STA141A HW3 Report

Ruochen Zhong

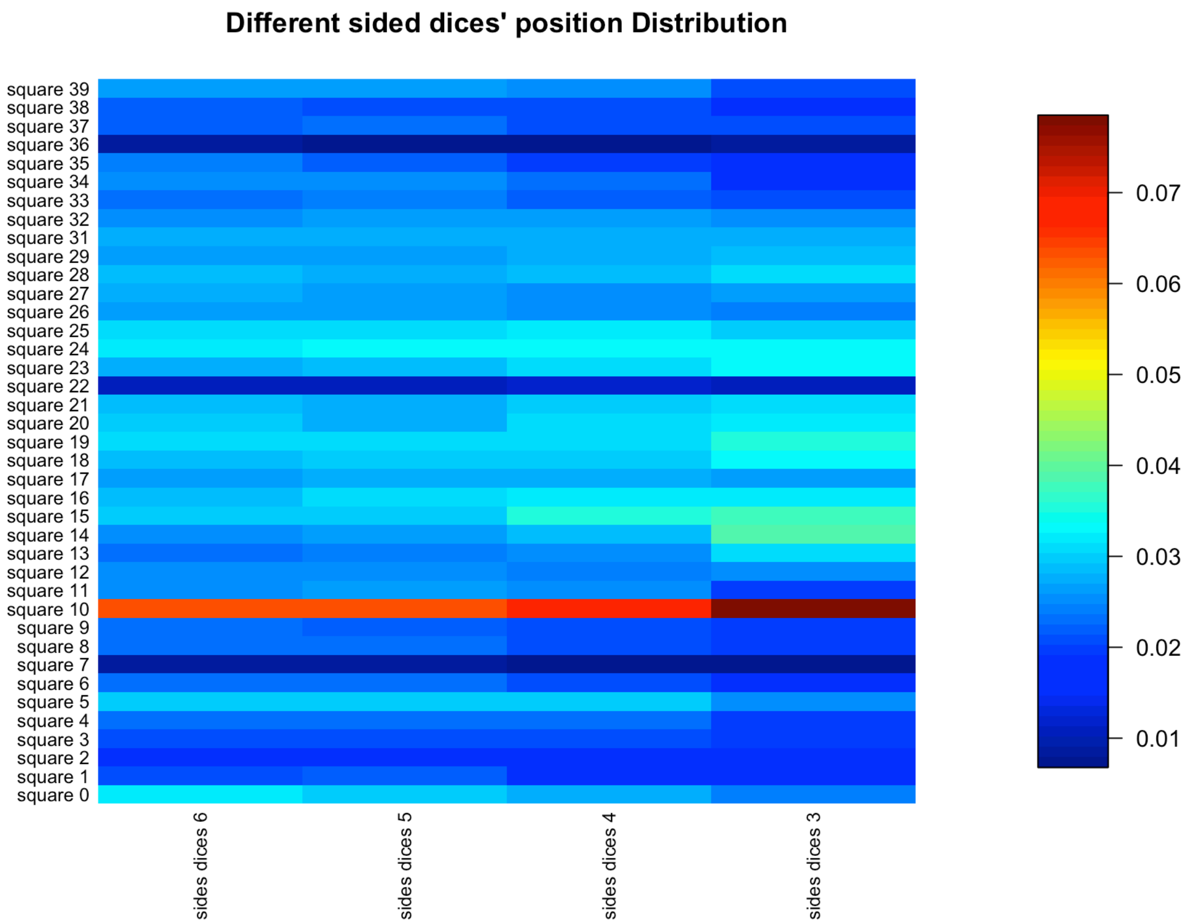
912888970

hw3

Part1 question 2

For **6-sided dice**, the 3 most likely squares to end a turn are ***10, 24, 0***. For **5-sided dice**, they are ***10, 24, 19***. For **4-sided dice**, they are ***10, 15, 24.*** For 3-sided dice, they are ***10,14,15.***

The distribution of long term probabilities for 3,4,5,6-sides dice are showed in the following:



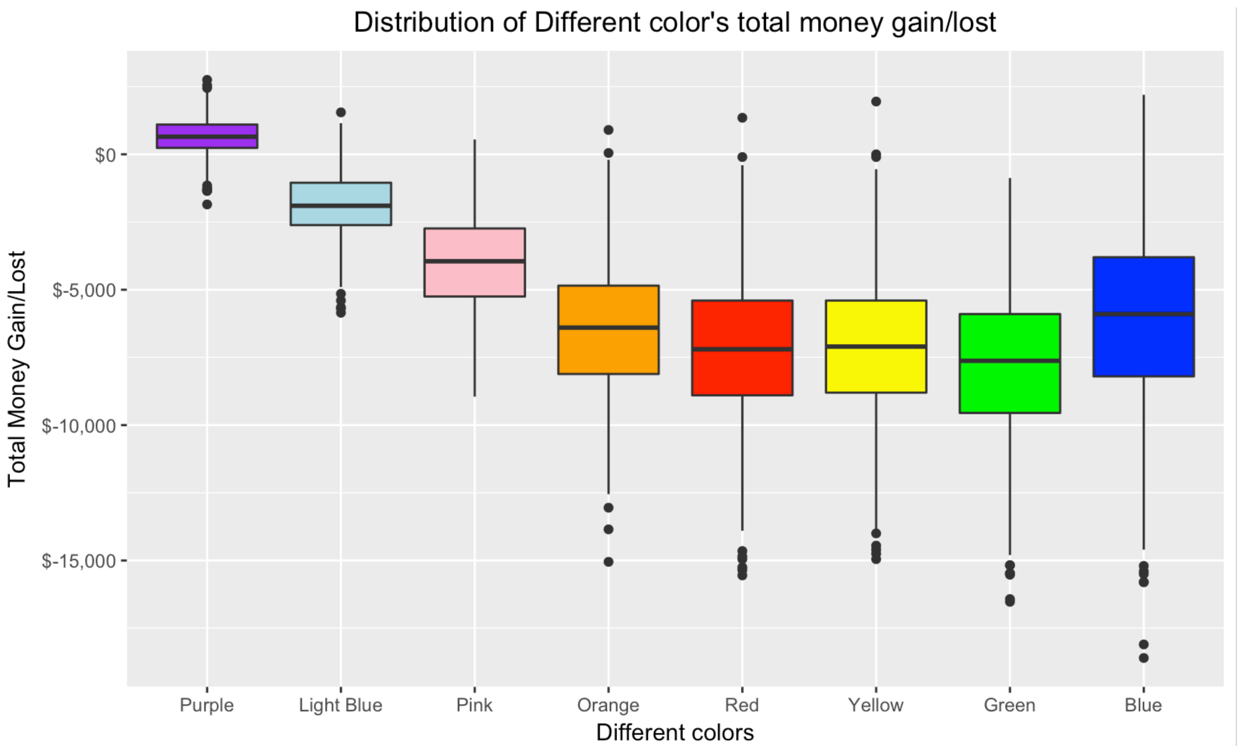
From this plot, it is clear to see that **square 10** always has the highest probability to be ended in a turn for all different sides dices. This is some rules improve the probability to land on **square 10**, such as if land on “G2J”, go to **Jail** immediately. Choosing cards in Chance and Community Chest can also go to **Jail** directly. In comparison, some squares have very low probability to land for all different sided dices, such as **square 36, square 7, and square 22.**

Another finding is that, as the sides of dices decrease, the probability of land on **square 10** tend to be increase because from the above graph, we can see that the color of **square 10** changes from orange to dark red as the sides of dices decrease. This means if play monopoly with a lower sided dice, it is more likely to go to the Jail.

Part1 Question 3

After 1000 times simulating with 10000 rolls and 6-sided dices, the stand error of ending a turn in Jail is 0.0021465.

Part2 Question 2



From this plot, the prerequisite is to assume the opponent own that kind of properties. Thus, if the total money gain/lost is very negative, it means that kind of properties is effective because we pay a lot to our opponent. In contrast, if the total money gain/lost is close to $0 or positive, that kind of properties is not effective. it is clear to see that the **Green properties** is the most effective when it has hotels because its distribution of Total Money Lost among all 8 colors is the most negative. The least effective property is the **Purple properties**, because it is the only one whose most simulations of total money gain/lost are positive. After checking the rules of monopoly, it is easy to explain these situations. The rent of Purple properties is the most cheaper among all 8 colors, so it is not strange that it becomes the least effective properties.

