Problem Set 4: Random Variables, Probability, and Screening Properties

Your name and student ID

July 22, 2022

```
BEGIN ASSIGNMENT
requirements: requirements.R
generate: true
```

Run this chunk of code to load the autograder package!

Instructions

- Solutions will be released on Friday, July 22nd.
- This semester, problem sets are for practice only and will not be turned in for marks.

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on Datahub. Alternatively, you may wish to view the code in the condensed PDFs posted on the course website. Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting! We recommend knitting your file each time after you write a few sentences/add a new code chunk, so you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration!

Part 1: Simulating Birth Defect Data and Sampling from an Infinitely Large Population

The Center for Disease Control and Prevention (CDC) estimates that 1 in every 33 infants is born with a birth defect in the United States each year.

1. [3 points] Define a random variable for "birth defect". Write down the probability model for the random variable. Round each percentage to two decimal places (e.g., 0.43224 would be rounded to 43.22%). Is the sample space discrete or continuous?

```
BEGIN QUESTION
name: p1
manual: true
```

You might want to use the table template below to write out your probability model. *Knit now* to see how this table is rendered in your PDF.

Tables	Are	Cool
yadi	type stuff	X
yadi	more stuff	Y
yada	etc	Z

[1 point] Let BD represent the event "birth defect".

[1 point] Then the probability model is:

Birth defect	BD	BD'
Probability	1/33 = 3.03%	32/33 = 97.0%

[1 point] The sample space is discrete.

2. [2 points] Simulate data that equals 0 if there is no birth defect and equals 1 if there is a birth defect. Simulate this data for 200 births at a local hospital. Be sure to use the risk of birth defect from part a). Assign your simulated output the name `sim_01`. Print your simulated births to the screen.

Before you run your simulation, we will "set the seed" to 100. This means that everyone's simulation will yield the exact same dataset.

```
# execute this line before you write your simulation code.
# only execute the set.seed() function one time.
set.seed(100)
```

```
BEGIN QUESTION
name: p2
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
sim_01 <- NULL # YOUR CODE HERE
sim_01
" # END PROMPT
# BEGIN SOLUTION NO PROMPT
sim_01 <- rbinom(200, 1, prob = 1/33)
sim_01
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
```

```
# END SOLUTION
```

```
## Test ##
test_that("p2a", {
  expect_true("integer" == class(sim_01))
  print("Checking: sim_01 should be an integer vector")
})
```

```
## [1] "Checking: sim_01 should be an integer vector"
## Test passed
```

```
## Test ##
test_that("p2b", {
  expect_true(length(sim_01) == 200)
  print("Checking: number of elements in sim_01")
})
```

```
## [1] "Checking: number of elements in sim_01"
## Test passed
```

Notice that `sim_01` is not a dataframe - it is a vector of numbers. The following code stores `sim_01` as a dataframe and changes its variable name. Run this code and view `sim_01` in the Viewer pane.

```
library(dplyr)
sim_01 <- as.data.frame(sim_01) # watch what happens to sim_01 in your environment
names(sim_01) # prints the variable names in the sim_01 data frame
```

```
## [1] "sim_01"
```

```
sim_01 <- sim_01 %>% rename(birth_defect = sim_01)
```

3. [2 points] Write code to determine the number of birth defects that occurred in your simulation, and the corresponding proportion with birth defects. Assign your output to the object `output_01`. Print `output_01` to the screen. Hint: Use `dplyr` functions to do this.

```
BEGIN QUESTION
name: p3
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
output_01 <- NULL # YOUR CODE HERE
output_01
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
output_01 <- sim_01 %>% summarize(number = sum(birth_defect),
                                  prop = mean(birth_defect))
output_01
```

```
##   number  prop
## 1      5 0.025
```

```
# END SOLUTION
```

```
## Test ##
test_that("p3a", {
  expect_true(is.data.frame(output_01))
  print("Checking: output_01 is a dataframe")
})
```

```
## [1] "Checking: output_01 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p3b", {
  expect_true(ncol(output_01) == 2)
  print("Checking: output_01 has two columns")
})
```

```
## [1] "Checking: output_01 has two columns"
## Test passed
```

```
## Test ##
test_that("p3c", {
  expect_true(nrow(output_01) == 1)
  print("Checking: output_01 has 1 row")
})
```

```
## [1] "Checking: output_01 has 1 row"
## Test passed
```

