Data 8
Spring 2019

Foundations of Data Science

Midterm Solutions

INSTRUCTIONS

- The exam is worth 100 points. You have 100 minutes to complete it.
- The exam is closed book, closed notes, closed computer/phone/tablet, closed calculator, except the official midterm exam reference guide provided with the exam.
- Write/mark your answers on the exam in the space/bubbles provided. Answers written anywhere else will not be graded.
- If you need scratch paper, you are welcome to use the reference sheet and the back of this cover page. Scratch work will not be graded.
- For all Python code, you may assume that the statements `from datascience import *` and `import numpy as np` have been executed. Do not use features of the Python language that have not been described in this course.
- In any part, you are free to use any tables, arrays, or functions that have been defined in previous parts of the same question, and you may assume they have been defined correctly.

Last name	
First name	
Student ID number	
Calcentral email (`_@berkeley.edu`)	
Lab GSI	
Name of the person to your left	
Name of the person to your right	
All the work on this exam is my own. **(please sign)**	

1. (10 points) Python Expressions

For each of the Python expressions in the table below, write the output when the expression is evaluated. If an error occurs, write "Error". The first row provides an example.

Example Expression: `make_array(1,2,3,4,5) == 3`

Example Answer: `array([False, False, True, False, False])`

(a) **(2 pt)** `make_array(1,1) * np.arange(1,10,5)`

`array([1,6])`

Be lenient in grading the way arrays are written out; for example, accept `[1,6]` as correct.

(b) **(2 pt)** `make_array(3,4,8) + np.arange(2,7,1)`

`Error`

(c) **(2 pt)** `np.average(np.arange(1,10,4))`

`5.0` (also accept 5)

(d) **(2 pt)** `make_array(1,2,3,4) + 2`

`array([3,4,5,6])`

(e) **(2 pt)** `"I love Data "+ 8`

`Error`

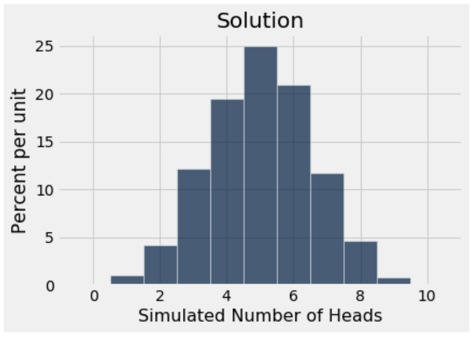
2. (8 points) Statistical Inference and Empirical Distributions

(a) **(2 pt)** A data scientist performs a statistical test. The null hypothesis is that a specified chance model is good and the alternative hypothesis is that the model is not good. The data scientist decides to use 3% as the cutoff for the P-value of the test.

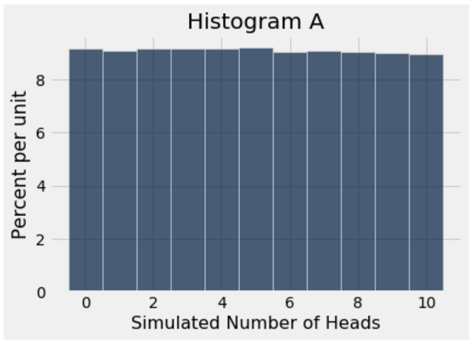
Pick **one** option: If the model is good, the chance that the test will conclude that the data are consistent with the model is:

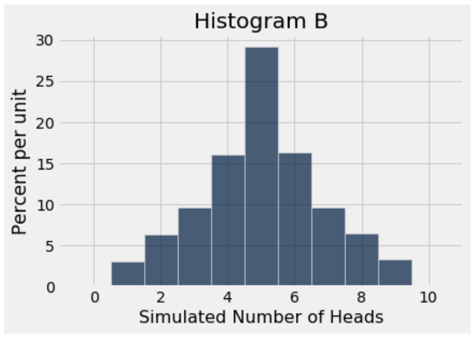
- ○ 0%
- ○ about 1.5%
- ○ about 3%
- ○ 50%
- ○ about 94%
- [✓ about 97%]
- ○ about 98.5%
- ○ 100%
- ○ not possible to approximate based on the information given

