STA141AHW4

Sam Tsoi

11/18/2017

1.

Write a function simulate_monopoly() that simulates n turns by a player in a game of Monopoly using two d-sided dice. The inputs to your function should be n and d. The output of your function should be a length n + 1 vector of positions, encoded as numbers from 0 to 39.

Answer is attached in the code.

2.

Write a function estimate_monopoly() that uses your simulation to estimate the long-term probabilities of ending a turn on each Monopoly square. What are the 3 most likely squares to end a turn on if you play Monopoly with 6-sided dice? What if you play with 4-sided dice? Display graphically the long-term probabilities for 3, 4, 5, and 6-sided dice.
Answer is attached in the code.

The indices of 3 most likely squares to end a turn on with a 6-sided dice, with their respective probabilities are:

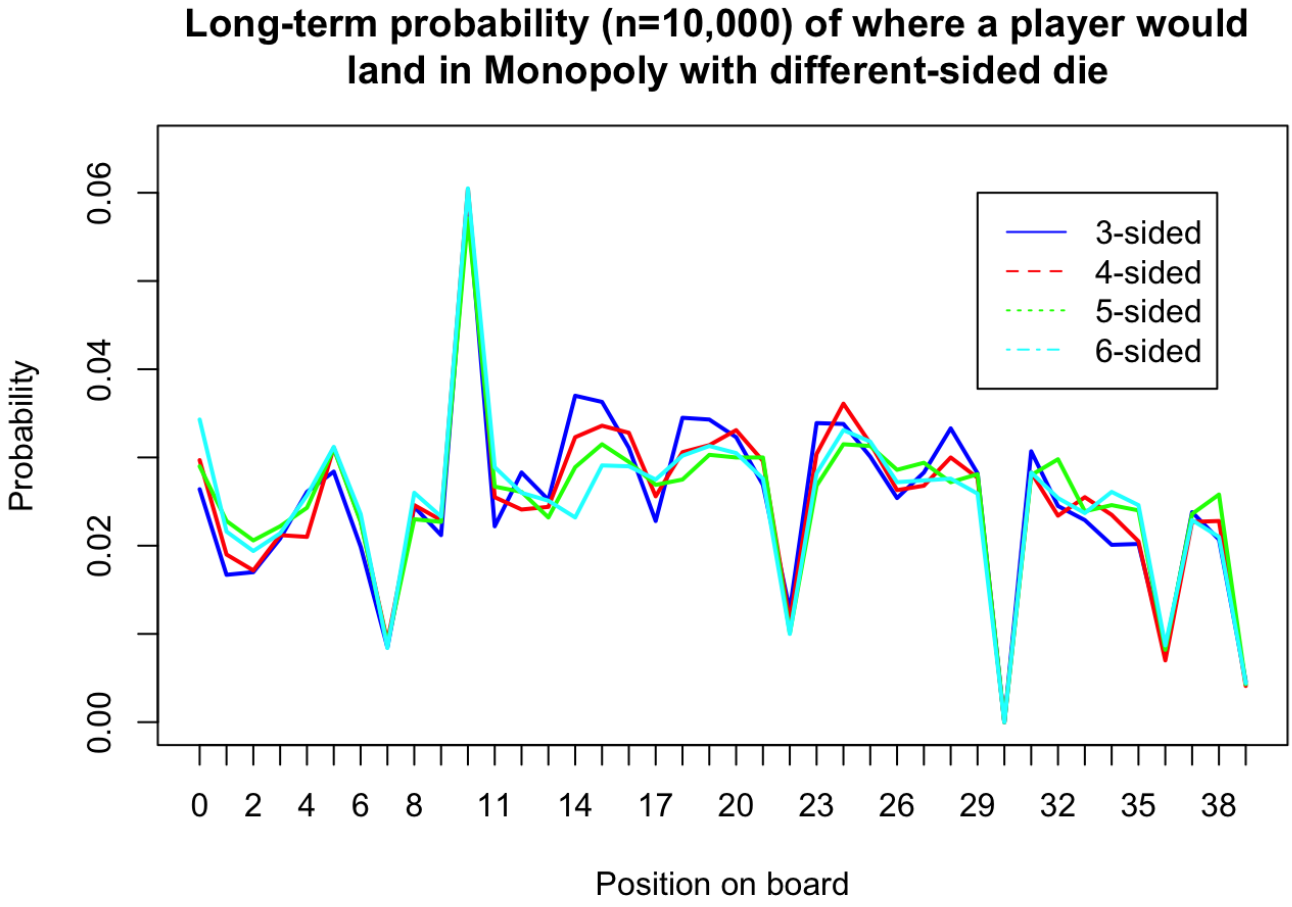
```
## rollResult
##         10         26         12
## 0.05194805 0.04195804 0.03396603
```

So, the 3 most likely squares are jail, F1, and UT1.

