BAS 471 Spring 2023 Homework on Unit 2A - Monte Carlo Simulation

Ryan Curling

2/9/2023

Note: these are your homework problems

Reminder about collaboration policy. You can develop a common set of R code with you and your friends, provided you all contributed substantially to its creation. However, anything that is written interpretation, i.e., anything that follows a **Response:** needs to be written up in your own words. Homeworks that look to be near copy/pastes of each other will receive substantially reduced credit.

Instructions

Questions 1-6 are worth 25 points each. You can complete as many as you'd like (partial credit is available). This homework is graded out of 100 points, so this gives you the opportunity to earn up to 50 bonus points by completing all questions.

Question 7 is a very difficult bonus Monte Carlo simulation that can give an additional 25 extra credit points. This simulation may take a long time to develop, so you can turn in this problem separately at any point before the midterm exam to earn these points.

Question 1: Halloween Candy

Unable to be at home on Halloween nights, you leave a bowl filled with 120 pieces of candy. Although kids are instructed to take just one piece, imagine they take between 1 and 6 pieces with probabilities:

Candies	Probability
1	0.76
2	0.09
3	0.06
4	0.04

Candies	Probability
5	0.03
6	0.02

a) Using a Monte Carlo simulation, estimate the probability that the first 25 kids take a total of 40 or more candies. Develop your code so that it works for 10 trials, then obtain a final estimate of the probability with 100,000 trials. Sanity check: about 42%.

```
ntrials <- 100000
counter <- 0



for (trials in 1:ntrials) {
  values <- sample(1:6 , size = 25 , replace = TRUE , prob=
c(0.76,0.09,0.06,0.04,0.03,0.02))
  candy_tot <- sum(values)
  if(candy_tot >= 40) {counter <- counter + 1 }
}

counter/ntrials
## [1] 0.41889
```

b) Using a Monte Carlo simulation, estimate the probability that at least 90 kids get to take candy from the bowl. Develop your code so that it works for 10 trials, then obtain a final estimate of the probability with 100,000 trials. Note: you may run into a situation where, say, two candies are left in the bowl, but the simulation has the next kid take four. In that situation, the next kid gets the final two pieces! One approach: consider the total amount of candies that the first 89 kids take… Sanity check: about 3.9%

```
prob_candies <- c(0.76,0.09,0.06,0.04,0.03,0.02)
tot_kids <- 90
ntrials <- 100000
counter <- 0

for (i in 1:ntrials) {
  candy_taken <- sample(1:6 , size = 90 , replace = TRUE , prob=
prob_candies)
  candy_tot <- sum(candy_taken)
  if(candy_tot <= 120 ) {counter <- counter + 1 }
}

counter/ntrials
## [1] 0.03524
```

Question 2: Call center

The daily number of calls coming into a call center varies from 1 to 250. An equation that does a good job capturing the relative frequencies ("weights") of these possible values is: `x*exp(-x/40)`, i.e.:

$$Weight = xe^{-x/40} \qquad x = 1, 2, \ldots 249, 250$$

Note: if the equation above doesn't pop up, hover your mouse over it so that it displays.

For example, 50 calls gets a weight of `50*exp(-50/40) = 14.3`, and 200 calls gets a weight of `200*exp(-200/40) = 1.35`. This means that the call center receives 50 calls about 14.3/1.35 = 10.6 times more often than it receives 200 calls.

