Spring 2024 Midterm Exam

Foundations of Data Science

Name

Total Score: __________ of 100 Points

Instructions

- Make sure to rip off the Table Reference and Midterm Exam Reference Guide attached at the end of the exam.
- Select the correct response(s) or provide a written response depending on the question type. If a prompt asks you to write code, then you can provide your own code or use the provided template. Try to provide your responses in the spaces provided. If you find that you need additional space, write your extended response(s) on one of the provided blank sheets of paper and number them, so we can connect your response to the question.
- You can assume the following code has been run, when you are writing your Python code:

```
from datascience import *
import numpy as np
import matplotlib+
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

- The Multiple choice questions (◯) and multiple answer questions (☐) will be scored like in Canvas.
- The open response questions will be graded as:
  - Full Points: The response is correct and may contain a very very small error.
  - Partial Points: A reasonable response was provided. The partial point value will depend on your response.
  - No Points: No reasonable attempt was provided.
- Once you are finished, turn in your exam and you are welcome to leave.

1. (3 points) In an effort to prove that a new treatment is effective at reducing hip pain, researchers randomly sample 1,000 from a relevant population, ask them each whether they'd like to receive a placebo or the new treatment, assign them to that treatment, and then assess the pain levels after the full course of the treatment. Overall, those who received the treatment observed a significant reduction in pain. Which of the following can you say is true based on the provided information? Select all that apply.

   ☐ This an experiment.

   ☐ This is a randomized experiment.

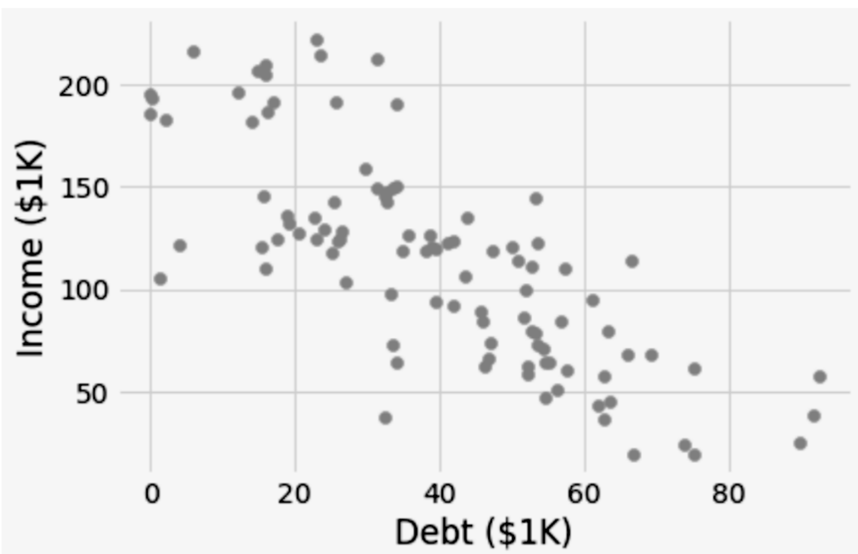
   ☐ This is a randomized controlled experiment.

   √ **This is an observational study.**

   ☐ The new treatment causes a reduction in pain.

   √ **There is a significant association between the new treatment and pain reduction.**

2. (3 points) Suppose you are curious about the financial situations of recent Berkeley graduates. You have data on 200 recent graduates. Included in the data set are the starting salaries for each of the graduates (`'Income($1K)'`) and their unpaid student debt (`'Debt($1K)'`). In order to understand how a graduate's debt might be associated with their salary, you make the following scatterplot:



   Which of the following are valid conclusions that can be drawn from this graph above? Choose all that apply.

   ☐ There is a positive association between student debt and salary.

   √ **There is a negative association between student debt and salary.**

   ☐ There is no association between student debt and salary.

   ☐ There are no Berkeley graduates with a debt greater than $100K.

   √ **Among the graduates surveyed, at least 3 of them have debt greater than $80K.**