The 3 most likely squares to end a turn on with a 4-sided dice, with their respective probabilities are:

```
## rollResult
##         10         21         19
## 0.06193806 0.03496503 0.03396603
```

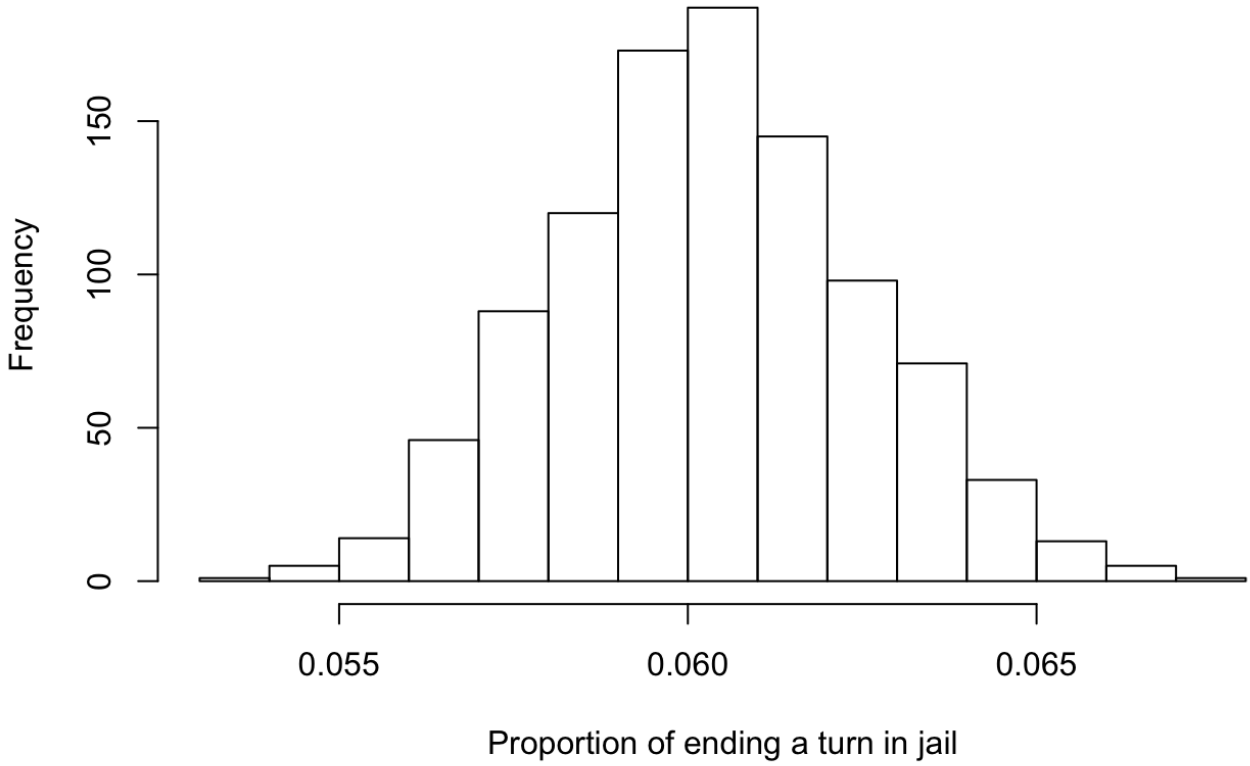
So, the 3 most likely squares are jail, E1, and D3.



From this graph, it seems like for all different-sided die, the probability of ending a turn in jail is higher than the rest of the cells in Monopoly. There are other common patterns, like there are no propobability of a player ending a turn on tile 30, which automatically moves the player to jail, which is tile 10. (It was important to make sure includes 0 in making a table and displaying graph, because it drops them automatically). Other observations are that proportion of cell 7, which is CH1 (a chance card) is quite low. This means that the chances are ending a turn actually at a chance spot is low, which makes sense, because 62.5% of the time one is on a chance tile, he or she would have to move their piece to another spot and ending their turn in that spot. This is similar to cell 22, which is another Chance spot, and also cell 36. There are also dips in proportion for cell 2, cell 17, and cell 33. This is because these are community chests and it has a 5% chance of moving after being on that tile. These cards, therefore, make some tiles being more visited than others. As mentioned, jail has the highest probability, but cell 11,24,39,5,15,25,12,28, and Go are all tile numbers that have a little higher chance of a player ending up there, because of these cards and as shown on this graph. Each run for different-sided die vary, sometimes one certain dice would have more drastic dips and another wouldn't, but it all depends on chance. The larger the n (turns), the more uniform it is across all different-sided die, which makes sense, because standard error would decrease with more turns.

3.