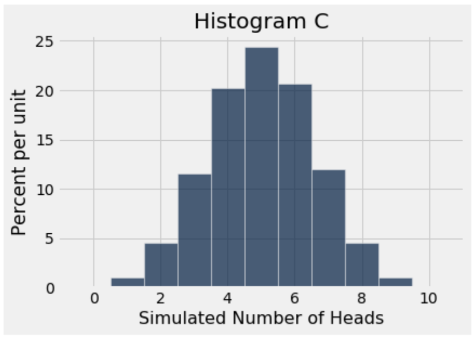
(b) **(3 pt)** Students in a data science class are asked to draw a histogram of 10,000 simulated values of the number of heads in 10 tosses of a fair coin. The answer in the solution set is below:



By mistake, a student drew a histogram of 100,000 simulated values instead of 10,000. One of Histograms A, B, and C is the student's answer. Which one is it?







○ Histogram A　○ Histogram B　[✓ Histogram C]

(c) **(3 pt)** When the following code is run, the expression in the last line evaluates to a number that is approximately one of the values below. Which one?

```
counter = 0
for i in np.arange(100000):
    rolls = np.random.choice(np.arange(1, 7), 10)
    if np.count_nonzero(rolls > 4) == 0:
        counter = counter + 1
counter / 100000
```

○ 1/3	○ 2/3
○ 10*(1/3)	○ 1 - 10*(1/3)
○ (1/3)**10	○ 1 - (1/3)**10
[✓ (2/3)**10]	○ 1 - (2/3)**10
○ 10*(2/3)	○ 1 - 10*(2/3)

3. (20 points) Counties

Every state in the United States is divided into counties that do not overlap with each other and together cover the whole state. A table `counties` contains one row for each county in the United States:

State	County	2010 Pop	2014 Pop
Alabama	Autauga County	54684	55395
Alabama	Baldwin County	183216	200111
Alabama	Barbour County	27336	26887

... (3139 rows omitted)

The table contains four columns:

- **State**: a string, the name of the state
- **County**: a string, the name of the county
- **2010 Pop**: an int, the population in 2010 (as estimated by the US Census Bureau)
- **2014 Pop**: an int, the population in 2014 (as estimated by the US Census Bureau)

In addition, we have the function `first` defined below:

```
def first(x):
    """x is an array"""
    return x.item(0)
```

In each part below, fill in the blanks of the Python expression. **You must use ONLY the lines provided.** Some of the chained operations we might normally do in one line have been broken up into two or more lines, storing intermediate results in temporary tables. Do not write any code outside the blanks provided. The expression in the last line should evaluate to the value described.

(a) (2 pt) The name of the largest county in the United States (by 2014 population):

```
sorted = counties.sort('2014 Pop', descending = True)
first(sorted.column('County'))
```

(b) (2 pt) How many counties saw their populations grow by more than 10,000 people between 2010 and 2014:

```
counties_with_change = counties.with_column('Pop Change',
        counties.column('2014 Pop') -
        counties.column('2010 Pop'))
counties_with_change.where('Pop Change', are.above(10000)).num_rows
```

(c) (2 pt) A new table called `states` which has one row for each state. It should have three columns: the state's name, the total population in the state in 2010, and the total population in 2014. It should not have any column corresponding to county names.

```
counties_3column = counties.drop('County')
states = counties_3column.group('State', sum)
states
```

(d) (4 pt) A new table called `biggest_county` which has one row for each state corresponding to the largest county in that state (by 2014 population). Its columns should be the state's name, the name of the largest county (by 2014 population), the 2010 population of that county, and the 2014 population of that county. It doesn't matter what the column names are.

```
sorted = counties.sort('2014 Pop', descending = True)
biggest_county = sorted.group('State', first)
biggest_county
```

Don't double penalize if students make the same mistake in the first line as in the first line of part (a).