   ☐ Among the graduates surveyed, higher debt caused them to have lower starting salaries.

3. A real estate company has a dataset of all their buildings, with three attributes for each building: its size (in square feet), its type (residential or commercial), and its estimated value (sale price) if sold (in dollars).

   (a) (2 points) The standard visualization to understand the distribution of building types is: (choose one)

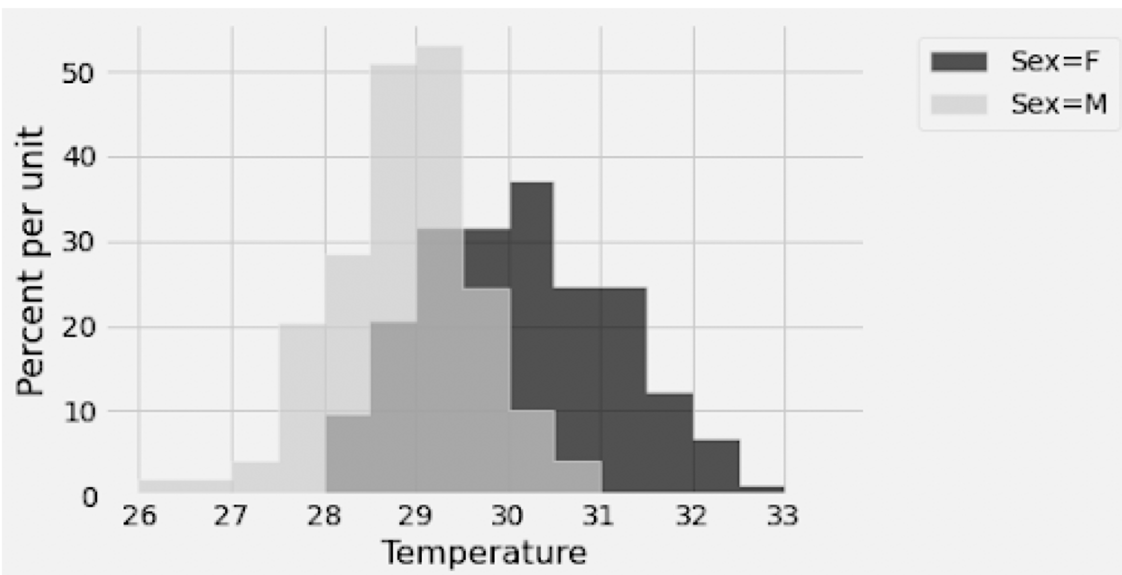
   √ **A bar chart** ◯ A line plot ◯ A scatter plot ◯ A histogram

   (b) (2 points) The standard visualization to check for an association between building size and building value is: (choose one)

   ◯ A bar chart ◯ A line plot √ **A scatter plot** ◯ A histogram

4. When hatching a baby turtle from an egg, we incubate the egg at some temperature. A researcher read that the temperature of which an egg is incubated influences whether or not the turtle hatches male or female. To test this, they randomly sample turtle eggs, and record the incubation temperature (in Celsius) and the sex of the turtle that hatches. The following histogram shows the distribution of temperatures based on the sex of the turtle.

   You can assume that 100% of the data is captured in this visualization.



   (a) (2 points) In the sample more than 50% of the male turtles were incubated at a temperature between 29.0 and 29.5 degrees.

   ◯ True
   √ **False**
   ◯ This is not possible to determine based on the provided information.

   (b) (2 points) In this sample, the number of male turtles with incubation temperatures between 29.5 and 30 degrees is the same as the number of female turtles incubated between 30.5 and 31 degrees.

   ◯ True
   ◯ False
   √ **This is not possible to determine based on the provided information.**

(c) (3 points) If the bins used to form the histogram for female turtles were replaced with a single bin from 28 to 33, how tall would the resulting bar be? Make sure to include the units in your answer.

Sample Solution: The width of the bin would be 33 - 28 = 5 degrees. The bin would create 100% of the female turtle data. Together, this means that the height of the resulting bar would be 100% / 5 degrees = 20 percent per degree.

5. In San Francisco, the Existing Buildings Energy Performance Ordinance (Environment Code Chapter 20) requires that each non-residential building with at least 10,000 square feet of conditioned (heated or cooled) space and each residential building with at least 50,000 square feet of conditioned space must be benchmarked using Energy Star Portfolio Manager annually. Each non-residential building specified above is also required to undergo an energy audit or retrocommissioning at least once every 5 years.

   The table `building_data` contains relevant San Francisco building information and 2021 energy use (measured in thousands of BTUs (British thermal units)). On the Table Reference page, you can see a preview of this table.

   (a) (4 points) How many `'Commercial'` buildings are there in `building_data`.