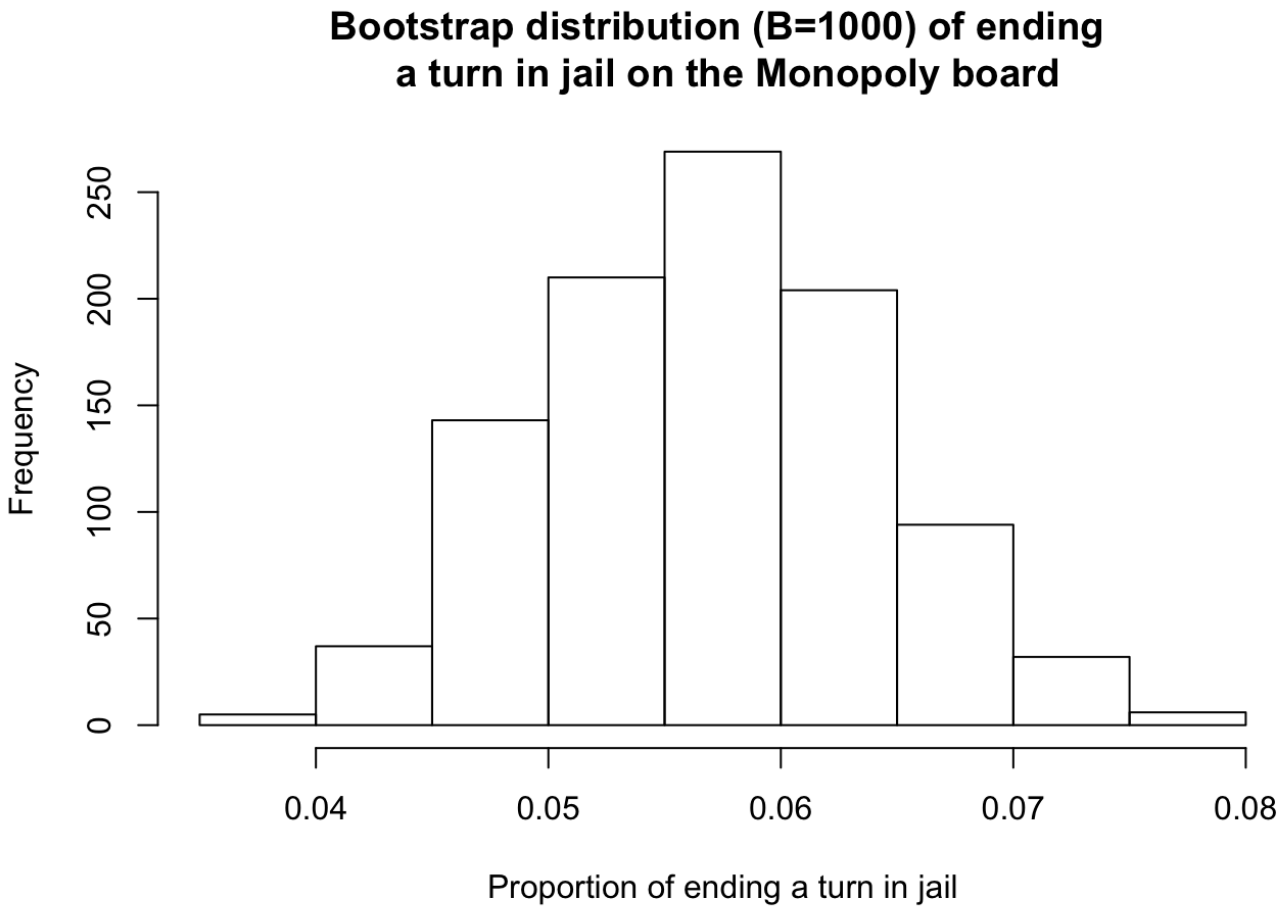
Use k = 1, 000 simulations with n = 10, 000 turns each to estimate the standard error for the long-term probability of ending a turn in jail.



Thus, the standard error for the long-term probability of ending a turn in jail, with a six-sided dice, is 0.0022046. The distribution seems pretty normal and it does not seemed to be skewed. The mean of ending a turn in jail is 0.0603618 and the median is 0.060294. They are pretty close to one another, which illustrates that the distribution isn't particularly skewed. As shown from the probability in the previous question, the frequency of a player ending a turn in jail is higher than the other tiles, for any-sided dice.

4.

Use the non-parametric bootstrap with b = 1, 000 samples from a simulation of n = 10, 000 turns to estimate the standard error for the long-term probability of ending a turn in jail.



Thus, the standard error for the long-term probability of ending a turn in jail for a B=1000 bootstrap distribution, with a six-sided dice, is 0.0071717. The distribution seems pretty normal and it does not seemed to be skewed. The mean of ending a turn in jail is 0.0603618 and the median is 0.060294. They are pretty close to one another, which illustrates that the distribution isn't particularly skewed. As shown from the probability in the previous question, the frequency of a player ending a turn in jail is higher than the other tiles, for any-sided dice.

a. How does the boostrap estimate compare to the simulation estimate? Which do you think is more accurate? Explain.