The **Blue properties** have the most expensive rents, but why they are not as effective as **Green properties**? The reason is that Blue properties only have 2 squares total but Green properties have 3 squares, which means it is more likely to land on Green properties rather than Blue properties in simulations. Therefore, if someone want to win in a monopoly game, he or she may try to buy Green properties to increase their probabilities to win.

Part2 Question 3

Distribution of 25 rolls

Color	Wins	Losses	Win %
Orange	468	232	.6686
Yellow	413	287	.59
Red	392	308	.56
Pink	348	352	.497
Blue	343	357	.49
Light Blue	323	377	.4614
Green	298	402	.4257
Purple	151	549	.2157

Distribution of 50 rolls

Color	Wins	Losses	Win %
Orange	482	218	.6886
Yellow	457	243	.65286
Red	442	258	.63143
Blue	398	302	.5686
Green	381	319	.5443
Pink	317	383	.4529
Light blue	223	477	.3186
Purple	41	659	.059

Distribution of 100 rolls

Color	Wins	Losses	Win %
Orange	507	193	.724
Yellow	493	207	.704
Red	458	242	.6543
Blue	429	271	.613
Green	419	281	.599
Pink	285	415	.407
Light blue	146	554	.209
Purple	4	696	.006

Comparing these three tables, we can find that in 25 roll's Distribution. There is no huge different between each color's probability of winning, even for **Purple,** there is still 21.57% to win. This is because 25 rolls are too small. Therefore, player1 and player2 may not land on each other's properties or land on it for 1 or 2 times. In this situation, the advantage of high rent properties cannot be showed very obviously.

However, if the rolls increase to 50 and 100, we can find that **the lower probability becomes extremely lower and the higher probability becomes even higher**. This is because as the rolls increase, the probability of land on each other's properties becomes higher, and this will make the players who own higher rents properties be more likely to win. We can see that in 50 rolls and 100 rolls distribution, the **Pink, Light blue and Purple** are always the lower 3 because they don't have rents advantages for their players.

The top properties probability to win is very close to each other. Therefore, to find an exactly rank, we may need a higher number of simulations and a rolls more than 100.

Appendix:

Citiation:

#discuss some ideas with classmates in office hour

#imitate some ideas and code Patrick post on Piazza

#learn how to change the x-axis and y-aixs in : https://stackoverflow.com/questions/10770550/r-how-to-edit-elements-on-x-axis-in-image-plot

#lean how to draw several boxplots together: https://stackoverflow.com/questions/37694234/two-boxplots-on-the-same-graph