4. [2 points] Re-run your simulation four more times and assign the output to a unique name each time. Print the number and proportion to the screen after each run. (Basically, "recycle" the code above four times).

```
BEGIN QUESTION
name: p4
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
sim_02 <- 'YOUR ANSWER HERE'
output_02 <- 'YOUR ANSWER HERE'
output_02
sim_03 <- 'YOUR ANSWER HERE'
output_03 <- 'YOUR ANSWER HERE'
output_03
sim_04 <- 'YOUR ANSWER HERE'
output_04 <- 'YOUR ANSWER HERE'
output_04
sim_05 <- 'YOUR ANSWER HERE'
output_05 <- 'YOUR ANSWER HERE'
output_05
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
#second run
sim_02 <- rbinom(200, 1, prob = 1/33)
sim_02 <- as.data.frame(sim_02) %>% rename(birth_defect = sim_02)
output_02 <- sim_02 %>% summarize(number = sum(birth_defect),
                                  prop = mean(birth_defect))
output_02
```

```
##   number  prop
## 1      5 0.025
```

```
#third run
sim_03 <- rbinom(200, 1, prob = 1/33)
sim_03 <- as.data.frame(sim_03) %>% rename(birth_defect = sim_03)
output_03 <- sim_03 %>% summarize(number = sum(birth_defect),
                                  prop = mean(birth_defect))
output_03
```

```
##   number prop
## 1     10 0.05
```

```
#fourth run
sim_04 <- rbinom(200, 1, prob = 1/33)
sim_04 <- as.data.frame(sim_04) %>% rename(birth_defect = sim_04)
output_04 <- sim_04 %>% summarize(number = sum(birth_defect),
                                  prop = mean(birth_defect))
output_04
```

```
##   number prop
## 1      6 0.03
```

```
#fifth run
sim_05 <- rbinom(200, 1, prob = 1/33)
sim_05 <- as.data.frame(sim_05) %>% rename(birth_defect = sim_05)
output_05 <- sim_05 %>% summarize(number = sum(birth_defect),
                                  prop = mean(birth_defect))
output_05
```

```
##   number prop
## 1      8 0.04
```

```
# alternative solution (for loop or...)
# students with programming experience may choose to use a for loop or other
# iteration function to solve this question.
# END SOLUTION
```

```
## Test ##
test_that("p4a", {
  expect_true(is.data.frame(output_02))
  print("Checking: output_02 is a dataframe")
})
```

```
## [1] "Checking: output_02 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p4b", {
  expect_true(ncol(output_02) == 2)
  print("Checking: output_02 has two columns")
})
```

```
## [1] "Checking: output_02 has two columns"
## Test passed
```

```
## Test ##
test_that("p4c", {
  expect_true(nrow(output_02) == 1)
  print("Checking: output_02 has 1 row")
})
```

```
## [1] "Checking: output_02 has 1 row"
## Test passed
```

```
## Test ##
test_that("p4d", {
  expect_true(is.data.frame(output_03))
  print("Checking: output_03 is a dataframe")
})
```

```
## [1] "Checking: output_03 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p4e", {
  expect_true(ncol(output_03) == 2)
  print("Checking: output_03 has two columns")
})
```

```
## [1] "Checking: output_03 has two columns"
## Test passed
```

```
## Test ##
test_that("p4f", {
  expect_true(nrow(output_03) == 1)
  print("Checking: output_03 has 1 row")
})
```

```
## [1] "Checking: output_03 has 1 row"
## Test passed
```

```
## Test ##
test_that("p4g", {
  expect_true(is.data.frame(output_04))
  print("Checking: output_04 is a dataframe")
})
```

```
## [1] "Checking: output_04 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p4h", {
  expect_true(ncol(output_04) == 2)
  print("Checking: output_04 has two columns")
})
```

```
## [1] "Checking: output_04 has two columns"
## Test passed
```

```
## Test ##
test_that("p4i", {
  expect_true(nrow(output_04) == 1)
  print("Checking: output_04 has 1 row")
})
```

```
## [1] "Checking: output_04 has 1 row"
## Test passed
```

```
## Test ##
test_that("p4j", {
  expect_true(is.data.frame(output_05))
  print("Checking: output_05 is a dataframe")
})
```

```
## [1] "Checking: output_05 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p4k", {
  expect_true(ncol(output_05) == 2)
  print("Checking: output_05 has two columns")
})
```

```
## [1] "Checking: output_05 has two columns"
## Test passed
```

```
## Test ##
test_that("p4l", {
  expect_true(nrow(output_05) == 1)
  print("Checking: output_05 has 1 row")
})
```

```
## [1] "Checking: output_05 has 1 row"
## Test passed
```