Looking at the standard errors of the two estimates, bootstrap estimate is closer to the predicted true probability than simulation estimate, with the standard errors as 0.0022046 and 0.0071717, respectively. The simulation estimate actually run the simulation 1000 times with 10,000 turns, and then estimates the standard error from the simulations. The standard error for the simulation estimate compares and calculates the deviation from the estimated true probability, which is sampled from the simulation 1,000 times. Bootstrap estimate runs the simulation once with 10,000 turns, but samples from that simulation 1000 times. Therefore, the bootstrap estimate is only limited to that one sample it is drawing from, so it makes sense that all bootstraps would be similar (thus the standard error low) since each sample is drawing from the likelihood of tiles from the same simulation. So if, by chance, there was something that happened out of the ordinary in that one simulation ran for bootstrap, the 1000 samples made from that simulation might not represent the long-term probability entirely for the cells but the deviation from other samples from that the probability of that one simulation should be very low.

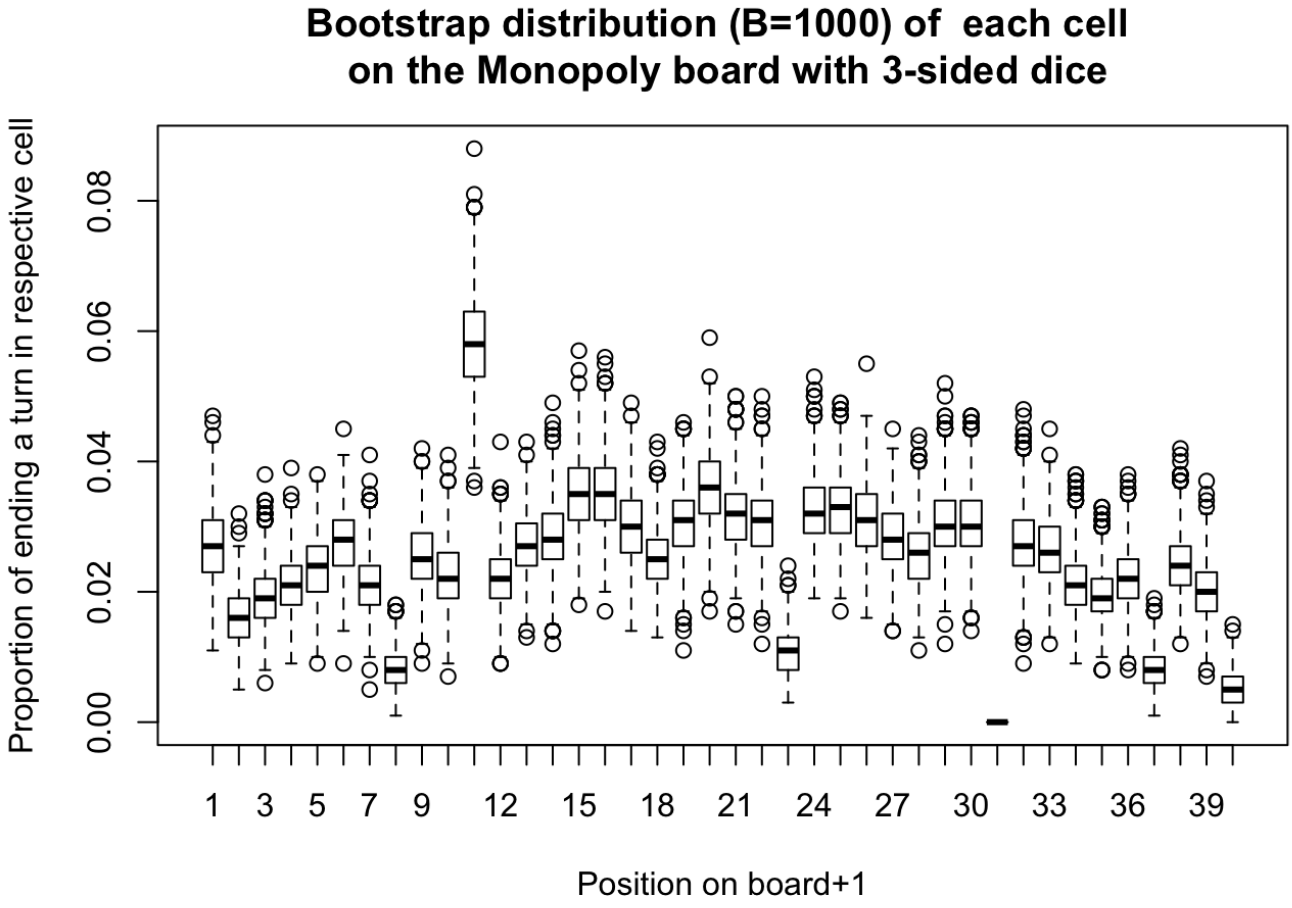
b. Which is faster to compute: the bootstrap estimate or the simulation estimate? Explain why there is a difference.