code:

```
# STA141A HW3 Ruochen Zhong 912888970

install.packages("fields")
library(ggplot2)
library("fields")
library(plyr)

hw3money <- read.csv("/Users/apple/Desktop/properties.csv")
hw3board <- read.csv("/Users/apple/Desktop/color_combos.csv")

# ***** Part1 question 1 *****

simulate_monopoly <- function(n,d){
  #define the output value
  position <- numeric(n+1)
  position[1] = 0
  #define the card of CC, CH
  CC_card <- numeric(16)
```

```
CH_card <- numeric(16)
CC_card <- sample(1:16, replace = FALSE)
CH_card <- sample(1:16, replace = FALSE)
#define double roll
double_roll = 0
  for(i in 1:n) {
    dice1 <- sample(1:d, 1, replace = FALSE)
    dice2 <- sample(1:d, 1, replace = FALSE)

    distance <- dice1 + dice2
    position[i+1] = (position[i] + distance) %% 40

      # check the situation of three consecutive turns
      if (dice1 == dice2) {
            double_roll = double_roll + 1
      }
      else { double_roll = 0
      }

      # if there is 3 consecutive turns, send to jails
      if (double_roll == 3) {
            double_roll = 0
            position[i+1] = 10
      }

      # if land on G2J, move to Jail
      if (position[i+1] == 30) {
            position[i+1] = 10
      }

      # if land on a CC square, choose a CC_card
      if (position[i+1] == 2 || position[i+1] == 17 || position[i+1] == 33) {
              # choose the first CC_card
```

```
        CC_card[1]
        # special movement if choose 1 and 2
        if(CC_card[1] == 1) {
          position[i+1] = 0
        } else if(CC_card[1] == 2) {
          position[i+1] = 10
        } else { position[i+1] = position[i+1] }
      #put the first Card to the last one in the pile
      CC_card <- c(CC_card[-1],CC_card[1])
}

# if land on a CH square, choose a CH_card
if (position[i+1] == 7 || position[i+1] == 22 || position[i+1] == 36) {
  # choose the first CH_card
  CH_card[1]
  # special movement if choose 1 to 10
  if(CH_card[1] == 1) {
    position[i+1] = 0
  } else if(CH_card[1] == 2) {
    position[i+1] = 10
  } else if(CH_card[1] == 3) {
    position[i+1] = 11
  } else if(CH_card[1] == 4) {
    position[i+1] = 24
  } else if(CH_card[1] == 5) {
    position[i+1] = 39
  } else if(CH_card[1] == 6) {
    position[i+1] = 5
  } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 7) ) {
    position[i+1] = 15
  } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 22) ) {
    position[i+1] = 25
  } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 36) ) {
```

```
            position[i+1] = 5
          } else if((CH_card[1] == 9) && (position[i+1] == 7)) {
            position[i+1] = 12
          } else if((CH_card[1] == 9) && (position[i+1] == 22)) {
            position[i+1] = 28
          } else if((CH_card[1] == 9) && (position[i+1] == 36)) {
            position[i+1] = 12
          } else if(CH_card[1] == 10) {
            position[i+1] = position[i+1] - 3
          } else { position[i+1] = position[i+1] }

        #put the first Card to the last one in the pile
        CH_card <- c(CH_card[-1],CH_card[1])

      }


    }

  return(position)
}

# ***** Part1 question 2 *****
#write the function estimate_monopoly
estimate_monopoly <- function(n,d) {
   prob_table <- table(simulate_monopoly(n,d))
   prob_table <- ( prob_table / sum(prob_table))
   return(prob_table)
}

# if play 6-sided dice
six_sides <- estimate_monopoly(100000,6)
# find top 3 values
```

```
six_sides[order(six_sides, decreasing = TRUE)[1:3]]

# if play 4-sided dice
four_sides <- estimate_monopoly(100000,4)
# find top 3 values
four_sides[order(four_sides, decreasing = TRUE)[1:3]]

# if play with 5-sided dice
five_sides <- estimate_monopoly(100000,5)
#find top 3 values
five_sides[order(five_sides, decreasing = TRUE)[1:3]]

# if play with 3-sided dice
three_sides <- estimate_monopoly(100000,3)
three_sides[order(three_sides, decreasing = TRUE)[1:3]]

# ***draw bar plot of each sided dice***
# Use heatmap
# create four matrix for each sides
six <- as.matrix(six_sides)
five <- as.matrix(five_sides)
four <- as.matrix(four_sides)
three <- as.matrix(three_sides)

# combine them to be a whole matrix
distribution_mat <- t(cbind(six,five,four,three))

# draw the heat map, adjust the xlab and ylab, and add a legend
image(distribution_mat, axes=F)
image.plot(distribution_mat, legend.only=F, axes = F, main = "Different sided dices' position
Distribution")
squ_number <- (0:39)
# exclude the square 30 because there is no value on it
```

```
squ_number <- squ_number[squ_number!=30]
# change the x-axes and y-axes to make them appropriate
mtext(text=c(paste("square",squ_number)), side=2, line=0.3, at=seq(0,1,0.02631), las=1,
cex=0.8)
mtext(text=c(paste("sides dices", 6:3)), side=1, line=0.3, at=seq(0,1,0.3333), las=2, cex=0.8)


# ***** Part1 question 3 ***** #
# write a function to caculate the ten square's probability
jail_monopoly <- function(n,d) {