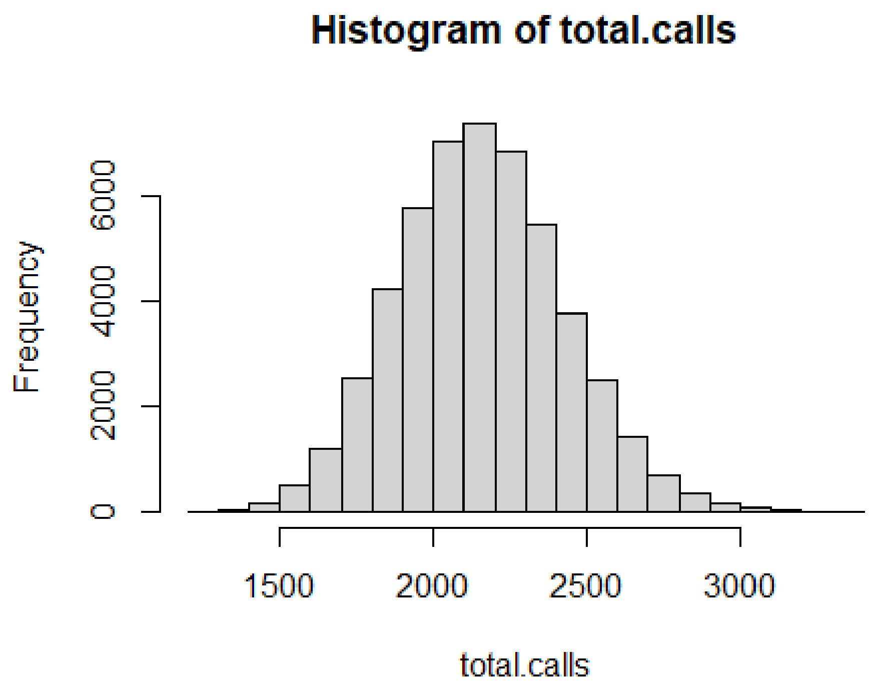
Consider the total number of calls that come in during February (28 total days).

a) Initialize an empty vector named `total.calls`. Then, use a Monte Carlo simulation with 50,000 trials to populate its elements with the total number of calls that arrive during all 28 days in February. Hint: define a vector `x<-1:250` (i.e., the possible values for the daily call volume), then your `sample` command can use `x` as the first argument, and you can type out the desired equation for the `prob=` argument to give each value the appropriate weight. Include the results of running `summary(total.calls)`. The commented values provide a sanity check. Since no random number seed has been set, your numbers for the 1st Qu., Median, Mean, and 3rd Qu. should be within 5 or so, while Min/Max should be within 100 or so.

```
total.calls <- c()
x <- 1:250
ntrials <- 50000

for (trial in 1:ntrials) {
  calls <- sample(x , size = 28, replace = TRUE , prob = x*exp(-x/40))
  total.calls[trial] <- sum(calls)
}


summary(total.calls)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1212    1970    2147    2155    2331    3321

#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#   1185    1971    2148    2156    2334    3353
```

b) Provide a histogram (`hist`) of the values in `total.calls` to see the variation in possible values.

```
hist(total.calls)
```



c) Using `total.calls`, the `mean` function, and logical conditions, estimate the probability that the total number of calls in February:

- Exceeds 2500 (about 10%)
- Is at most 1800 (about 9%)
- Is between 2000 and 2100 (inclusive); don't use `%in%` (about 14%)
- Is a multiple of 20 (inclusive). Check the Course Outline on Canvas for how `%in%` should be used (about 5%)

```
#Exceeds 2500

mean(total.calls > 2500)
## [1] 0.10218


#Is at most 1800

mean(total.calls <= 1800)
## [1] 0.08914

#Is between 2000 and 2100 (inclusive)

mean(total.calls %in% 2000:2100)
## [1] 0.14206
```

```
#Is a multiple of 20 (inclusive)

mean(total.calls %% 20 == 0)
## [1] 0.05028
```

Question 3: Inspections

The number of times a part is inspected before it is cleared for shipping is random but varies between 5 and 12 (inclusive). The probabilities of these integers aren't equal. Rather, a value of 12 is 1.5 times less likely than a value of 5, with the probabilities of the values varying linearly (see the Unit2A-Part3 activity purchase amounts model to review how to set up weights appropriately).