5. [1 point] Assign the vector `p5` to the simulated proportions from each of your five (total) simulations in *increasing* order.

```
BEGIN QUESTION
name: p5
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p5 <- c('YOUR ANSWER HERE')
p5
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p5 <- c(0.025, 0.025, 0.03, 0.04, 0.05)
# END SOLUTION
```

```
## Test ##
test_that("p5a", {
  expect_true("numeric" %in% class(p5))
  print("Checking: p5 is a numeric vector")
})
```

```
## [1] "Checking: p5 is a numeric vector"
## Test passed
```

```
## Test ##
test_that("p5b", {
  expect_true(length(p5) == 5)
  print("Checking: p5 has 5 elements in it")
})
```

```
## [1] "Checking: p5 has 5 elements in it"
## Test passed
```

```
## Test ##
test_that("p5c", {
  expect_true( p5[1] <= p5[2] & p5[2] <= p5[3] & p5[3] <= p5[4] & p5[4] <= p5[5])
  print("Checking: checking elements of p5")
})
```

```
## [1] "Checking: checking elements of p5"
## Test passed
```

6. [1 point] Did you get close to the true value? Explain why there is variation in the proportions across the simulations.

```
BEGIN QUESTION
name: p6
manual: true
```

The true value is $1/33 = 0.03$ percent, which is very close to the median of the simulated values. There is variation across the simulations because the data was generated randomly, so some samples had more birth defects than others, even though they were all drawn from the same underlying distribution.

7. [1 point] **Suppose that rather than simulating 5 samples of size 200, we simulated 5 samples of size 1000. In 1-2 sentences, how would you expect the group of proportion estimates from the question above to be different? Comment both on the accuracy of these values at predicting the true value and their variance. If you're not sure, you can re-run your simulation with a larger sample size and see how the results change to deduct the difference.**

```
BEGIN QUESTION
name: p7
manual: true
```

The proportion estimates would be less variable (because sample size is larger) and closer to the true underlying value.

Part 2: Probability of HIV and Hepatitis C

Approximately 1.1 million Americans have HIV and 3.5 million Americans have Hepatitis C (HCV). The number of individuals with coinfection (e.g. both HIV and HCV) is 300,000. Among individuals with HIV, approximately 25% have Hepatitis C. The total US population was approximately 321 million at the time of these statistics.

References for these stats:

- https://www.cdc.gov/hiv/basics/statistics.html
- https://www.cdc.gov/media/releases/2016/p0504-hepc-mortality.html
- https://www.cdc.gov/hepatitis/populations/hiv.htm

8. [2 points] Calculate the probability that a randomly chosen American will have HIV. Then calculate the probability that a randomly chosen American will have HCV. Convert to percentages and round to two decimal places. Save the HIV and HCV probabilities (without the %) to the vector called p8.

```
BEGIN QUESTION
name: p8
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p8 <- c('YOUR ANSWER HERE')
p8
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p8 <- c(0.34, 1.09)
# END SOLUTION
```

```
## Test ##
test_that("p8a", {
  expect_true("numeric" %in% class(p8))
  print("Checking: p8 is  a numeric vector")
})
```

```
## [1] "Checking: p8 is  a numeric vector"
## Test passed
```

```
## Test ##
test_that("p8b", {
  expect_true(length(p8) == 2)
  print("Checking: p8 has two values")
})
```

```
## [1] "Checking: p8 has two values"
## Test passed
```

```
## Test ##
test_that("p8c", {
  expect_true(all.equal(p8[1], 0.34, tol = 0.1) & all.equal(p8[2], 1.09, tol = 0.1))
  print("Checking: values of p8")
})
```

```
## [1] "Checking: values of p8"
## Test passed
```

Let HIV represent HIV and HCV represent HCV. P(HIV) = 1.1M/321M = 0.003426791 = 0.34% P(HCV) = 3.5M/321M = 0.01090343 = 1.09%