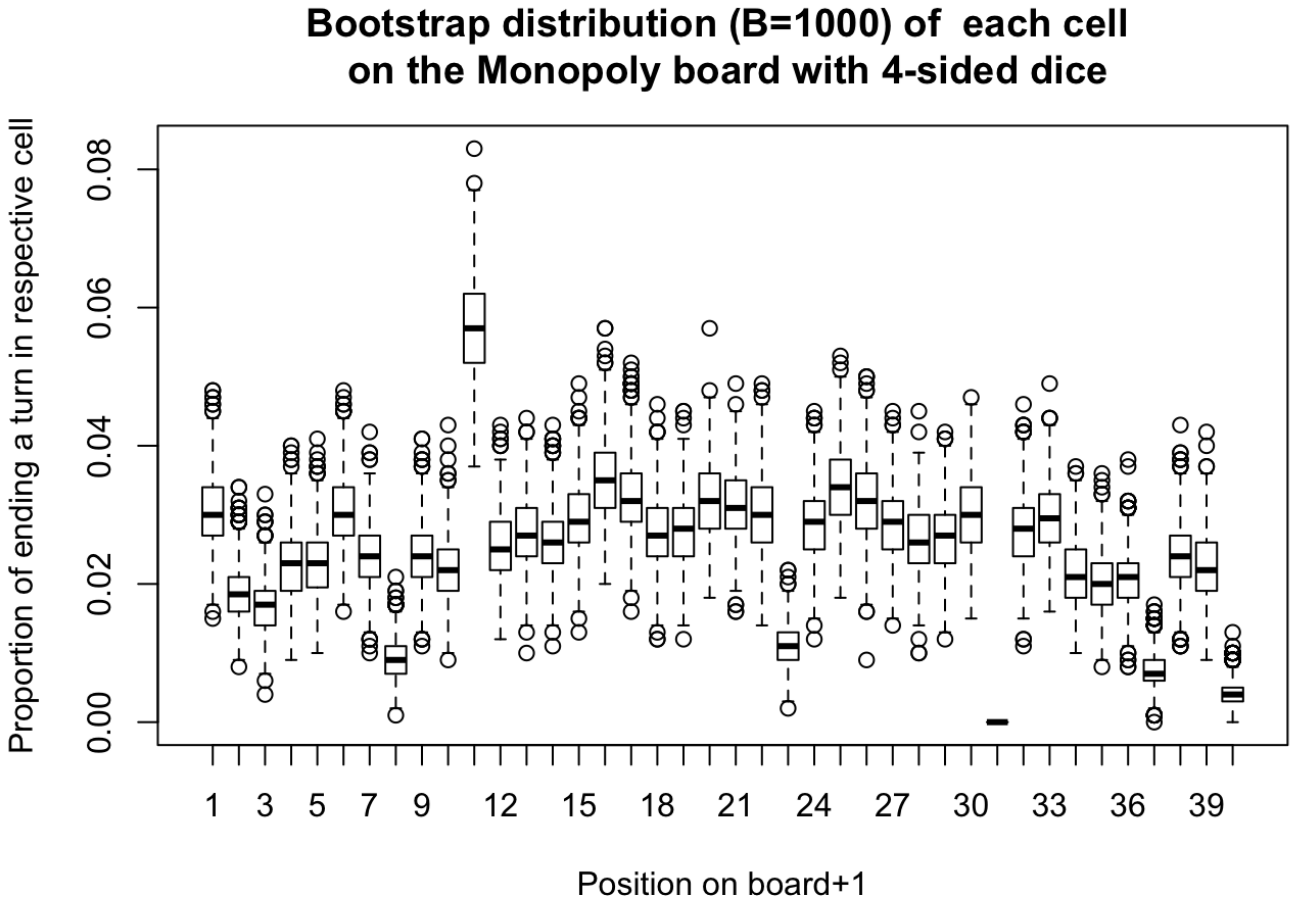
The bootstrap estimate is much faster to compute compared with the simulation estimate, because the simulation estimate requires to code to run simulate_monopoly() 1000 times to compare the proportion of

