Final\_q2

Amber Duevel

4/26/2022

#rolling the dice and adding it's sum to the players current position  
dice\_simple<- function(verbose= FALSE){  
 face<- sample(1:6, 2, replace=TRUE)  
 movement=sum(face)  
}

#mono\_board is a dataframe which holds 40 space objects, with Go being the first space  
mono\_board = data.frame()  
#the 40 space objects as a vector  
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")  
#using rbind to add the 40 space objects to mono\_board  
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)  
#the column names for mono\_board are space\_name which holds characters of the name of the space, count which is numeric and holds the number of times the space was landed on in one "game", x-dimension which is numeric and holds the x-coordinate on the graph, y-coordinate which is numeric and holds the y-coordinate on the graph, and space\_color which is a character and holds the color of the space or the group that the space belongs best to.   
colnames(mono\_board) = c('space\_name', 'count', 'x\_dimension', 'y\_dimension', 'space\_color')  
#using transform() to have the x-dimension, y-dimension, and count columns be numeric instead of character  
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))

#player\_df dataframe which is storing position and money data for each player  
player\_df = data.frame()  
#four players were created with position = 1 and starting money is 1500  
player1 = c("player1", 1, 1500)  
player2 = c("player2", 1, 1500)  
player3 = c("player3", 1, 1500)  
player4 = c("player4", 1, 1500)  
#using rbind function to add the four player objects to the player\_df dataframe  
player\_df = rbind(player\_df, player1, player2, player3, player4)  
#the columns in the player\_df dataframe are name, position, and money  
colnames(player\_df) = c("name", "position", "money")  
#used transform() to have the position be numeric instead of a character  
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]))  
#the mono\_graph\_data dataframe stores the player, x-coordinate on the board plot, and y-coordinate on the board plot, which are each a column  
colnames(mono\_graph\_data) = c('player', 'x\_coordinate', 'y\_coordinate')  
#initially set rounds and current player to 0  
rounds = 0  
player = 0  
#move function which stores roll from dice\_simple output and add roll to the current position in player\_df and stores in space, and space is returned  
move = function() {  
 roll = dice\_simple()  
 space = player\_df[player+1, 2] + roll  
 #this if else is making sure that space doesn't exceed 40 because the spaces on the board are from 1-40  
 if (space == 40) {  
 space = 40  
 }  
 else {  
 space = space%%40  
 }  
 return(space)  
}  
#player\_turn takes in the current player, calls the move function and stores the current space in s, then increases the count on that space by 1 and updates player\_df to be the new current space  
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 is adding a new row to the dataframe with the player name, their x-coordinate, and y-coordinate  
 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]))  
   
 #need to return the assign mono\_board, player\_df, and mono\_graph\_data so that the new dataframes can be stored and called from the global environment  
 return(c(assign("mono\_board", mono\_board, .GlobalEnv), assign("player\_df", player\_df, .GlobalEnv), assign("mono\_graph\_data", mono\_graph\_data, .GlobalEnv)))  
}  
#running 100 rounds of the game  
while (rounds < 100) {  
 player\_turn(player)  
 cat(" Player: ", player\_df[player+1,1], " Position: ", mono\_board[player\_df[player+1, 2], 1], "\n")  
 player = (player+1)%%num\_players  
 rounds = rounds + 1  
}

