Problem Set 2 Solutions

Lars Lien Ankile

February 18, 2020

Problem 1.

a)

```
#use r as a calculator
total <- 100
at_least_3_years <- 100 - 20

at_least_3_years / total
```

```
## [1] 0.8
```

The probability of someone in the MBA program having three or more years of work experience is the complement of the people having 2 years of experience divided by the total. This comes out to be $\frac{100-20}{100} = 0.8$, or 80%.

b)

```
#use r as a calculator
four_years <- 15

four_years / at_least_3_years
```

```
## [1] 0.1875
```

The probability of someone having four years of experience given that the population has at least three years is the total number of people having exactly four years of experience devided by the number of people having three or more years of experience, which is $\frac{15}{80} = 0.1875$ or 18.75%.

c) The probability of all three having five or more years of experience will be the probability of drawing one out of the population, then one more, and lastly one more. This is $\frac{35}{100}\frac{34}{99}\frac{33}{98} \approx 0.0405$, or roughly 4.1%. One important assumption here is the mix of people from year to year stays roughly constant for this calculation to be valid. Without knowing too much about who goes to HBS, I think that the mix of people stays relatively constant, and that the above calculation is within some reasonable margin of error for most years.

```
#use r as a calculator
five_or_more <- 35
```

```
((five_or_more / total)
  * ((five_or_more - 1) / (total - 1))
  * ((five_or_more - 2) / (total - 2)))
```

```
## [1] 0.04047619
```

d) Whether one could use the probability from a) for MIT Sloan MBA depends on whether or not they have more or less the same mix of people with different levels of experience. From what I know about Harvard and MIT in general, it seems likely for them to admit poeple with similar backgrounds. Therefore, I think it's plausible that the result can be applicable, but one should really do some research into the MIT population before one draws too many conclusions.

Problem 2.

a) i.

If x is the number of positive tests, the probability of two or more people testing positive is the complement of $p(x = 0 \cup x = 1)$. This comes out to be

$$\begin{aligned} p(x > 1) &= 1 - p(x = 0 \cup x = 1) \\ &= 1 - (0.2^5 + 5 \cdot 0.8 \cdot 0.2^4) \\ &= 1 - 0.0016 \\ &= 0.99328 \end{aligned}$$

or 99.328% probability.

```
#use r as a calculator

# Probability of no-one and one person testing positive
zero_positive <- 0.2^5
one_positive <- 5 * (0.8 * 0.2^4)

# Probability of at least two people testing postive will be the complement
# of the probability of none and exactly one person testing positive
1 - (zero_positive + one_positive)
```

```
## [1] 0.99328
```

ii.

```
#set the parameters
prob_positive <- 0.8
num_tests <- 5
num_simulations <- 100000

#create empty vectors to store results
outcomes <- vector("numeric", length = num_simulations)
```

```r
#set the seed
set.seed(2019)

#simulate the tests
for(k in 1:num_simulations){

  # Outcome 0 is negative test, otcome 1 is positive
  outcome = sample(c(0, 1), size = num_tests,
                       prob = c(1 - prob_positive, prob_positive), replace = TRUE)

  outcomes[k] = sum(outcome)
}

#view results
#outcomes
addmargins(table(outcomes))
```

```
## outcomes
##      0      1      2      3      4      5    Sum
##     33    626   5202  20396  40928  32815 100000
```

```r
two_or_more = (outcomes > 1)
table(two_or_more)
```

```
## two_or_more
## FALSE  TRUE
##   659 99341
```

```r
#calculation
length(two_or_more[two_or_more == TRUE]) / length(two_or_more)
```

```
## [1] 0.99341
```

As the simulation above shows, the algebraic number of 0.99328 seems reasonable when compared to the empiracal number of 0.99341, only a 0.00013 difference.

b) i. Let $P(A)$ is the probability that a driver has BAC over the legal limit and let $P(B)$ be the probability that someone tests positive for the breath test. Then, the probability of someone who tests positive actually has over the legal limit will be $P(A|B)$. According to Bayes' theorem, this comes out to be

$$\begin{aligned} P(A|B) &= \frac{P(A)P(B|A)}{P(B)} \\ &= \frac{0.05 \cdot 0.8}{0.05 \cdot 0.8 + (1 - 0.05) \cdot 0.8} \\ &= 0.173913. \end{aligned}$$

```r
#use r as a calculator
(0.05 * 0.8) / (0.05 * 0.8 + (1-0.05) * (1-0.8))
```

```
## [1] 0.173913
```

The above equation comes out to be roughly 0.174, or 17.4%, when we plug in the numbers. This is actually pretty low for a breathalyzer, if you ask me.