Consider a batch of 50 of these parts. The number of times each is inspected is independent of one another. Using a Monte Carlo simulation, estimate the probability that:

a) none of the parts are inspected exactly 12 times (about 0.5%)
b) 35 or fewer parts are inspected at least 8 times (about 97.8%)
c) the number of parts that are inspected exactly 7 times is larger than the number of parts that are inspected exactly 10 times (about 56.2%)

Develop your code so that it works for 10 trials, then obtain a final estimate of the probabilities with 100,000 trials. You can write a separate Monte Carlo simulation for each part, or you can write a single Monte Carlo simulation with three different counters, e.g., `counter.a`, `counter.b`, `counter.c`

```
counter.a <- 0
counter.b <- 0
counter.c <- 0
ntrials <- 100000

for (i in 1:ntrials) {
    bang <- sample(5:12 , size = 50, replace = TRUE, prob= seq(1.5,1,length =
8))
    if (!12 %in% bang) {counter.a <- counter.a + 1 }
    if (sum(bang >= 8) <= 35) {counter.b <- counter.b + 1 }
    if (length(which(bang == 7)) > length(which(bang == 10))) {counter.c <-
counter.c + 1}
}



counter.a/ntrials
## [1] 0.00494
```

```
counter.b/ntrials
## [1] 0.97756
counter.c/ntrials
## [1] 0.56447
```

Question 4: Restaurant Simulation

Imagine that the following model provides a fairly reasonable description of how people show up and order at a restaurant:

- Each night, between 9 and 22 parties dine at the restaurant (inclusive; all values equally likely)
- Each party has between 1 and 4 people with the following probabilities:

People	Probability
1	0.15
2	0.40
3	0.15
4	0.30

- Each diner in the party orders a dish that costs between 11 and 20 dollars (inclusive; all values equally likely)
- There's a 15% chance that a diner orders a 7 dollar drink

a) In theory, what is the absolute minimum and maximum amount of money that could be spent in a single night among all diners?

Response: The absolute minimum amount of money that could be spent in a single night among all diners would be $11. This would occur when 1 party of 1 orders the cheapest dish at $11 with no drinks. The maximum amount of money that could be spent in one night would be $2,376. This would occur when 22 parties come , each with 4 people, and they all get the $20 dish plus the $7 drink. This gives us the equation 22 * 4 * 27 to get the maximum possible amount spent as $2,376.

b) Many people might be tempted to assume that each value in the range from (a) is equally likely; after all, diner's behaviors are independent of one another and each diner's behavior has random elements to it. Explain why, in this situation, we wouldn't necessarily expect each value in the range from (a) to be equally likely.

Response: We should not expect each value to be equally likely, because it is more typical for certain groups to go to dinner over others. If I were to go into a restaurant right now around dinner, I would definitely expect to see more 2 person tables than 1. This is a pretty universal observation, and explains why someone would be unjust in assuming each probability in a to be the same.