(e) **(4 pt)** The table `biggest_county` with an additional column called `'Pct of State Population'`. It should store what percent of that state's population lived in the largest county in 2014 (for example, if a state has only three counties and the populations are 250,000, 200,000, and 50,000, `'Pct of State Population'` should be 50).

```
bc_with_state = biggest_county.join('State', states)
biggest_county_with_pct = biggest_county.with_column('Pct of State Population',
        100 * bc_with_state.column('2014 Pop first') /
        states.column('2014 Pop sum'))
biggest_county_with_pct
```

(f) **(6 pt)** The table `states` with an additional column called `'Pop in Large Counties'`. It should contain the total number of people in each state that lived in counties with population more than 100,000 in 2014 (for example, if a state has only three counties and the populations are 250,000, 200,000, and 50,000, `'Pop in Large Counties'` should be 450,000). You may assume every state has at least one such county.

```
all_large = counties.where('2014 Pop', are.above(100000))
all_large = all_large.group('State', sum)
all_large = all_large.relabeled(3, 'Pop in Large Counties').select(0,3)
states_with_large_pop.join('State', all_large)
states_with_large_pop
```

4. (25 points) Cereals

A table `cereal` contains one row with nutritional information for each of a set of cereal brands:

name	mfr	calories	protein	fat	fiber	sugars	shelf	rating
100% Bran	N	70	4	1	10	6	3	68.403
100% Natural Bran	Q	120	3	5	2	8	3	33.9837
All-Bran	K	70	4	1	9	5	3	59.4255

... (74 rows omitted)

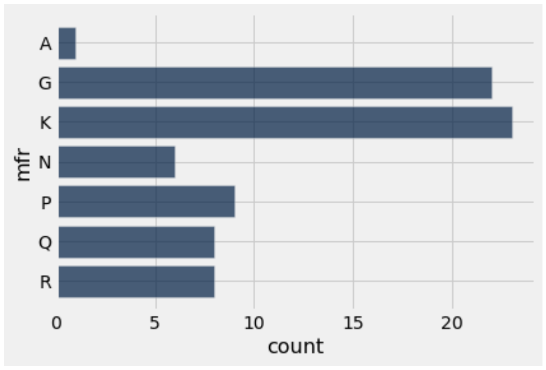
The table contains nine columns:

- **name**: a string, the name of the cereal,
- **mfr**: a string, the initial of the manufacturer:

A:	American Home Food Products	P:	Post
G:	General Mills	Q:	Quaker Oats
K:	Kelloggs	R:	Ralston Purina
N:	Nabisco		

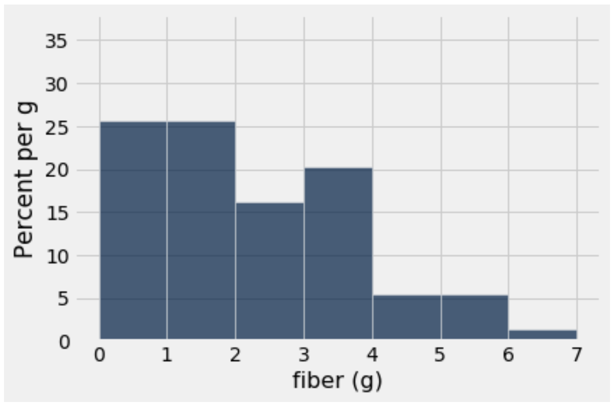
- **calories**: an int, calories per serving
- **protein**: an int, grams of protein per serving
- **fat**: an int, grams of fat per serving
- **fiber**: an int, grams of fiber per serving
- **sugars**: an int, grams of sugar per serving
- **shelf**: an int, what shelf the cereal is displayed on at a certain supermarket (1, 2, or 3)
- **rating**: a float, giving a rating of the cereal on a scale of 1–100 from Consumer Reports (CR)

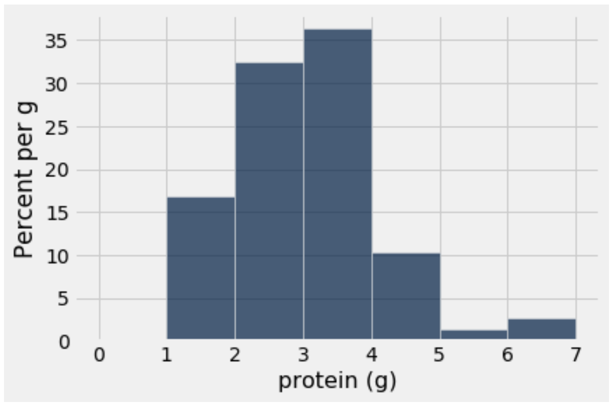
(a) (1 pt) Which of the following lines of code would produce the figure below?