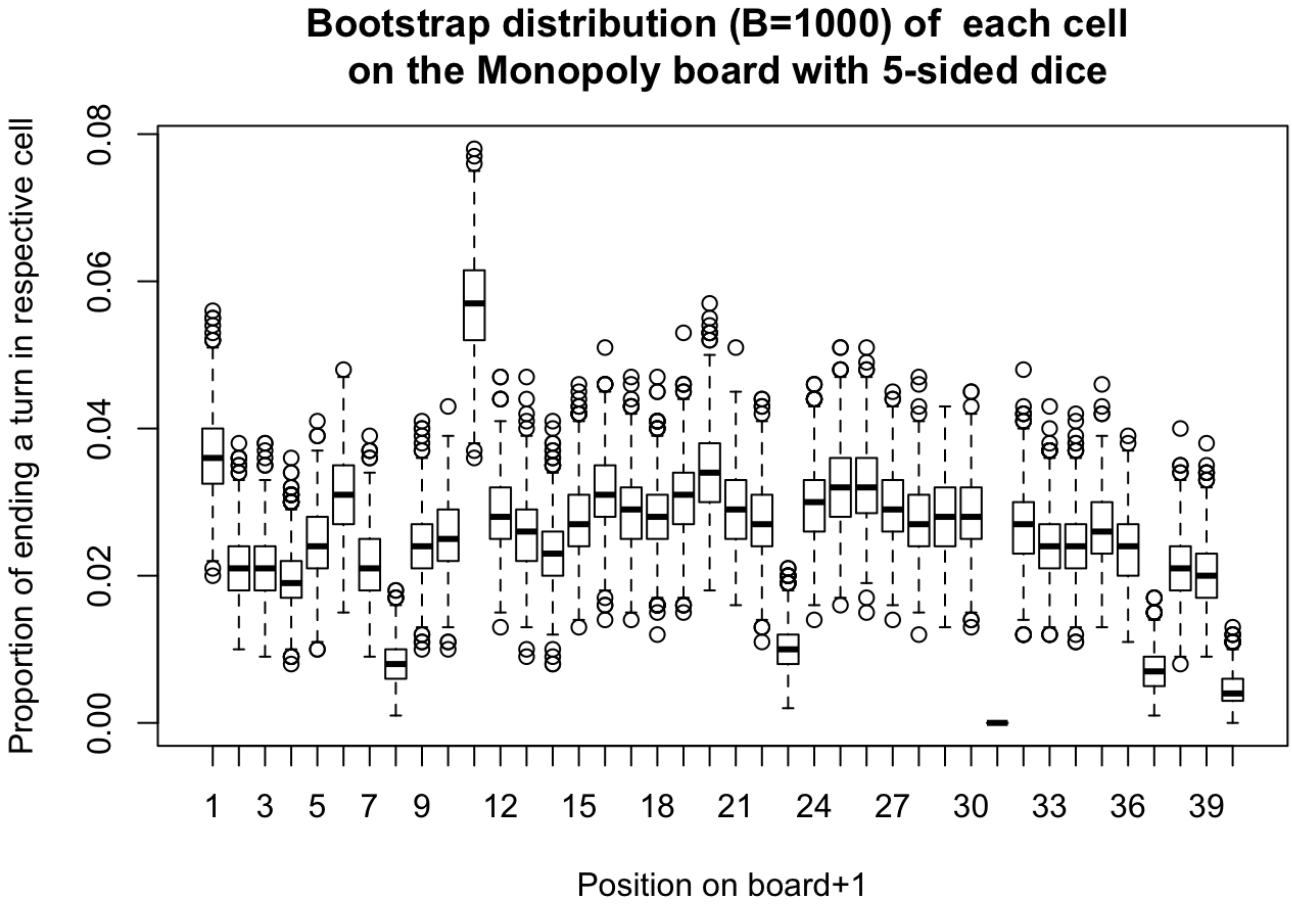
ending a turn in jail for each simulation of 10,000 turns, whereas the bootstrap estimate only requires the code to run simulate _monopoly() once. From that one simulation, you will have to sample that simulation 1000 times with replacement, which doesn't require running the whole simulation over and over.