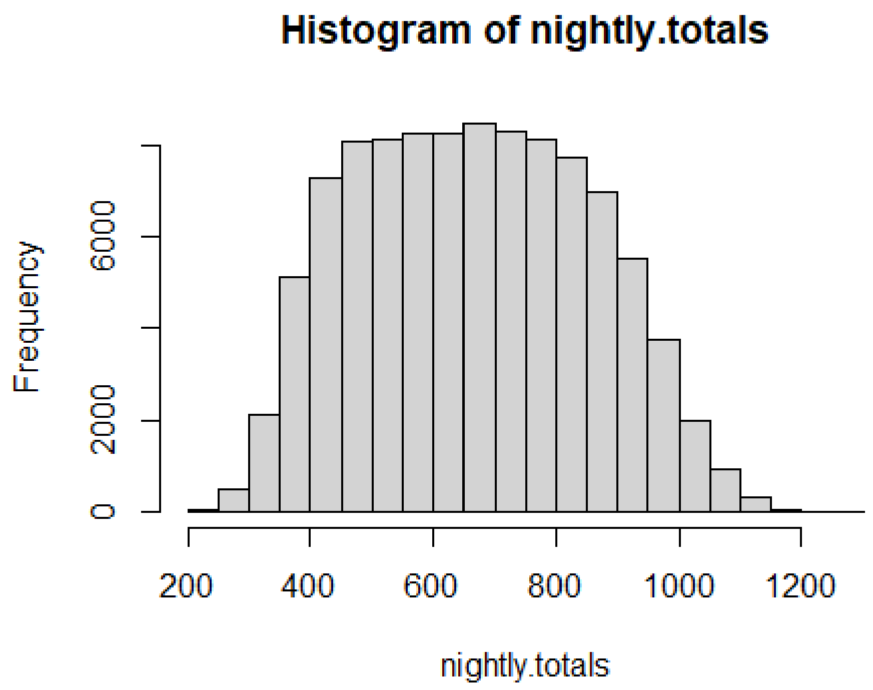
Run a Monte Carlo simulation that records the nightly total amounts spent at the restaurant over the course of 100,000 nights (i.e., create a vector name `nightly.totals` and store the totals found during each trial as elements of `nightly.totals`). For inspiration, review the first question in the Unit 2 Part 3 worksheet. Develop your code so that it works for 10 trials, then once you know that works, run your final simulation with 100,000 trials.

c) Create a histogram of the values in `nightly.totals` to see the distribution of possible values.

d) Use `mean` along with a logical condition on the values in `nightly.totals` to estimate the probability that a nightly total is at most 400 (about 7.6%).

e) Use `mean` along with a logical condition on the values in `nightly.totals` to estimate the probability that a nightly total is between 580 and 870 dollars inclusive (about 47.4%)

```
ntrials <- 100000
nightly.totals <- c()
drinks <- c(0,7)


for (i in 1:ntrials) {
  parties <- sample(9:22, size = 1)
  people <- sample(1:4, size = parties, replace = TRUE, prob =
c(0.15,0.4,0.15,0.30))
  prices <- sample (11:20, size = sum(people), replace = TRUE)
  nightly.totals[i] <- sum(prices)
  drinks <- sample(c(0,7), size = sum(people), replace = TRUE, prob =
c(.85,.15))
  drink_totals <- sum(drinks)
  nightly.totals[i] <- nightly.totals[i] + drink_totals
}

hist(nightly.totals)
```



```
mean(nightly.totals <= 400)
## [1] 0.0776
mean(nightly.totals >= 580 & nightly.totals <= 870)
## [1] 0.47378
```

Question 5: Stock price evolution

A stock's price is currently at 85 dollars. You have a reasonably realistic model for daily price changes. Each day, the stock's price:

- goes down by 3 (6% chance)
- goes down by 2 (11% chance)
- goes down by 1 (16% chance)
- stays the same (40% chance)
- goes up by 1 (14% chance)
- goes up by 2 (9% chance)
- goes up by 3 (4% chance)

You have a substantial investment in options related to the stock, and if the stock *always* closes between 75 and 95 dollars (inclusive) over the next 100 trading days, you will make a fortune! In this problem, you will write a Monte Carlo simulation to study the evolution of the stock's price. Use the following strategy (or develop your own):

- In each trial, generate 100 stock movements using a `sample` command
- Use the `cumsum()` function on the output of `sample` and add 85 (the starting price) to each element to find the daily closing prices
- Record the highest closing price seen during the trial in an element of a vector named `highest`
- Record the lowest closing price seen during the trial in an element of a vector named `lowest`
- Develop your code so that it works for 10 trials, then obtain a final estimate of the probability with 10,000 trials.

a) Run `hist(lowest); abline(v=75)` and `hist(highest); abline(v=95)` to see just how low and how high the closing price typically gets during the course of 100 trading days (`abline` adds a vertical line to provide a reference for the critical closing prices).

b) Use a `mean()` function on some logical conditions using `lowest` and `highest` to estimate the probability that you make a fortune. Sanity check: about 14%.

c) You have the ability to exchange your options for ones that pay off if the stock *ever* closes above 105 over those 100 days. Does this look like it has a higher chance of paying off than your current set of options? Explain, quoting an estimate for the probability that this new option pays off. No need to re-run the simulation; use the values in `highest`.