- ○ cereal.hist('mfr')
- ○ cereal.barh('mfr')
- ○ cereal.hist('count')
- ● cereal.group('mfr').barh('mfr')
- ○ cereal.group('mfr').plot('mfr')

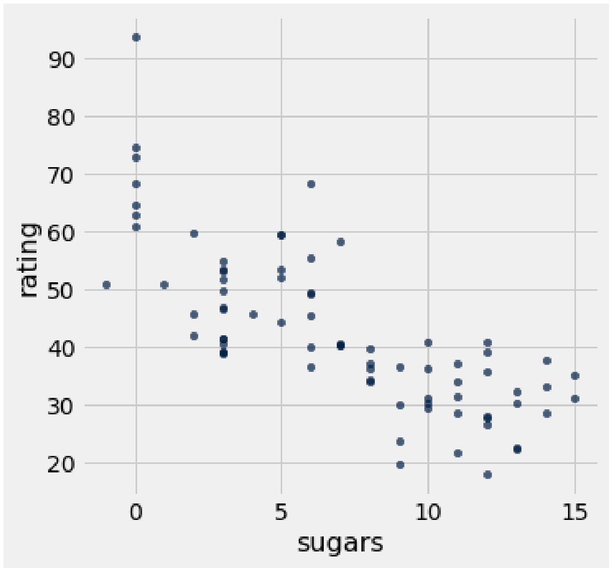
(b) **(3 pt)** Which conclusions are justified by the the two histograms below? The bins in both histograms are generated by `bins = np.arange(0,16)`. Remember that all data are integers. **Bubble in all correct answers**.





- [✓] About half of cereals have 0 or 1 grams of fiber per serving.
- ○ Most cereals have more grams of protein than grams of fiber per serving.
- ○ Most cereals have more grams of fiber than grams of protein per serving.
- [✓] There are more cereals with at least 5 grams of fiber per serving than there are cereals with at least 5 grams of protein per serving.
- ○ Cereals with more fiber per serving tend to have more protein per serving.

(c) **(3 pt)** Which conclusions are justified by the scatter plot below? **Bubble in all correct answers**.



- ○ There is an apparent positive association between the amount of sugar per serving and the CR rating
- [✓] There is an apparent negative association between the amount of sugar per serving and the CR rating
- ○ There is no apparent association between the amount of sugar per serving and the CR rating
- ○ For most cereals, their CR ratings would likely be reduced if the manufacturer added more sugar.
- ○ For most cereals, their CR ratings would likely be increased if the manufacturer added more sugar.
- ○ For most cereals, their CR ratings would likely remain unchanged if the manufacturer added more sugar.

(d) **(4 pt)** Suppose we want to predict the CR rating for a new cereal based on its amount of sugar, by averaging the ratings of all cereals whose sugar content per serving differs by no more than 1 gram. Fill in the blanks below to define a function called `predict_rating` that takes in an amount of sugar in grams and returns the prediction for the CR rating.

```
def predict_rating(sugar):
    comparison_group = cereal.where(sugars’, are.between(sugar - 1.1, sugar 1.1))+
    return np.mean(comparison_group.column(’rating’))
```

(also accept if they did 1 instead of 1.1, even though technically this would exclude sugar + 1. Something like 1.5 is equivalent to 1.1.)

(e) **(4 pt)** Using the function `predict_rating` (you may assume it is correctly defined), in approximately which range will the predicted rating be for a cereal with 1 gram of sugar per serving?