ii. To figure this one out we would need to reorganize Bayes' rule a little. We now want to find $P(B|A)$, so I end up with the following equation

$$
\begin{aligned}
& & P(A|B) &= \frac{P(A)P(B|A)}{P(B)} \\
&\iff & P(B|A) &= \frac{P(B)P(A|B)}{P(A)} \\
& & &= \frac{1 - P(A)}{\frac{P(A)}{P(A|B)} + 1 - 2P(A)} \\
& & &= \frac{1 - 0.05}{\frac{0.05}{0.8} + 1 - 2 \cdot 0.05}
\end{aligned}
$$

```r
#use r as a calculator
(1-0.05)/((0.05/0.8) + 1 - 2*0.05)
```

```
## [1] 0.987013
```

From the above we see that for the breathalyzer to actually be correct if it gives a positive result in 80% of the cases it would need to give positive results in 98.7% of cases where people have illegal BAC levels, and give negative results in 98.7% of cases where people have legal BAC levels.

iii. The reason the probability is so low is mainly because the probability of people having an illegal BAC level is so small compared to the amount of people who don't. So, if you stop someone at random the probability that they are correctly identified as over the limit is $0.05 \cdot 0.8 = 0.04$, while the probability of it being a case of false positive is $0.95 \cdot 0.20 = 0.19$, which is a much larger probability.

Problem 3.

a) Let $P(A)$ be the probability of knowing the answer to a question, $P(B)$ be the probability of answering a question correctly, and $\frac{1}{m}$ be the probability answering correctly when just guessing, then the probability of knowing the answer to a question given that it is answered correctly is

$$\begin{aligned} P(A|B) &= \frac{P(A)P(B|A)}{P(B)} \\ &= \frac{p \cdot 1}{p \cdot 1 + \frac{1}{m}(1-p)} \\ &= \frac{p}{p + \frac{1}{m}(1-p)} \end{aligned}$$

b) Let $P(A)$ be the probability that any given person has an IQ below 132, $P(B)$ be the probability that someone scores 132 or better on the test. We also know that $P(A) = 0.98$, $P(B|A) = 0.001$, $P(B|A^C) = 0.95$. The probability of someone scoring 132 or better on the test, but actually having less than an IQ of 132 is given by

$$\begin{aligned} P(A|B) &= \frac{P(A)P(B|A)}{P(B)} \\ &= \frac{P(A)P(B|A)}{P(A)P(B|A) + P(A^C)P(B|A^C)} \\ &= \frac{0.98 \cdot 0.001}{0.98 \cdot 0.001 + (1-0.98) \cdot 0.95} \\ &= 0.04904905. \end{aligned}$$

If we plug these numbers into R we get the following.

```
(0.98 * 0.001) / (0.98 * 0.001 + (1-0.98) * 0.95)
```

```
## [1] 0.04904905
```

Assuming that a representative sample of the population actually takes the test, and we find an individual at random who has a score of 132 or above, the probability of this person actually having an IQ of lower than 132 is roughly 4.9%.

Problem 4.

Since cards are drawn without replacement, we know that there must exist some strict ordering over the cards. That is because no card can be drawn twice. Since there are 3 cards to be drawn, there is $3! = 6$ different ways of ordering these cards in ascending order ($\{X < Y < Z, X < Z < Y, Y < X < Z, Y < Z < X, Z < X < Y, X < Y < X\}$), and one of these must hold (because one can always order unique integers). Since the probability of drawing any card is uniform, the probability

of ending up in any of these configurations are the same, so the probability of $X < Y < Z$ is $\frac{1}{6} \approx 16.67\%$.

Problem 5.

```
#define parameters
number.rolls = 2
replicates = 100

#create empty vector to store results
successes.5 = vector("numeric", replicates)
successes.1 = vector("numeric", replicates)

#set seed
set.seed(2020)

#simulate the draws
for(k in 1:replicates){

  rolls = sample(1:6, number.rolls, replace = TRUE)

  if (sum(rolls) == 5) {

    successes.5[k] = 1

    if (rolls[1] == 1 | rolls[2] == 1) {
      successes.1[k] = 1
    }
  }
}
```

a) Unfortunately, true randomness is hard to come by in computers, so they instead use pseudo-random number generators. These produce near-uniform distributions of numbers, based on a starting value. If this starting value is known, then all subsquent values are predictable. The reason for setting the seed here is so my results will be the same as anyone else's results, even though there's "randomness" involved.

b) The actual rolls are generated by the sample-function provided by R. The sample function first accepts a list of elements to sample from. Second, the code specifies how many numbers one wants from that list. The last argument makes sure that the two dice rolls are independant of each other, i.e. one can roll a 6 even though you just rolled 6 on another dice. That function outputs a vector with 2 elements in it, each a number between 1 and 6, inclusive, with equal pribability of being any of them, just like a normal dice.