Response:

Question 6: More candy

A pack of candy contains 30 pieces. Each candy is one of four colors, but each color isn't equally likely. Each candy has a

- 10% chance of being turquoise
- 34% chance of being blue
- 11% chance of being green
- 45% chance of being pink

Although pink is the most likely color for a piece, there is no guarantee that it will actually be the color that appears most often in the pack.

a) Using a Monte Carlo simulation, estimate the probability that pink *won't* be the most frequently occurring color. Note: a tie for most frequent color qualifies as being the most frequent color. Develop your code so that it works for 10 trials, then obtain a final estimate of the probability with 20,000 trials. Sanity check: about 22%.

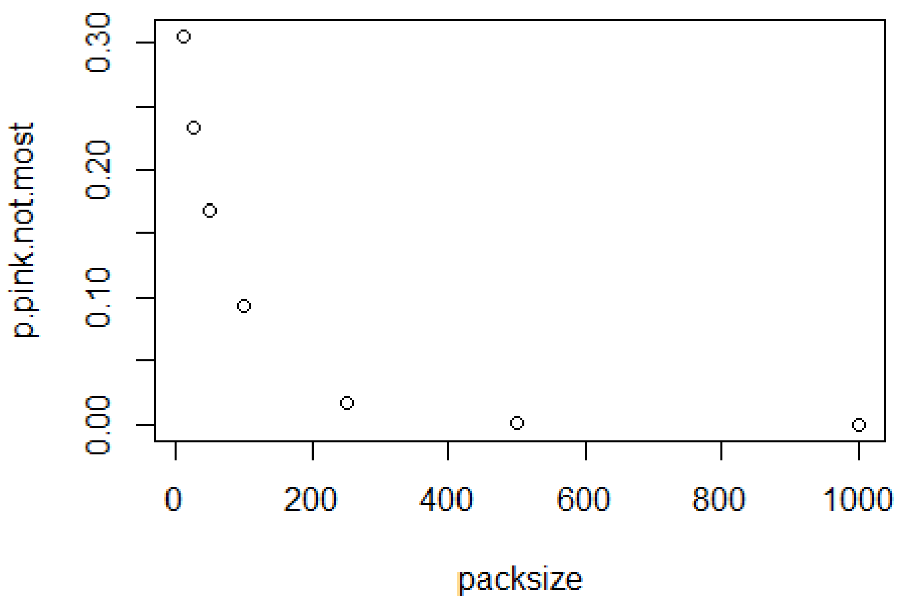
```
ntrials <- 20000
results <- c()
```

```r
for (i in 1:ntrials) {
   colors <- sample(c("turquoise", "blue", "green", "pink"), 30,
replace=TRUE, prob=c(0.1, 0.34, 0.11, 0.45))
  color.count <- table(colors)
  max.count <- max(color.count)
  max.colors <- names(color.count[color.count == max.count])
  results[i] <-  ifelse("pink" %in% max.colors, 0 , 1 )
}
mean(results)
## [1] 0.21695
```

b) The ~22% estimate is specific to packs with 30 candies. You might wonder how this probability changes with the number of candies in the bag. Think about it first: do you expect the probability that pink won't be the most frequently occurring color to stay the same, increase, or decrease as pack-size increases? (You don't need to include your answer in the writeup.) Now, write a Monte Carlo simulation to see if you're right! Here's one strategy (you are free to choose a different approach or choose different variable names if you wish):