  n_sides <- as.data.frame(estimate_monopoly(n,d))
  ten_square <- n_sides[11,]
  print(ten_square$Freq)
}
# stimulate 1000 times with the probability
jail_simulate <- replicate(1000, jail_monopoly(10000,6))
# caculate the standard error
std_error <- sd(jail_simulate)

# ***** Part2 question 1 ****** #
# create two new vector
hw3money <- read.csv("/Users/apple/Desktop/properties.csv")
property <- hw3money$Index
rent <- hw3money$Rent

# rewrite the simulate_monopoly to be simulate_monopoly2
# steps with *** means new changes compare to simulate_monopoly

# ***new changes: add two more input, vector property and rent
simulate_monopoly2 <- function(n,d,property,rent) {
  #define the output value
  position <- numeric(n+1)
  position[1] = 0
```

```
#define the card of CC, CH
CC_card <- numeric(16)
CH_card <- numeric(16)
CC_card <- sample(1:16, replace = FALSE)
CH_card <- sample(1:16, replace = FALSE)
#define double roll
double_roll = 0
#*** new changes: define variable "wealth"
wealth <- numeric(n+1)
wealth[1] = 0

for(i in 1:n) {
  dice1 <- sample(1:d, 1, replace = FALSE)
  dice2 <- sample(1:d, 1, replace = FALSE)

  distance <- dice1 + dice2
  position[i+1] = (position[i] + distance) %% 40

  # check the situation of three consecutive turns
  if (dice1 == dice2) {
    double_roll = double_roll + 1
  }
  else { double_roll = 0
  }

  # if there is 3 consecutive turns, send to jails
  if (double_roll == 3) {
    double_roll = 0
    position[i+1] = 10
  }

  # if land on G2J, move to Jail
  if (position[i+1] == 30) {
```

```
    position[i+1] = 10
  }

  # if land on a CC square, choose a CC_card
  if (position[i+1] == 2 || position[i+1] == 17 || position[i+1] == 33) {
    # choose the first CC_card
    CC_card[1]
    # special movement if choose 1 and 2
    if(CC_card[1] == 1) {
      position[i+1] = 0
    } else if(CC_card[1] == 2) {
      position[i+1] = 10
    } else { position[i+1] = position[i+1] }
    #put the first Card to the last one in the pile
    CC_card <- c(CC_card[-1],CC_card[1])
  }

  # if land on a CH square, choose a CH_card
  if (position[i+1] == 7 || position[i+1] == 22 || position[i+1] == 36) {
    # choose the first CH_card
    CH_card[1]
    # special movement if choose 1 to 10
    if(CH_card[1] == 1) {
      position[i+1] = 0
    } else if(CH_card[1] == 2) {
      position[i+1] = 10
    } else if(CH_card[1] == 3) {
      position[i+1] = 11
    } else if(CH_card[1] == 4) {
      position[i+1] = 24
    } else if(CH_card[1] == 5) {
      position[i+1] = 39
    } else if(CH_card[1] == 6) {
```

```
        position[i+1] = 5
      } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 7) ) {
        position[i+1] = 15
      } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 22) ) {
        position[i+1] = 25
      } else if( (CH_card[1] == 7 || CH_card[1] == 8) && (position[i+1] == 36) ) {
        position[i+1] = 5
      } else if((CH_card[1] == 9) && (position[i+1] == 7)) {
        position[i+1] = 12
      } else if((CH_card[1] == 9) && (position[i+1] == 22)) {
        position[i+1] = 28
      } else if((CH_card[1] == 9) && (position[i+1] == 36)) {
        position[i+1] = 12
      } else if(CH_card[1] == 10) {
        position[i+1] = position[i+1] - 3
      } else { position[i+1] = position[i+1] }

      #put the first Card to the last one in the pile
      CH_card <- c(CH_card[-1],CH_card[1])


    }

    # ***New changes: Caculate money if goes to speicial properties, if not, keep the wealth the same
    if(position[i+1] %in% property){
      wealth[i+1]= -rent[match(position[i+1],property)]
    } else { wealth[i+1] = 0 }

    # ***New changes: if pass the "GO" instead of pass it by card, get $200
    if((position[i] + distance) > 39) {
      wealth[i+1] =    200
    }
```

```
    # ***New changes:if pass the "G0", but go to the jail in the end in that trail, Cannot get the $200!!!
    if((position[i] + distance > 39)&&(position[i+1]==10) ){
      wealth[i+1] = 0
    }
    # ***New changes: if pass "0", but go to properties, plus 200 firstly and minus rent
    if((position[i] + distance > 39) && (position[i+1] %in% property)){
      wealth[i+1]= 200 -rent[match(position[i+1],property)]
    }
    # ***New changes: if pass"0" and  land on T1 in the same step, plus 200 firstly and minus 200
    if((position[i] + distance > 39) && (position[i+1] == 4)){
      wealth[i+1] =   200 - 200
    }

    # ***New changes: if land on Taxes, lose money
    # For T1, if land on T1 without pass "0", loss $200
    if((position[i] + distance < 39) && (position[i+1] == 4)) {
      wealth[i+1] = - 200
    }
    # For T2, loss $100
    if (position[i+1] == 38) {
      wealth[i+1] = - 100
    }
  }

  # ***New changes:change the output to be both the positions and their corresponding wealth
  money <- data.frame(position, wealth)
  return(money)
}

# ***** Part2: Question 2 ***** #
# Define the input vector of all eight colors
```