## Player: player1 Position: Just Visiting/Jail   
## Player: player2 Position: Vermont Avenue   
## Player: player3 Position: Electric Company   
## Player: player4 Position: Reading Railroad   
## Player: player1 Position: Community Chest2   
## Player: player2 Position: Community Chest2   
## Player: player3 Position: New York Avenue   
## Player: player4 Position: Virginia Avenue   
## Player: player1 Position: Illinois Avenue   
## Player: player2 Position: Illinois Avenue   
## Player: player3 Position: Illinois Avenue   
## Player: player4 Position: Illinois Avenue   
## Player: player1 Position: Water Works   
## Player: player2 Position: Marvin Gardens   
## Player: player3 Position: Pacific Avenue   
## Player: player4 Position: Community Chest3   
## Player: player1 Position: Pennsylvania Avenue   
## Player: player2 Position: North Carolina Avenue   
## Player: player3 Position: Park Place   
## Player: player4 Position: Go   
## Player: player1 Position: Community Chest1   
## Player: player2 Position: Mediterranean Avenue   
## Player: player3 Position: Mediterranean Avenue   
## Player: player4 Position: Electric Company   
## Player: player1 Position: Oriental Avenue   
## Player: player2 Position: Just Visiting/Jail   
## Player: player3 Position: Oriental Avenue   
## Player: player4 Position: Tennessee Avenue   
## Player: player1 Position: St. Charles Place   
## Player: player2 Position: Free Parking   
## Player: player3 Position: Just Visiting/Jail   
## Player: player4 Position: Water Works   
## Player: player1 Position: Community Chest2   
## Player: player2 Position: Atlantic Avenue   
## Player: player3 Position: St. James Place   
## Player: player4 Position: Boardwalk   
## Player: player1 Position: Chance2   
## Player: player2 Position: Short Line   
## Player: player3 Position: Indiana Avenue   
## Player: player4 Position: Vermont Avenue   
## Player: player1 Position: Atlantic Avenue   
## Player: player2 Position: Go   
## Player: player3 Position: North Carolina Avenue   
## Player: player4 Position: Community Chest2   
## Player: player1 Position: Short Line   
## Player: player2 Position: Reading Railroad   
## Player: player3 Position: Luxury Tax   
## Player: player4 Position: B&O Railroad   
## Player: player1 Position: Mediterranean Avenue   
## Player: player2 Position: Virginia Avenue   
## Player: player3 Position: Chance1   
## Player: player4 Position: North Carolina Avenue   
## Player: player1 Position: Income Tax   
## Player: player2 Position: Indiana Avenue   
## Player: player3 Position: States Avenue   
## Player: player4 Position: Income Tax   
## Player: player1 Position: Connecticut Avenue   
## Player: player2 Position: Pacific Avenue   
## Player: player3 Position: Illinois Avenue   
## Player: player4 Position: Oriental Avenue   
## Player: player1 Position: Free Parking   
## Player: player2 Position: Go   
## Player: player3 Position: Water Works   
## Player: player4 Position: Just Visiting/Jail   
## Player: player1 Position: North Carolina Avenue   
## Player: player2 Position: St. Charles Place   
## Player: player3 Position: North Carolina Avenue   
## Player: player4 Position: Free Parking   
## Player: player1 Position: Community Chest1   
## Player: player2 Position: Free Parking   
## Player: player3 Position: Short Line   
## Player: player4 Position: Illinois Avenue   
## Player: player1 Position: Vermont Avenue   
## Player: player2 Position: B&O Railroad   
## Player: player3 Position: Mediterranean Avenue   
## Player: player4 Position: Pacific Avenue   
## Player: player1 Position: Virginia Avenue   
## Player: player2 Position: Pennsylvania Avenue   
## Player: player3 Position: Vermont Avenue   
## Player: player4 Position: Go   
## Player: player1 Position: Illinois Avenue   
## Player: player2 Position: Boardwalk   
## Player: player3 Position: Virginia Avenue   
## Player: player4 Position: Chance1   
## Player: player1 Position: Pacific Avenue   
## Player: player2 Position: Chance1   
## Player: player3 Position: Community Chest2   
## Player: player4 Position: States Avenue   
## Player: player1 Position: Boardwalk   
## Player: player2 Position: Virginia Avenue   
## Player: player3 Position: B&O Railroad   
## Player: player4 Position: New York Avenue   
## Player: player1 Position: Chance1   
## Player: player2 Position: Free Parking   
## Player: player3 Position: Go to Jail   
## Player: player4 Position: B&O Railroad   
## Player: player1 Position: St. James Place   
## Player: player2 Position: Illinois Avenue   
## Player: player3 Position: Short Line   
## Player: player4 Position: North Carolina Avenue

library(grid)  
library(jpeg)  
library(tidyverse)

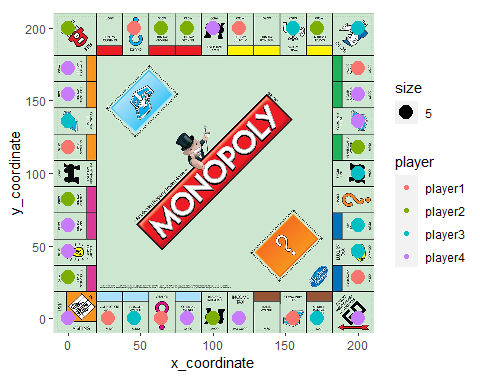
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## 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

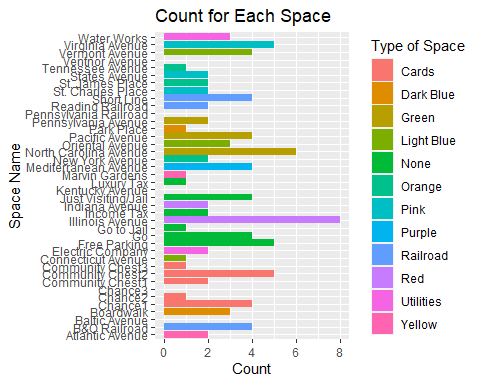
## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

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

#using the transform() function to change the x and y-coordinate columns to numeric in mono\_graph\_data  
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 is a scatterplot with the monopoly board as the background and the points are the plots where the players have been. There are four different colors, which represent each player  
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



#count\_plot is a bar graph with the space name as the y-axis and count for each space as the x-axis. Colors are grouped based on the type of space and color associated with each space on the board, if any  
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



Each spot has an equal probability of being landed on. Due to the fact that we are not checking for doubles, using the cards or having a jail function, all spots have an equal chance. This part of monopoly is simple because it’s just relying on the chances of a dice. It’s all dictated on the probability of a number on the dice being rolled. A dice has an even probability of each number being landed on so each space should be somewhat equal by the end of the game. We ran our program for one hundred rounds because there isn’t a natural stop to the game when you don’t deal with money or property. When we did this we got income tax to be landed on the most frequently. If we ran the program for more rounds we would find that each space was landed on an equal amount.