c) i. The condition for a 1 to be recorded in the successes.5 vector is that the sum of all elements in the vector produced by sample is equal to 5. This is equivalent to throwing two dice and having them sum to 5.

ii. The second, nested if-statement checks whether any of the dice thrown happens to have the value 1. This check happens only if we already found that the sum of the two dice is 5.

d)

```
#view the results
addmargins(table(successes.5))
```

```
## successes.5
##   0   1 Sum
##  84  16 100
```

```
addmargins(table(successes.1))
```

```
## successes.1
##   0   1 Sum
##  91   9 100
```

```
#calculate the estimated probability
length(successes.1[successes.1 == 1]) / length(successes.5[successes.5 == 1])
```

```
## [1] 0.5625
```

Based on the above simulations and calculations, we find that the probability of one of the dice is 1, given that their sum is 5, is 0.5625, or roughly 56.3%.

e) If your dice add up to 5, then there's exactly four different combinations of dice one could have: $\{(1,4),(2,3),(3,2),(4,1)\}$. Exactly half of these possibilities contain a one. Therefore, the actual probability of having at least one dice being a 1, given that their sum is 5, is 0.5, or 50%.

f) There is a different between the emprical results and the theoretical results by 6.25 percetage points. Most likely, this is caused by the fact that the number of replications of the dice throws are relatively low at 100, and therefore could be off just because of randomness, and too few samples. Even though our sample are drawn from a uniform distribution, it is very possible that with just 100 runs, we by pure randomness draw a little more samples from one end than another. To check this we can either increase the number of replicates or try a different seed and see if that changes things. By increasing the number of replicates I observe a convergence towards 0.5. By changing the seeds, the value jumps around 0.5 a lot.

Problem 6.

a) There's three cases to consider: first card being from a black suit (p=0.5), first card being a diamond (p=0.25), and the first card being a heart (p=0.25), which all have different probabilities for the second card being a heart given that it is of a red suit. In the first case the pobability for it being a heart is $\frac{13}{26} = 0.5$, in the second case it is $\frac{13}{25}$, and in the last it is $\frac{12}{25}$. The full expression and solution is

$$0.5 \cdot \frac{13}{26} + 0.25 \cdot \frac{13}{25} + 0.25 \cdot \frac{12}{25} = 0.5$$

```
0.5*0.5 + 0.25*(13/25 + 12/25)
```

```
## [1] 0.5
```

b) There's 4 ways of drawing exactly 1 heart: (1) drawing first one heart and then 3 non-hearts, HNNN, (2) drawing a non-heart, a heart, and then 2 non-hearts, NHNN, (3) NNHN, and (4) NNNH. All of these have the same probability of happening, so we can calculate the probability of one and multiply that by four:

$$\begin{aligned} P(\text{Exactly 1 heart}) &= 4 \cdot \frac{13}{52}\frac{39}{51}\frac{38}{50}\frac{37}{49} \\ &= 0.4388475 \end{aligned}$$

```
4 * (13/52 * 39/51 * 38/50 * 37/49)
```

```
## [1] 0.4388475
```

The probability of drawing exactly one heart is 0.4388, or 43.9%.

c)

The simulation is as follows.

```
#set parameters
num_suits <- 4
num_cards <- 52
cards_in_suit <- num_cards / num_suits
replicates <- 10000

# Create an empty vector to hold my deck of cards
cards = vector('numeric', length = num_cards)

# A double for-loop for filling up my deck of
# cards with 13 cards in each suit
for (i in 1:num_suits) {
  for (j in 1:cards_in_suit) {
    cards[(i-1) * cards_in_suit + j] <- i
  }
}


#create empty vectors to store results
successes <- vector('numeric', replicates)

#set seed
set.seed(1969)
```

```r
#simulate the draws
# This loops is one full experiment, and will be
# performed replicates amount of times
for (k in 1:replicates) {

  # Take out 4 cards at random out of the deck of
  # cards constructed above, with no replacement
  four_cards <- sample(cards, 4, replace = FALSE)

  # The unique function only keeps one of each number,
  # so if the length is 4 after calling unique
  # then there's must've been one card from each
  # suit in the random sample
  if (length(unique(four_cards)) == num_suits) {
    # In case we found a card from each suit we add a
    # one to the vector at that place
    successes[k] <- 1
  }

}

#view results
# Count how many times we had success in all replications
num_successes <- sum(successes)

# Print a table for a visual of the mix of successes to failures
addmargins(table(successes))
```

```
## successes
##     0     1   Sum
##  8928  1072 10000
```

```r
# Calculate the percentage of draws that was successful
num_successes / replicates
```

```
## [1] 0.1072
```

From the results of the simulation we see that the probability of drawing exactly one of each suit is 0.1072, or 10.7%.