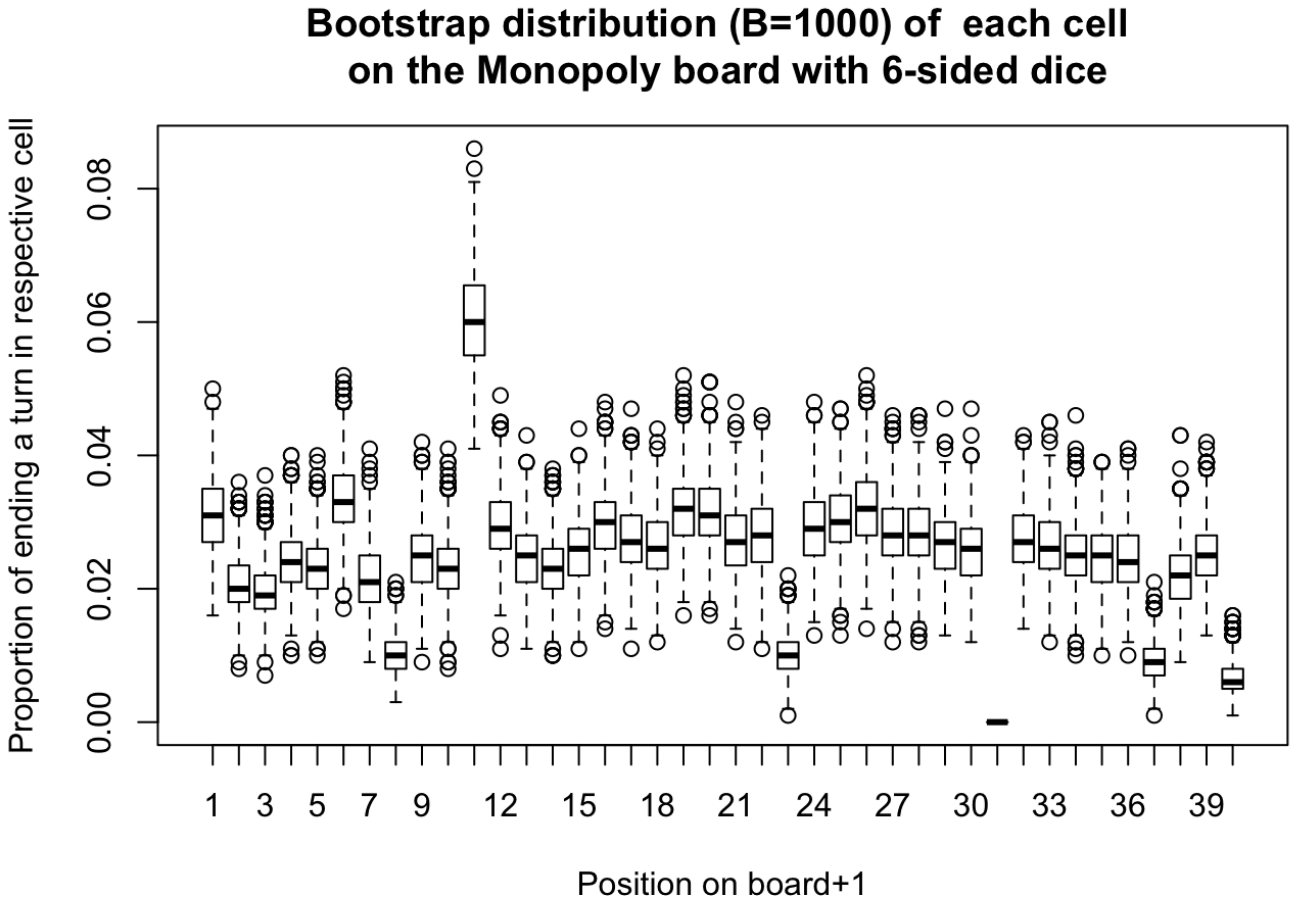
5.

Display graphically the standard errors for the long-term probabilities for 3, 4, 5, and 6-sided dice (use the same settings you used in question 2). Discuss why some probabilities have much larger standard errors than others.




Bootstrap distribution (B=1000) of each cell
on the Monopoly board with 4-sided dice
Proportion of ending a turn in respective cell
0.00
0.02
0.04
0.06
0.08
1
3
5
7
9
12
15
18
21
24
27
30
33
36
39
Position on board+1


Bootstrap distribution (B=1000) of each cell
on the Monopoly board with 5-sided dice
Proportion of ending a turn in respective cell
0.00
0.02
0.04
0.06
0.08
1
3
5
7
9
12
15
18
21
24
27
30
33
36
39
Position on board+1



An observation for all of these graphs is that the standard error for tile 30 (31 in the graph) is 0. This is because there is no chance that a player would end their turn on G2J. They will automatically go to cell 10 if they roll to cell 30. The standard error for cell 10, which is the jail cell, seems to be larger than others. This is probably because is it difficult to predict the frequency of ending up on a community chance or on the chance tile AND also moving to another tile. There is a lot of deviation here because it really depends. As seen, this also makes the standard error/quartiles smaller on cell 2,17,and 33, and 7,22,and 36, because these tiles are community chests and chance tiles, and the variation of landing in these tiles is much much lower than the other tiles that do not require a player to move their piece to another cell before ending a turn. These patterns are observed across different-sided die. I would say it is difficult to say whether different sided die differ from each other in terms of proportin of ending a turn in certain cells. It doesn't seem like it on these graphs, but further testing is required to make a more informed conclusion!

6.

What happens to the standard errors for the long-term probability estimates as n increases? Why does this happen?

As n increases, the standard errors for the long-term probability decreases, because having more turns would give less variation in the results. You can think of it as: the more turns, the more confident you are about your results because the probability of each cell would be closer to the true probability. Less turns would allow more variation, and might give probability that is only true to that sample/those turns, but not representative of the true probability.

7.

Please list the names of your final project group members here. Some details about the final project are at the end of this document.

Brody Lowry

Citations:

Looked on Piazza for tips on simulate_monopoly() and estimate _monopoly() Consulted dicussion notes and my STA 104 notes for bootstrap code