9. [2 points] Without using the total number of individuals with co-infections (300,000), calculate the probability that someone will have both HIV and HCV. Convert your answer to a percentage rounded to two decimal places and save it to the object called `p9`.

```
BEGIN QUESTION
name: p9
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p9 <- NULL # YOUR CODE HERE
p9
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p9 <- 0.09
# END SOLUTION
```

```
## Test ##
test_that("p9a", {
  expect_true(is.numeric(p9))
  print("Checking: p9 is a number")
})
```

```
## [1] "Checking: p9 is a number"
## Test passed
```

```
## Test ##
test_that("p9b", {
  expect_true(round(p9, 2) == p9)
  print("Checking: rounded to two decimals")
})
```

```
## [1] "Checking: rounded to two decimals"
## Test passed
```

```
## Test ##
test_that("p9c", {
  expect_true(all.equal(p9, 0.09, tol = 0.1))
  print("Checking: value of p9")
})
```

```
## [1] "Checking: value of p9"
## Test passed
```

We know that P(HCV | HIV) = 0.25

We also know that P(HCV | HIV) = P(HCV & HIV)/P(HIV) This implies that P(HCV & HIV) = P(HCV | HIV) X P(HIV) = 0.25 X (1.1/321) = 0.0008566978 = 0.09%

10. [2 points] Are HIV and HCV infections independent? Uncomment your selection.

```
BEGIN QUESTION
name: p10
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
# p10 <- 'independent'
# p10 <- 'not independent'
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p10 <- "not independent"
# END SOLUTION
```

```
## Test ##
test_that("p10", {
  expect_true(p10 == "not independent")
  print("Checking: p10 selection")
})
```

```
## [1] "Checking: p10 selection"
## Test passed
```

If HIV and HCV were independent, then:

$P(HIV\&HCV) = P(HCV) \times P(HIV) = 0.01090343 \times 0.00342679 = 0.00003736378 = 0.003736378\%$

However, from part b) we know that P(HIV & HCV) = 0.09%, which is much higher. Thus these infections are not independent.

Alternative explanation:

We know that P(HCV) = 1.09% from part (a) and that P(HCV | HIV) = 25% from the question. If these infections were independent, these two quantities would be the same. Thus, they are not independent.

11. [2 points] In general, does P(A|B) equal P(B|A)? Calculate P(HIV | HCV) and report whether or not it is equal to P(HCV | HIV).

```
BEGIN QUESTION
name: p11
manual: true
```

Using information from part (b) and part (a):

[1 mark] P(HIV | HCV) = P(HIV & HCV)/P(HCV) = 0.0008566978/0.01090343 = 7.86%

[0.5 marks] The P(HIV | HCV) is 7.86% which is different from P(HCV | HIV) which is 25%.

Part 3: Screening for Lung Cancer

Background reading: Read pages 258-261 (and optionally 261-264) of Baldi & Moore, Edition 4. (For earlier editions, look for the section on diagnostic testing in medicine or on screening which covers sensitivity, specificity, negative predictive value, and positive predictive value).

Lung cancer is a leading cause of cancer-related deaths in the United States. Researchers examined the idea of testing all Medicare-enrolled heavy smokers for lung cancer with a computed tomography (CT) scan every year. In this population, the lifetime chance of developing lung cancer is high. In any given year, approximately 3% of heavy smokers develop lung cancer. The CT scan positively identifies lung cancer 89% of the time, and it gives negative results for 93% of individuals who do not have lung cancer.

12. [3 points] Use probability notation to express the three probabilities cited. Make sure to define each event using a capital letter (or two capital letters). What do the 89% and 93% values represent?

```
BEGIN QUESTION
name: p12
manual: true
```

[0.5 marks] Establish RVs as letters: Let L represent the event lung cancer and CT represent a positive CT scan.

[1 mark] P(CT | L) = 0.89. This is the sensitivity of the test. [1 mark] P(CT' | L') = 0.93. This is the specificity of the test. [0.5 marks] P(L) = 0.03. This is the marginal probability of lung cancer.

13. [3 points] What percent of CT scans in this target population would be positive? Answer this question by making either a probability tree or using absolute frequencies. Show your work.

```
BEGIN QUESTION
name: p13
manual: true
```

Solution 1. Absolute frequencies. Suppose that there were 1000 people in the target population.

- 30 of them truly have lung cancer. Of these 30, 26.7 will test positive for lung cancer. The remaining 3.3 will test negatively.
- 970 of them will not have lung cancer. Of these, 902.1 will correctly test negative, will the remaining 67.9 incorrectly test positive
- The total number of positive tests is the sum of the true positives and the false positives. This is: 26.7 + 67.9 = 94.6. Thus 94.6/1000 = 9.46% of the population will test positive.

