data) = c('player', 'x\_coordinate', 'y\_coordinate')  
rounds = 0  
player = 0  
#moving the players to new positions based on the number they rolled  
move = function() {  
 roll = dice()  
 #adding their roll to their current position  
 space = player\_df[player+1, 2] + roll  
 if (space == 40) {  
 space = 40  
 }  
 else {  
 space = space%%40  
 }  
 return(space)  
}  
#uses the returned space value and assigns it to s  
player\_turn = function(player) {  
 s = move()  
 mono\_board[s, 2] = as.numeric(mono\_board[s, 2] + 1)  
 player\_df[player+1, 2] = s  
   
 mono\_graph\_data = rbind(mono\_graph\_data, c(player\_df[player+1, 1], mono\_board[player\_df[player+1, 2], 3], mono\_board[player\_df[player+1, 2], 4]))  
   
 return(c(assign("mono\_board", mono\_board, .GlobalEnv), assign("player\_df", player\_df, .GlobalEnv), assign("mono\_graph\_data", mono\_graph\_data, .GlobalEnv)))  
}  
#number of rounds that the game is played  
while (rounds < 100) {  
 player\_turn(player)  
 cat(" Player: ", player\_df[player+1,1], " Position: ", player\_df[player+1,2])  
 player = (player+1)%%num\_players  
 rounds = rounds + 1  
}

## [1] 11  
## Player: player1 Position: 12[1] 9  
## Player: player2 Position: 10[1] 8  
## Player: player3 Position: 9[1] 5  
## Player: player4 Position: 6[1] 5  
## Player: player1 Position: 17[1] 11  
## Player: player2 Position: 21[1] 11  
## Player: player3 Position: 20[1] 7  
## Player: player4 Position: 13[1] 4  
## Player: player1 Position: 21[1] 4  
## Player: player2 Position: 25[1] 10  
## Player: player3 Position: 30[1] 6  
## Player: player4 Position: 19[1] 10  
## Player: player1 Position: 21[1] 7  
## Player: player2 Position: 32[1] 8  
## Player: player3 Position: 38[1] 5  
## Player: player4 Position: 24[1] 6  
## Player: player1 Position: 27[1] 3  
## Player: player2 Position: 35[1] 7  
## Player: player3 Position: 5[1] 4  
## Player: player4 Position: 28[1] 10  
## Player: player1 Position: 37[1] 8  
## Player: player2 Position: 3[1] 5  
## Player: player3 Position: 10[1] 8  
## Player: player4 Position: 36[1] 7  
## Player: player1 Position: 4[1] 5  
## Player: player2 Position: 8[1] 5  
## Player: player3 Position: 15[1] 10  
## Player: player4 Position: 6[1] 9  
## Player: player1 Position: 13[1] 7  
## Player: player2 Position: 15[1] 3  
## Player: player3 Position: 18[1] 5  
## Player: player4 Position: 11[1] 10  
## Player: player1 Position: 23[1] 6  
## Player: player2 Position: 21[1] 9  
## Player: player3 Position: 27[1] 4  
## Player: player4 Position: 15[1] 7  
## Player: player1 Position: 30[1] 7  
## Player: player2 Position: 28[1] 7  
## Player: player3 Position: 34[1] 10  
## Player: player4 Position: 25[1] 8  
## Player: player1 Position: 38[1] 7  
## Player: player2 Position: 35[1] 4  
## Player: player3 Position: 38[1] 5  
## Player: player4 Position: 30[1] 6  
## Player: player1 Position: 4[1] 8  
## Player: player2 Position: 3[1] 9  
## Player: player3 Position: 7[1] 5  
## Player: player4 Position: 35[1] 8  
## Player: player1 Position: 12[1] 4  
## Player: player2 Position: 7[1] 5  
## Player: player3 Position: 12[1] 4  
## Player: player4 Position: 39[1] 8  
## Player: player1 Position: 20[1] 9  
## Player: player2 Position: 16[1] 10  
## Player: player3 Position: 22[1] 12  
## Player: player4 Position: 23[1] 3  
## Player: player1 Position: 23[1] 6  
## Player: player2 Position: 22[1] 5  
## Player: player3 Position: 27[1] 6  
## Player: player4 Position: 29[1] 7  
## Player: player1 Position: 30[1] 7  
## Player: player2 Position: 29[1] 6  
## Player: player3 Position: 33[1] 7  
## Player: player4 Position: 36[1] 10  
## Player: player1 Position: 40[1] 8  
## Player: player2 Position: 37[1] 4  
## Player: player3 Position: 37[1] 7  
## Player: player4 Position: 3[1] 5  
## Player: player1 Position: 5[1] 5  
## Player: player2 Position: 2[1] 6  
## Player: player3 Position: 3[1] 9  
## Player: player4 Position: 12[1] 6  
## Player: player1 Position: 11[1] 7  
## Player: player2 Position: 9[1] 10  
## Player: player3 Position: 13[1] 8  
## Player: player4 Position: 20[1] 7  
## Player: player1 Position: 18[1] 11  
## Player: player2 Position: 20[1] 3  
## Player: player3 Position: 16[1] 9  
## Player: player4 Position: 29[1] 9  
## Player: player1 Position: 27[1] 9  
## Player: player2 Position: 29[1] 9  
## Player: player3 Position: 25[1] 6  
## Player: player4 Position: 35[1] 9  
## Player: player1 Position: 36[1] 7  
## Player: player2 Position: 36[1] 8  
## Player: player3 Position: 33[1] 5  
## Player: player4 Position: 40[1] 7  
## Player: player1 Position: 3[1] 9  
## Player: player2 Position: 5[1] 5  
## Player: player3 Position: 38[1] 9  
## Player: player4 Position: 9[1] 5  
## Player: player1 Position: 8[1] 6  
## Player: player2 Position: 11[1] 3  
## Player: player3 Position: 1[1] 4  
## Player: player4 Position: 13[1] 6  
## Player: player1 Position: 14[1] 11  
## Player: player2 Position: 22[1] 5  
## Player: player3 Position: 6[1] 6  
## Player: player4 Position: 19