Purple_color <- hw3money$Index[(which(hw3money$Color == "Purple"))]

Lightblue_color <- hw3money$Index[(which(hw3money$Color == "Light Blue"))]

Pink_color <- hw3money$Index[(which(hw3money$Color == "Pink"))]

Orange_color <- hw3money$Index[(which(hw3money$Color == "Orange"))]

Red_color <- hw3money$Index[(which(hw3money$Color == "Red"))]

Yellow_color <- hw3money$Index[(which(hw3money$Color == "Yellow"))]

Green_color <- hw3money$Index[(which(hw3money$Color == "Green"))]

Blue_color <- hw3money$Index[(which(hw3money$Color == "Blue"))]

# For Purple color's distribution
simulate_monopoly2(100,6,property = Purple_color, rent = rent[match(Purple_color,hw3money$Index)])
Purple <- replicate(1000, sum(simulate_monopoly2(100,6,property = Purple_color, rent = rent[match(Purple_color,hw3money$Index)])[,2]))

# For Light Blue's distribution
simulate_monopoly2(100,6,property = Lightblue_color, rent = rent[match(Lightblue_color,hw3money$Index)])
Light_Blue <- replicate(1000, sum(simulate_monopoly2(100,6,property = Lightblue_color, rent = rent[match(Lightblue_color,hw3money$Index)])[,2]))