- between 30 and 35
- between 40 and 45
- between 50 and 55
- [✓ between 60 and 65]

(f) **(4 pt)** Fill in the blanks below to add a new column to the table `cereal` containing the predictions from the `predict_rating` function for each row. The column should be called `’Predicted Rating’`.

```
predictions = cereal.apply(predict_rating, ’sugars’)
cereal = cereal.with_column(’Predicted Rating’, predictions)
```

(g) **(2 pt)** Next, we want to see how well our predictions are doing on average. Write a Python expression to compute the average absolute error for all of our predictions (the absolute error is the absolute value of the difference between the predicted rating and the actual rating).

```
avg_abs_error = np.average(abs(cereal.column(’Predicted Rating’) - cereal.column(’rating’)))
avg_abs_error
```

(h) **(4 pt)** Complete the Python expression that would produce the table below, which summarizes the average number of calories per serving in cereal brands grouped by manufacturer **and** shelf.

```
cereal.pivot(’mfr’,’tier’,’calories’,np.average)
```

shelf	A	G	K	N	P	Q	R
1	0	106.667	107.5	86.6667	105	100	102.5
2	100	111.429	111.429	95	110	113.333	0
3	0	114.444	107.5	70	110	80	127.5

5. (12 points) Grabbing Socks

Professor Fithian stays up late laundering his socks and sleeps in one day when he is going to give his Data 8 lecture. While running out the door in a state of panic, he grabs two socks completely at random from the dryer, which contains 28 total socks: 16 black socks, 10 white socks, and 2 lime green socks.

The socks are not in pairs and haven't been touched since they were "shuffled" by the dryer the night before, so the two socks are two draws at random without replacement.

In each part below write a mathematical expression (not Python) that evaluates to the probability described. **You do not need to simplify any arithmetic. Please do not multiply by 100 to get percents.**

(a) (3 pt) The probability that both of the socks are black:

```
(16 / 28) * (15 / 27)
```

(b) (3 pt) The probability that at least one sock is lime green sock:

```
1 - (26 / 28) * (25 / 27)
```

(c) (3 pt) The probability that one sock is black and one is white:

```
(16 / 28) * (10 / 27) + (10 / 28) * (16 / 27)
```

Also OK: something like:

```
2 * (16 / 28) * (10 / 27)
```

(d) (3 pt) The probability that the two socks **are not** the same color:

```
1 - ( (16 / 28) * (15 / 27) + (10 / 28) * (9 / 27) + (2 / 28) * (1 / 27) )
```

also accept `16/28 * 12/27` $10/28 * 18/27 + 2/28 * 26/27+$

6. (19 points) Matching Socks

After lecture, while trying to reassure Prof. Fithian about his mismatched socks (see last question), Prof. Adhikari tells him that the same thing happens to her all the time. In fact, she claims that every time she gives a lecture there is a 25% chance her socks will be mismatched, regardless of whether they match on any other day.

You overhear this conversation and you strongly suspect that she is just exaggerating to make Prof. Fithian feel better; that is, you believe her socks aren't really mismatched as often as she claims. You decide to put her claim to the test by watching videos of the last 100 Data 8 lectures she has given.

In all of the questions below, a *mismatched lecture* is a lecture in which the professor's socks don't match, and a *matched lecture* is a lecture in which the professor's socks do match.

(a) (4 pt) Select the null hypothesis from the options below. **Bubble in all answers that are correct** (or that can be correct after blanks are filled in appropriately), and fill in any blanks for the answers you bubble in. You do **not** need to fill in the blanks for answers you do not bubble in.

- ● In 100 lectures, Prof. Adhikari wears mismatched socks about ________ times,

  and any deviation from that is ______________________________________.

- ○ In any given lecture, Prof. Adhikari's socks are just as likely to be mismatched as they are to be matched, and any difference between the number of matched lectures and mismatched lectures

  is ______________________________________.

- ○ Prof. Adhikari is just as likely to sleep in on the day of a lecture as Prof. Fithian is, and any difference between the number of times the two professors sleep in is

  is ______________________________________.

- ● If we count the number of mismatched and matched lectures out of 100 of Prof. Adhikari's lectures,

  it will be like a sample ____________ replacement from a distribution that has a

  ____________ chance of being mismatched and a ____________ chance of being matched.

- ○ Prof. Adhikari wears mismatched socks as often as Prof. Fithian wears mismatched socks, and any difference in the fraction of mismatched lectures between the two professors

  is ______________________________________.