Solution 2. Probability tree.

Note: If you are writing your solutions in R markdown you may want to upload an image of a hand-drawn tree diagram (this is optional). If so, use the following code. Be sure to remove the option "eval = F" if using this code or it won't run when you knit the file!

14. [1 point] Write the probability statement that represents a Medicare-enrolled heavy smoker who gets a positive scan and actually has lung cancer.

```
BEGIN QUESTION
name: p14
manual: true
```

[1 mark] We are trying to solve for P(L | CT).

15. [1 point] Calculate the probability from question 14 using your work from question 13. Store the answer as a percentage rounded to one decimal place in the object called `p15`.

```
BEGIN QUESTION
name: p15
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p15 <- NULL # YOUR CODE HERE
p15
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p15 <- 28.2
# END SOLUTION
```

```
## Test ##
test_that("p15a", {
  expect_true(is.numeric(p15))
  print("Checking: p15 is a number")
})
```

```
## [1] "Checking: p15 is a number"
## Test passed
```

```
## Test ##
test_that("p15b", {
  expect_true(round(p15, 1) == p15)
  print("Checking: rounded to one decimal")
})
```

```
## [1] "Checking: rounded to one decimal"
## Test passed
```

```
## Test ##
test_that("p15c", {
  expect_true(all.equal(p15, 28.2, tol = 0.1))
  print("Checking: value of p15")
})
```

```
## [1] "Checking: value of p15"
## Test passed
```

Based on absolute frequencies: Of the 94.6 people with a positive scan, 26.7 will actually have lung cancer. Thus, 26.7/94.6 = 28.2% of the positive scans will actually have lung cancer.

Using the probability tree: 0.0946 of the population will have a positive scan, of which 0.0267 will actually have lung cancer. Thus 0.0267/0.0946 = 28.2% of the positive scans will actually have lung cancer.

16. [1 point] What is the definition of the value you calculated in the previous question? Uncomment your selection.

```
BEGIN QUESTION
name: p16
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
# p16 <- 'Positive Predictive Value'
# p16 <- 'Negative Predictive Value'
# p16 <- 'Sensitivity'
# p16 <- 'Specificity'
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p16 <- 'Positive Predictive Value'
# END SOLUTION
```

```
## Test ##
test_that("p16", {
  expect_true(tolower(p16) == "ppv" | tolower(p16) == "positive predictive value")
  print("Checking: solution to p16")
})
```

```
## [1] "Checking: solution to p16"
## Test passed
```

Part 4: Prenatal Microdeletion Screening

The following questions are based on a New York Times article published on January 1, 2022.

In 2020, a biotech company called Natera conducted 400,000 prenatal screening tests for DiGeorge Syndrome, which is a disorder associated with heart defects and intellectual disability. The company claimed that they could correctly identify 200 true cases of the disorder while identifying an equal number of false positives. However, the data showed that they incorrectly identified 606 tests as positive cases. Assume the screening test has a 93% sensitivity.

17. Fill out the table based on the information above. Round your answers to the nearest whole number.

```
BEGIN QUESTION
name: p17
manual: true
```

	True +	True -	Total
test +			
test -			
Total			

	True +	True -	Total
test +	200	606	806
test -	15	399179	399194
total	215	399785	400,000

18. [1 point] Calculate the positive predictive value for this screening test.

```
BEGIN QUESTION
name: p18
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p18 <- NULL # YOUR CODE HERE
p18
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p18 <- 200/806
# END SOLUTION
```

```
## Test ##
test_that("p18a", {
  expect_true(is.numeric(p18))
  print("Checking: p18 is a number")
})
```

```
## [1] "Checking: p18 is a number"
## Test passed
```

```
## Test ##
test_that("p18b", {
  expect_true(all.equal(p18, 0.248139, 0.01))
  print("Checking: correct PPV")
})
```

```
## [1] "Checking: correct PPV"
## Test passed
```

19. Why does the test have a high sensitivity but a low PPV?

```
BEGIN QUESTION
name: p19
manual: true
```

The prevalence of DiGeorge Syndrome is low. In other words, since it is a rare disease, there are fewer chances of true positive tests and many chances for false positive tests.

END