- Initialize an empty vector named `p.pink.not.most`
- Define a vector named `packsize` that contains the elements 10, 25, 50, 100, 250, 500, 1000
- Create a `for` loop with a looping variable named `packsizeposition` and with the looping vector `1:length(packsize)`
- Inside this `for` loop, you can copy/paste your code from part (a). However:
  - change the number of trials to be 1000
  - change the `size=` argument of `sample` appropriately
  - store the estimated probability for that pack size into `p.pink.not.most` appropriately
- Once the simulation has concluded, include a `plot(p.pink.not.most ~ packsize)` to show the relationship!

```r
p.pink.not.most <- c()
ntrials <- 1000

packsize <- c(10, 25, 50, 100, 250, 500, 1000)


for (packsizeposition in 1:length(packsize)) {
  counter <- 0
  for (i in 1:ntrials) {
      colors <- sample(c("turquoise", "blue", "green", "pink"),
packsize[packsizeposition], replace=TRUE, prob=c(0.1, 0.34, 0.11, 0.45))
  color.count <- table(colors)
  max.count <- max(color.count)
  max.colors <- names(color.count[color.count == max.count])
  if(!"pink" %in% max.colors ) {counter <- counter + 1 }
  }
  p.pink.not.most[packsizeposition] <- counter/ntrials

}

plot(p.pink.not.most ~ packsize)
```



Question 7 (Long-term Bonus): Inning in baseball (high difficulty; you can turn this in anytime before the midterm for full points)

BAS 471 is immensely useful if you want to get into sports analytics since they use probability calculations, modeling, and Monte Carlo simulation all the time. I have seen multiple sports analytics interview questions, and I can say that mastering Units 1-3 in BAS 471 gives you a big edge when trying to get internships and jobs with sports teams.

This question is aimed at fairly advanced programmers who have a decent knowledge of baseball, or for those people who are very interested in getting into sports analytics. You will write a simulation of an inning in a much-simplified game of baseball. While the simulation will be missing quite a few key bits of realism, it does provide a jumping off point. Note: even with this simplified game, my own simulation was about 60 lines of code!

Imagine:

- Each pitch has an independent 60% chance of being a strike (40% being a ball) for all batters. No batters ever get hit by a pitch and are automatically walked. This number is close the league average
- Every batter has a 65% chance of swinging at a pitch that would be a strike (Z-Swing%); league average
- Every batter has a 30% chance of swinging at a pitch that would be a ball (O-Swing% / Chase Rate)
- When the batter swings at a pitch that would be a strike:
    - misses with a 20% chance (counts as a strike)
    - is "out" with a 53% chance (foul/fly ball caught, or thrown out at first base, etc.)
    - hits a single with a 15% chance
    - hits a double with a 6% chance
    - hits a triple with a 2% chance
    - hits a home run with a 4% chance
- When the batter swings at a pitch that would be a ball, he:
    - misses with a 40% chance (counts as a strike)
    - is "out" with a 48% chance (foul/fly ball caught, or thrown out at first base, etc.)
    - hits a single with a 7% chance
    - hits a double with a 3% chance
    - hits a triple with a 1% chance
    - hits a home run with a 1% chance
- When the batter hits the ball and ends up on a base
    - All other players on bases move forward the same number of bases
    - Example, if a batter hits a double and a player is on 2nd base, the player advances two bases (arriving home and scoring a point).
- No player ever tries to steal a base. Obviously this is not very realistic, but it makes the simulation simpler.
- Players already on base are "safe". In other words, a player already on base is never tagged out or thrown out via a double/triple play. Obviously this is not very realistic, but it makes the simulation simpler.

Using a Monte Carlo simulation, estimate:

a) The probability that no runs are scored in an inning. Sanity check: close to 79%

b) The average number of runs scored during an inning. Sanity check: close to 0.37.

You can estimate both at once if, in each trial, you store the number of runs that have been scored. If you follow baseball and don't find these numbers realistic, that indicates that this simulation is overly simplistic and that perhaps some/many of the probabilities (ball vs. strike, outcome of a swing, etc.) are not reflective of reality. In theory, you could play around with these numbers to get a closer match to reality.