- ○ In 100 past lectures of Data 8, Prof. Adhikari wore mismatched socks exactly ____ times.

1. In 100 lectures, Prof. Adhikari wears mismatched socks about 25 times, and any deviation from that is due to random chance.

2. If we count the number of mismatched and matched lectures out of 100 of Prof. Adhikari's lectures, it will be like a sample with replacement from a distribution that has a 25% chance of being mismatched and a 75% chance of being matched.

(b) **(4 pt)** Select the alternative hypothesis from the options below. **Bubble in all answers that are correct** (or that can be correct after blanks are filled in appropriately), and fill in any blanks for the answers you bubble in. You do **not** need to fill in the blanks for answers you do not bubble in.

- ○ Prof. Adhikari is less likely than Prof. Fithian

  to ____.

- ○ Prof. Adhikari is more likely than Prof. Fithian

  to ____.

- ○ Prof. Adhikari and Prof. Fithian are not equally likely

  to ____.

- ● The probability Prof. Adhikari wears mismatched socks in a lecture is less than ________.
- ○ The probability Prof. Adhikari wears mismatched socks in a lecture is more than ________.
- ○ The probability Prof. Adhikari wears mismatched socks in a lecture is different from ________.

The probability Prof. Adhikari wears mismatched socks in a lecture is less than 25%.

(c) **(2 pt)** You watch the videos for the 100 past lectures, and carefully count how many times her socks were mismatched. What is a good test statistic to use if you want to test the null hypothesis?

- ● The number of lectures with mismatched socks minus ________
- ○ The distance between the number of lectures with mismatched socks and ________
- ○ The number of mismatched lectures for Prof. Adhikari minus the number of matched lectures
- ○ The distance between the number of mismatched and matched lectures for Prof. Adhikari
- ○ The number of mismatched lectures for Prof. Adhikari minus the number for Prof. Fithian
- ○ The distance between the number of mismatched lectures for Prof. Adhikari and Prof. Fithian

25% minus the faction of mismatched lectures, or 25 minus the number of mismatched lectures.

The distance from 25%, or from 25 lectures, is **not** a good test statistic because we should be testing the alternative that he's mismatched less than 60% of the time. But give credit if the student chose a two-sided alternative in part (b).

(d) **(6 pt)** Fill in the missing Python code below to simulate the value of your test statistic under the null hypothesis 10,000 times. The last line should evaluate to an array of 10,000 simulated values.

```
def sim_discrepancy():
""" function to simulate one draw of the test statistic"""
    return 25 - 100 * sample_proportions(100, make_array(0.25, 0.75)).item(0)
```

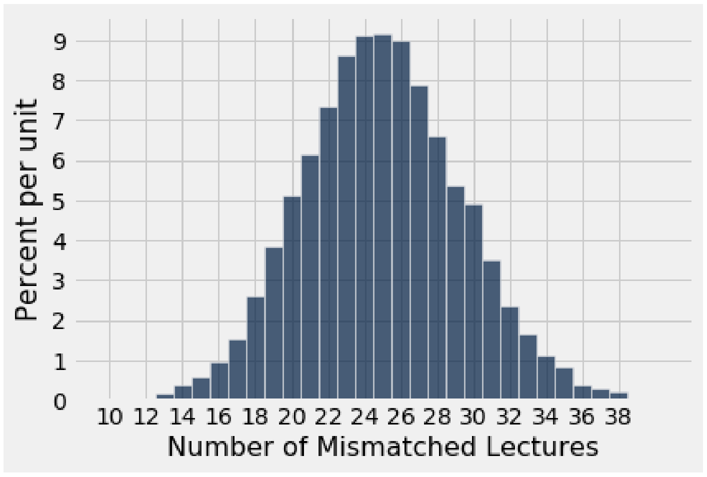
Give credit for `abs(...)` if the student gave a two-sided alternative in part (b)