Problem 7.

Let $P(A)$ be the probability that a set of twins are identical and $P(B)$ be the probability that a set of twins are both girls. Probabilities given in the problem are $P(A) = 0.3$, $P(B|A) = 0.5$, $P(B) = P(A)P(B|A) + P(A^C)P(B|A^C) = 0.3 \cdot 0.5 + 0.7 \cdot 0.25 = 0.325$. The probability that a set of twins are identical, given that they're both girls is

$$\begin{aligned} P(A|B) &= \frac{P(A)P(B|A)}{P(B)} \\ &= \frac{0.3 \cdot 0.5}{0.325} \\ &= 0.4615385 \end{aligned}$$

```
#use r as a calculator
(0.3 * 0.5) / (0.3*0.5 + 0.7*0.25)
```

```
## [1] 0.4615385
```

From the above explaination and calculations in r, we see that the probability of a pair of twin girls being identical is roughly 0.462, or 46.2%.

Problem 8.

```
# Here I've defined a function of the simulation for simplicity of
# solving the subsequent tasks.
# I've also put in default arguments I can later override in b) and c).
count_positives <- function(p_true_positive = 0.95, p_true_negative = 0.97) {

  # define parameters
  replicates <- 100000
  # The proportion of people having the predisposition
  p_sick <- 0.03

  # create empty vectors to store results
  # All people, 0 if healthy, 1 if have predisposition
  population <- vector('numeric', replicates)
  # Hold whether someone tests positive, 0 negative, 1 positive
  results <- vector('numeric', replicates)
  # Count the people who tests positive and actually have the predisposition
  true_positives <- vector('numeric', replicates)

  # set seed
  set.seed(666)

  # assign genetic status
  # Fill the population vector with 3% sick people
```

```
  for (i in 1:(p_sick * replicates)) {
    population[i] <- 1
  }

  # Shuffle the population for more realistic feel
  population <- sample(population)

  #assign test result
  for (k in 1:replicates) {
    # 0 means the person does not have the disposition
    if (population[k] == 0) {
      # If the person doesn't have the predisposition, get
      # whether they test positive or not
      test_result <- sample(0:1,
                            prob = c(p_true_negative, 1-p_true_negative),
                            size = 1)
    } else {
      # If the person have the predisposition, get whether
      # they test positive or not
      test_result <- sample(0:1,
                            prob = c(1-p_true_positive, p_true_positive),
                            size = 1)
      # Also add them to the list of true positives when the
      # people with the predisposition tests positive
      true_positives[k] <- test_result
    }
    # Add the test result for person k to the list
    results[k] <- test_result
  }

  #view results
  addmargins(table(results))

  # The first return element is the actual number of
  # people who have the predisposition.
  # The second return element is the number of people who tested positive.
  # The third return element is the number of people who
  # have the disposition and tested positive.
  return (c(sum(population), sum(results), sum(true_positives)))
}
```

```
results1 <- count_positives()
results1
```

```
## [1] 3000 5696 2852
```

a) Based on the above simulations we see that 5696 people are estimated to be testing positive for the predisposition, out of a population of 100,000 people where we have assumed 3%, or 3,000,

actually has the disposition. Here I've assumed that exactly 3% of the population has the predisposition. I.e. I've put in exactly 3000 people with the predisposition and then shuffled. I could've also sampled randomly with a 3% probability, but I thought that could introduce unnecessary noise.s

b)

```
results1[3] / results1[2]
```

```
## [1] 0.5007022
```

Since we know that 3,000 people actually has the disposition and that 5696 people tests positive, we can estimate that the probability that someone who tests positive actually has the predisposition is $\frac{3000}{5696} = 0.5267$, or 52.3% of the positives.

c) Intuitively, it makes sense that it must be alterative B that increases the chance of someone testing positive actually being a true positive. This is because, increasing the accuracy of the test for people who do not have the predisposition means that less people will be testing false positive for the test, which means that a higher proportion of the people testing positive will actually have the predisposition. This is also confirmed by the below simulations.

```
result2 <- count_positives(p_true_positive = 0.98)
result2[3] / result2[2]
```

```
## [1] 0.5079585
```

From the above we see that the percentage is a little improved, but since so few people actually has the predisposition the 3 percentage points increase in accuracy won't make much of a difference.

```
result3 <- count_positives(p_true_negative = 0.99)
result3[3] / result3[2]
```

```
## [1] 0.7491463
```

Here we see that the chance of actually having the predisposition is much larger. This is because the number of people who doesn't have the predisposition is so large that i small increase in the accuracy make a big difference.