#putting the image of the monopoly board on the graph  
library(grid)  
library(jpeg)  
library(tidyverse)

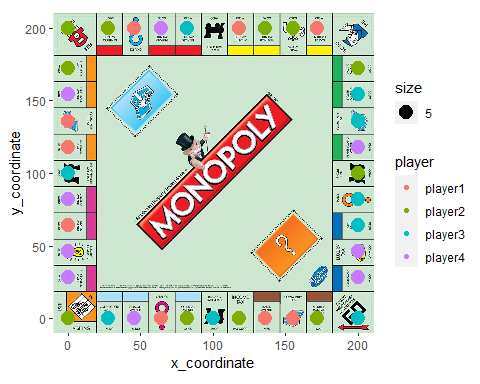
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.6 v dplyr 1.0.8  
## v tidyr 1.2.0 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1

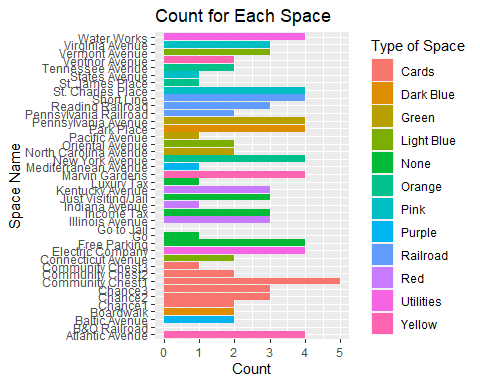
## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(ggplot2)  
library(gganimate)  
Sys.setenv("VROOM\_CONNECTION\_SIZE" = 131072 \* 2)  
board\_img = rasterGrob(readJPEG("monopolyboard.jpg"),   
 width = unit(1, "npc"), height = unit(1, "npc"))

#plotting all the positions that the players are at on the graph  
mono\_graph\_data = transform(mono\_graph\_data, x\_coordinate = as.numeric(x\_coordinate))  
mono\_graph\_data = transform(mono\_graph\_data, y\_coordinate = as.numeric(y\_coordinate))  
mono\_graph = ggplot(mono\_graph\_data, group = "player") +  
 annotation\_custom(board\_img) +  
 theme(aspect.ratio = 1) +  
 scale\_x\_continuous(limits = c(0, 200)) +  
 scale\_y\_continuous(limits = c(0, 200)) +  
 geom\_point(aes(x = x\_coordinate, y = y\_coordinate, color = player, size = 5))  
mono\_graph



#creating a bar graph of the amount of times each spot on the board is landed on  
count\_plot = ggplot(mono\_board, aes(x = space\_name, y = count), group = "space\_color") +  
 geom\_bar(stat = "identity", aes(fill = space\_color)) +  
 coord\_flip() +  
 ggtitle("Count for Each Space") +  
 ylab("Count") +  
 xlab("Space Name") +  
 labs(fill = "Type of Space")  
count\_plot

 The outcome of this question is very similar to the first question due to the face that we aren’t adding cards or the jail function. In this question we are taking into account the double rule. If a player roles a double they can roll again but if a player roles three doubles in a row they are sent to jail. This could impact the space jail. Rolling doubles three times in a row is very unlikely but possible. So we suspect on jail spot to have a higher count than other spaces because it’s the only position a player can move to. The other spots are still equally likely to occur. We ran our program one hundred rounds because there wasn’t a natural stop to the game due to the fact we aren’t dealing with money. After one hundred rounds we had North Carolina Avenue being landed on the most frequently. If we ran the program more we would get a more evenly split count for each space.