Code Appendix

```
library(ggplot2)
##1
simulate_monopoly <- function(n,d)
{
  currRoll <- numeric(2)
  currRollSum <- 0
  spot = 0
  position <- numeric(n+1)
  roll <-numeric(n+1)
  i = 1
  for(i in 1:n+1)
  {
    currRoll <- sample(1:d, 2, replace=T)
    currRollSum <- sum(currRoll)
    currPos <- ((position[i-1] + currRollSum) %%39)

    chProb <- c(rep((1/16),10), 6/16) #probability of CH 10/16
    if (currPos == 30) #G2J
    {
      currPos <- 10
    }
    else if (currPos == 2 || currPos == 17 || currPos == 33)#CC
    {
      currPos <- sample(c(0,10, currPos), 1, prob = c(1/16,1/16,14/16)) #advance t
o GO:1/16 or go to JAIL:1/16 or stay:14/16
    }
    else if(currPos == 7) #CH1
    {
      currPos <- sample(c(0,10,11,24,39,5,15,15,12,currPos-3,currPos),1, prob = c
hProb)
    }
    else if(currPos == 22) #CH2
    {
      currPos <- sample(c(0,10,11,24,39,5,25,25,28,currPos-3,currPos),1, prob = c
hProb)
    }
    else if(currPos == 36) ##CH3
    {
      currPos <- sample(c(0,10,11,24,39,5,5,5,12,currPos-3,currPos),1, prob = chPr
ob)
    }

    position[i] <- currPos
  }
  return(position)
}
##2
estimate_monopoly <- function(n,d)
{
  rollResult <- simulate_monopoly(n,d)
 # prop <- prop.table(table(rollResult))
  prop <- prop.table(table(factor(rollResult, levels = c(0:39)))) #we need factor(
) so it doesn't drop the 0 for tile 30 which is G2J
```

```
  return(prop)
}
longtermprob3 <- estimate_monopoly(10000,3)
longtermprob4 <- estimate_monopoly(10000,4)
longtermprob5 <- estimate_monopoly(10000,5)
longtermprob6 <- estimate_monopoly(10000,6)
mostlikely6 <- sort(longtermprob6, decreasing=TRUE)
mostlikely6[1:3]
mostlikely4 <- sort(longtermprob4, decreasing=TRUE)
mostlikely4[1:3]
plot(longtermprob3,type ="l",col="blue",xlab = "Position on board", ylab= "Probabil
ity", ylim = c(0,0.065), main = "Long-term probability (n=10,000) of where a playe
r would \n land in Monopoly with different-sided die")
lines(longtermprob4, type="l", col="red")
lines(longtermprob5, type="l", col ="green")
lines(longtermprob6, type = "l", col = "cyan")
legend(x=29,y=.06, legend =c("3-sided", "4-sided","5-sided","6-sided"), col=c("blu
e","red","green","cyan"),lty=1:4)


##3
simulations <- replicate(1000, estimate_monopoly(10000,6))
hist(simulations[11,], main="distribution of ending a turn in jail", xlab = "Propor
tion of ending a turn in jail")
se <- sd(simulations[11,])
mean <- mean(simulations[11,])
median <- median(simulations[11,])
##4
mono6 <- simulate_monopoly(10000,6)
boot<-function(){
boot.jail = sample(mono6,1000,replace = TRUE)
jail.prop<-length(which(boot.jail==10))/1000
return(jail.prop)
}
temp<-boot()
B = replicate(1000,boot())

hist(B, main = "Bootstrap distribution (B=1000) of ending \n a turn in jail on the
Monopoly board", xlab = "Proportion of ending a turn in jail")
bootse<-sd(B)
bootmean<-mean(B)
bootmedian <- median(B)
##5
mono3 <- simulate_monopoly(10000,3)
mono4 <- simulate_monopoly(10000,4)
mono5 <- simulate_monopoly(10000,5)
mono6 <- simulate_monopoly(10000,6)
boots<-function(d){
allcellprop <- c()
boot.allcell = sample(d,1000,replace = TRUE)
i = 0
for(i in 0:39)
{
  allcellprop <- c(allcellprop, length(which(boot.allcell==i))/1000)
}
allcellpropdf <- data.frame(allcellprop)
return(allcellpropdf)
```

```
}
#boots6 <- boots(mono6)
B3 = replicate(1000,boots(mono3))
B4 = replicate(1000,boots(mono4))
B5 = replicate(1000,boots(mono5))
B6 = replicate(1000,boots(mono6))
B3df <- t(data.frame(B3))
B4df <- t(data.frame(B4))
B5df <- t(data.frame(B5))
B6df <- t(data.frame(B6))


boxplot(B3df, main = "Bootstrap distribution (B=1000) of  each cell \n on the Monop
oly board with 3-sided dice", xlab = "Position on board+1", ylab = "Proportion of e
nding a turn in respective cell")
boxplot(B4df, main = "Bootstrap distribution (B=1000) of  each cell \n on the Monop
oly board with 4-sided dice", xlab = "Position on board+1", ylab = "Proportion of e
nding a turn in respective cell")
boxplot(B5df, main = "Bootstrap distribution (B=1000) of  each cell \n on the Monop
oly board with 5-sided dice", xlab = "Position on board+1", ylab = "Proportion of e
nding a turn in respective cell")
boxplot(B6df, main = "Bootstrap distribution (B=1000) of  each cell \n on the Monop
oly board with 6-sided dice", xlab = "Position on board+1", ylab = "Proportion of e
nding a turn in respective cell")
```