# For Pink's distribution
simulate_monopoly2(100,6,property = Pink_color, rent = rent[match(Pink_color,hw3money$Index)])
Pink <- replicate(1000, sum(simulate_monopoly2(100,6,property = Pink_color, rent =

```
rent[match(Pink_color,hw3money$Index)])[,2]))

# For Orange's distribution
simulate_monopoly2(100,6,property = Orange_color, rent = 
rent[match(Orange_color,hw3money$Index)])
Orange <- replicate(1000, sum(simulate_monopoly2(100,6,property = Orange_color, rent = 
rent[match(Orange_color,hw3money$Index)])[,2]))

# For Red's distribution
simulate_monopoly2(100,6,property = Red_color, rent = 
rent[match(Red_color,hw3money$Index)])
Red <- replicate(1000, sum(simulate_monopoly2(100,6,property = Red_color, rent = 
rent[match(Red_color,hw3money$Index)])[,2]))

# For Yellow's distribution
simulate_monopoly2(100,6,property = Yellow_color, rent = 
rent[match(Yellow_color,hw3money$Index)])
Yellow <- replicate(1000, sum(simulate_monopoly2(100,6,property = Yellow_color, rent = 
rent[match(Yellow_color,hw3money$Index)])[,2]))

# For Green's distribution
simulate_monopoly2(100,6,property = Green_color, rent = 
rent[match(Green_color,hw3money$Index)])
Green <- replicate(1000, sum(simulate_monopoly2(100,6,property = Green_color, rent = 
rent[match(Green_color,hw3money$Index)])[,2]))

# For Blue's distribution
simulate_monopoly2(100,6,property = Blue_color, rent = 
rent[match(Blue_color,hw3money$Index)])
Blue <- replicate(1000, sum(simulate_monopoly2(100,6,property = Blue_color, rent = 
rent[match(Blue_color,hw3money$Index)])[,2]))

# create dataframe for each colors 1000 simulations
```

```
Purple_dt <- as.data.frame(Purple)
LBlue_dt <- as.data.frame(Light_Blue)
Pink_dt <- as.data.frame(Pink)
Orange_dt <- as.data.frame(Orange)
Red_dt <- as.data.frame(Red)
Yellow_dt <- as.data.frame(Yellow)
Green_dt <- as.data.frame(Green)
Blue_dt <- as.data.frame(Blue)
# combine all colors dataframe together and create a features which can divide them
total_data <- rbind(cbind(stack(Purple_dt), group='Purple'), cbind(stack(LBlue_dt), group='Light Blue'),
                    cbind(stack(Pink_dt), group='Pink'), cbind(stack(Orange_dt), group='Orange'),
                    cbind(stack(Red_dt), group='Red'), cbind(stack(Yellow_dt), group='Yellow'),
                    cbind(stack(Green_dt), group='Green'), cbind(stack(Blue_dt), group='Blue')
)

# draw the seperate boxplot of all distributions in one graph
graph2_2 <- ggplot(total_data, aes(group, values)) +
  geom_boxplot(fill = c("purple","light blue","pink","orange","red","yellow","green","blue"))
+
  ggtitle("Distribution of Different color's total money gain/lost") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = scales::dollar) +
  labs(x = "Different colors", y = "Total Money Gain/Lost")

print(graph2_2)

#*******Part2 Question3********#

# create a fucntion to automatically assign those 28 situations
```

```
cost <- hw3money$Cost
compare_player <- function(n,d){
  player_1 <- numeric(28)
  player_2 <- numeric(28)
  for (i in 1:28) {
    #for each players who own a different color
    player2 <- simulate_monopoly2(n,d,property =
hw3money$Index[(which(hw3money$Color == as.character(hw3board[i,1])))], rent =
rent[match((hw3money$Index[(which(hw3money$Color == as.character(hw3board[i,1])))]),
hw3money$Index)])
    player1 <- simulate_monopoly2(n,d,property =
hw3money$Index[(which(hw3money$Color == as.character(hw3board[i,2])))], rent =
rent[match((hw3money$Index[(which(hw3money$Color == as.character(hw3board[i,2])))]),
hw3money$Index)])
    #///player1's total money = $5000 + the rest after paying to player2 "sum((player1)[,2])"
+
    #///revenue received from player2 "sum(player2$wealth[which(player2$wealth < 0)])" +
the cost of buying house
    # for player2, the same
    player_1[i] <- sum((player1)[,2]) - sum(player2$wealth[which(player2$wealth < 0)]) -
(hw3money$Cost[(which(hw3money$Color == as.character(hw3board[i,1])))][1]) + 5000