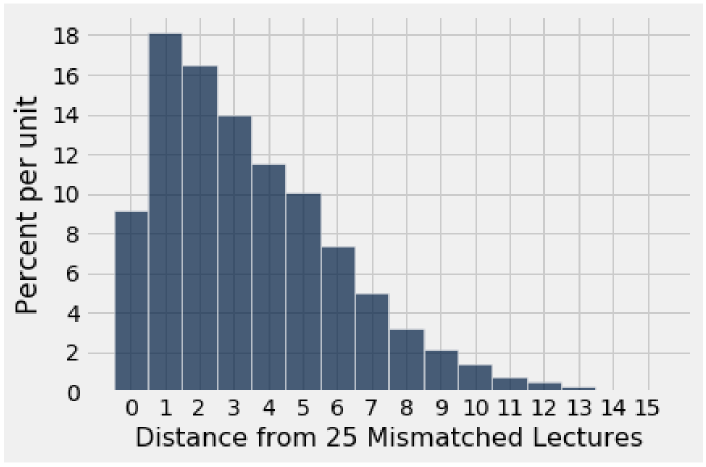
```
discrepancies = make_array()


## loop to simulate 10,000 draws of the test statistic:
for i in np.arange(10000):
    new_discrepancy = sim_discrepancy()
    discrepancies = np.append(discrepancies, new_discrepancy)
discrepancies
```

(e) **(3 pt)**

The two histograms printed below show the simulated distributions of (top) the number of mismatched lectures and (bottom) the distance between 25 and the number of mismatched lectures. The simulation is carried out under the null hypothesis, with 10,000 simulation runs of 100 lectures each (this is enough simulation runs to give a good approximation to the probability distribution). Remember that all of the numbers are integers. The bins in both figures all have width 1 and are centered at integers.





In your 100 videos, Prof. Adhikari's socks are mismatched 16 times. Use whichever histogram above is more useful for evaluating the p-value using the test statistic you defined in part (c).

From looking at the histogram above, which answer is closest to the p-value for your hypothesis test?

○ 0% ○ 1% ● 2% ○ 4% ○ 10%

Should be about 2%: 1% for the bar including 16, and about 1% more for the bars including 13, 14, and 15.

Give credit for 4% if the student defined a two-sided test statistic in part (c).

7. (6 points) Blood Pressure

In a randomized controlled experiment, 300 patients are randomized into the treatment group (Group A) and 200 to the control group (Group B). At the end of the experiment, each patient's blood pressure is measured. The research team wants to test the null hypothesis that the treatment has no effect versus the alternative hypothesis that the treatment has an effect. As their test statistic they decide to use the absolute difference between the average blood pressures of Group A and Group B.

The table `data` contains has 500 rows, one for each patient in the experiment. The table has just one column. The column is labeled `'bp'` for blood pressure, and contains a numerical measurement of blood pressure for each patient. All measurements are in the same units.

The table `data` has no other information. If possible, fill in the blanks in the code below so that the last line evaluates to one value of the test statistic simulated under the null hypothesis. If this is not possible, pick the option below the code and give a brief explanation.

- ◯ Possible using the information given, using the following code:

```
shuffled = data.sample(with_replacement = False)

# Two tables

group_A_shuffled_table = shuffled.___________________(______________________________)

group_B_shuffled_table = shuffled.___________________(______________________________)

# Two averages

shuffled_mean_A = np.average(group_A_shuffled_table.column('bp'))

shuffled_mean_B = np.average(group_B_shuffled_table.column('bp'))

# Test statistic

abs(shuffled_mean_A - shuffled_mean_B)
```

- ◯ This is not possible with the information given. Explain below:

```
shuffled = data.sample(with_replacement = False)


# Two tables

group_A_shuffled_table = shuffled.take(np.arange(300))

group_B_shuffled_table = shuffled.take(np.arange(300,500))


# Two averages

shuffled_mean_A = np.average(group_A_shuffled_table.column('bp'))

shuffled_mean_B = np.average(group_B_shuffled_table.column('bp'))
```

```
# Test statistic

abs(shuffled_mean_A - shuffled_mean_B)
```

8. (0 points) **W**rite your name in the space provided on one side of every page of the exam. You're done!