    player_2[i] <- sum((player2)[,2]) - sum(player1$wealth[which(player1$wealth < 0)]) -
(hw3money$Cost[(which(hw3money$Color == as.character(hw3board[i,2])))][1]) + 5000
  }
  compare <- cbind(player_1,player_2)
  return(compare)
}

# create a new function to caculate the times of wins in simulation
win_table <- function(n,d){
  # call compare_player firstly
  M <- compare_player(n,d)
  # define all 8 colors win times to be 0 firstly
```

```
Purple_win = 0
L_blue_win = 0
Pink_win = 0
OR_win = 0
Red_win = 0
Yellow_win = 0
Green_win = 0
Blue_win = 0
# logic for each conditions based on the "color_combo,csv" which is a 28*2 matrix
for(i in 1:7){

    if((M[i,1]) > (M[i,2])){
    Purple_win = Purple_win + 1
    }
}
    if((M[1,1]) < (M[1,2])){
    L_blue_win = L_blue_win + 1
    }
     if((M[2,1]) < (M[2,2])){
    Pink_win = Pink_win + 1
    }
     if((M[3,1]) < (M[3,2])){
    OR_win = OR_win + 1
    }
     if((M[4,1]) < (M[4,2])){
     Red_win = Red_win + 1
    }
     if((M[5,1]) < (M[5,2])){
     Yellow_win = Yellow_win + 1
    }
     if((M[6,1]) < (M[6,2])){
     Green_win = Green_win + 1
    }
```

```
    if((M[7,1]) < (M[7,2])){
    Blue_win = Blue_win + 1
   }




for(i in 8:13){

   if((M[i,1]) > (M[i,2])){
   L_blue_win = L_blue_win + 1
   }
}

   if((M[8,1]) < (M[8,2])){
     Pink_win = Pink_win + 1
   }
   if((M[9,1]) < (M[9,2])){
     OR_win = OR_win + 1
   }
   if((M[10,1]) < (M[10,2])){
     Red_win = Red_win + 1
   }
   if((M[11,1]) < (M[11,2])){
     Yellow_win = Yellow_win + 1
   }
   if((M[12,1]) < (M[12,2])){
     Green_win = Green_win + 1
   }
   if((M[13,1]) < (M[13,2])){
     Blue_win = Blue_win + 1
   }
```

```
for(i in 14:18){

    if((M[i,1]) > (M[i,2])){
    Pink_win = Pink_win + 1
    }
}
      if((M[14,1]) < (M[14,2])){
      OR_win = OR_win + 1
      }
      if((M[15,1]) < (M[15,2])){
      Red_win = Red_win + 1
      }
      if((M[16,1]) < (M[16,2])){
      Yellow_win = Yellow_win + 1
      }
      if((M[17,1]) < (M[17,2])){
      Green_win = Green_win + 1
      }
      if((M[18,1]) < (M[18,2])){
      Blue_win = Blue_win + 1
      }



for(i in 19:22){
  if((M[i,1]) > (M[i,2])){
    OR_win = OR_win + 1
   }
 }
     if((M[19,1]) < (M[19,2])){
      Red_win = Red_win + 1
```

```
    }
  if((M[20,1]) < (M[20,2])){
   Yellow_win = Yellow_win + 1
    }
  if((M[21,1]) < (M[21,2])){
   Green_win = Green_win + 1
    }
  if((M[22,1]) < (M[22,2])){
   Blue_win = Blue_win + 1
    }



for(i in 23:25){
  if((M[i,1]) > (M[i,2])){
    Red_win = Red_win + 1
  }
}
  if((M[23,1]) < (M[23,2])){
  Yellow_win = Yellow_win + 1
  }
  if((M[24,1]) < (M[24,2])){
   Green_win = Green_win + 1
  }
  if((M[25,1]) < (M[25,2])){
   Blue_win = Blue_win + 1
  }



for(i in 26:27){
  if((M[i,1]) > (M[i,2])){
```

```
        Yellow_win = Yellow_win + 1
      }
    }
              if((M[26,1]) < (M[26,2])){
              Green_win = Green_win + 1
              }
              if((M[27,1]) < (M[27,2])){
              Blue_win = Blue_win + 1
              }




    for(i in 28){
      if((M[i,1]) > (M[i,2])){
        Green = Green_win + 1
      }
    }
    if((M[28,1]) < (M[28,2])){
      Blue_win = Blue_win + 1
    }



    table <- cbind(Purple_win, L_blue_win, Pink_win, OR_win, Red_win, Yellow_win,
Green_win, Blue_win)
    return(table)
}

# return three different simulations' dataframe
twenty_five_times <- rdply(100, win_table(25,6))

fifty_times <- rdply(100, win_table(50,6))
```

```
hundred_times <- rdply(100, win_table(100,6))

# caculate the total win times of each color out of 700 times each
colSums(twenty_five_times)

colSums(fifty_times)

colSums(hundred_times)

# rank each of them, find who has the most probability to win, who has the least
sort(colSums(twenty_five_times), decreasing = TRUE)

sort(colSums(fifty_times), decreasing = TRUE)

sort(colSums(hundred_times), decreasing = TRUE)
# more information of this question is in the report.
```