```
commercial_buildings = _____(a)_____.____(b)______(_____(c)_____, ____(d)______)
commercial_buildings.______(e)______
```

Sample Solution:

```
commercial_buildings = building_data.where('property_type', 'Commercial')
commercial_buildings.num_rows
```

(b) (4 points) What is the address for the building with the largest floor area? You can assume there is a unique building with the largest floor area.

```
sorted_data = ____(a)______.____(b)______(_____(c)_____, _____(d)_____)
____(e)______._____(f)_____(____(g)______)._____(h)_____
```

Sample Solution:

```
sorted_data = building_data.sort('floor_area', True)
sorted_data.column('building_address').item(0)
```

(c) (2 points) You've received a CSV file called `zip_code.csv`. Write code that will create a table called `zip_codes` from that CSV file that contains all the information in the `zip_code.csv` file. On the Table Reference page, you can see a preview of what `zip_codes` looks like. Zip codes and postal codes are equivalent in this context.

Sample Solution:

```
zip_codes = Table.read_table('zip_codes.csv')
```

(d) (3 points) Use the join method to create a table called `building_data_geo` that adds the latitude, longitude, and population estimate information from `zip_codes` to the data in `building_data`. You do not need to do any additional sorting or re-ordering beyond using the join method. On the Table Reference page, you can see a preview of what `building_data_geo` should look like.

Sample Solution:

```
building_data_geo = building_data.join('postal_code', zip_codes, 'zip')
```

(e) (3 points) When reading the data, it seems that Python assumed the postal code (zip code) values were numerical. Write code that will check if the data type of the values in the `postal_code` column of `building_data_geo` is float. Your code should output the `bool` value `True` or `False`. As a hint, `type(2.0)` would evaluate to be float.

Sample Solution:

```
type(building_data_geo.column('postal_code').item(0)) == float
```

(f) (4 points) The postal codes in `building_data_geo` are actually float values, but they need to be strings. Create a function called `float_to_str` that takes a float and returns a string version of the float ignoring any decimal part.

For example, `float_to_str(94118.0)` should return `'94118'`.

Hints: `str(94118.0)` would create the string `'94118.0'`, not `'94118'`.

Sample Solution:

```
def float_to_str(a_float):
    return str(int(a_float))
```

(g) (3 points) Use the `float_to_str` function to create an array called `postal_codes` of the postal codes formatted as strings.

Sample Solution:

```
postal_codes = building_data_geo.apply(float_to_str, 'postal_code')
```

(h) (3 points) Update the `building_data_geo` table such that the values in the `'postal_code'` column are strings, not floats.

Hint: Remember that `postal_codes` is an array of the postal codes as strings.

Sample Solution:

```
building_data_geo = building_data_geo.with_column('postal_code', postal_codes)
```

(i) (4 points) Create a bar chart of the distribution of the postal codes in the `building_data_geo` table. Make sure the bars are in order such that the longest bars are at the top of the visualization.

```
by_zip =_____(a)_____._____(b)_____(_____(c)_____)
by_zip_sorted = _____(d)_____._____(e)_____(_____(f)_____, _____(g)_____)
_______________(h)_______________
```

Sample Solution:

```
by_zip = building_data_geo.group('postal_code')
by_zip_sorted = by_zip.sort('count', True)
by_zip_sorted.barh('postal_code')
```

(j) (4 points) Create a table with two columns showing the mean energy use for 2021 for each postal code based on the data in `building_data_geo`. Your table should have a row for each postal code showing the mean energy use for the buildings with that postal code.

```
reduced_data =  _____(a)_____._____(b)_____(_____(c)_____, _____(d)_____)
_____(e)_____._____(f)_____(_____(g)_____, _____(h)_____)
```

Sample Solution:

```
reduced_data = building_data_geo.select('postal_code', 'energy_use_2021')
reduced_data.group('postal_code', np.mean)
```

(k) (3 points) Using the data in `building_data_geo`, create a visualization to show the relationship between the floor area of a building and its energy usage in 2021.

Sample Solution:

```
building_data_geo.scatter('floor_area', 'energy_use_2021')
```

6. (4 points) Which of the following functions correctly returns the number of occurrences of a specific value in a given array? For example, `count_arr_occurences(make_array(0,1,0,5,1), 1)` should evaluate to `2` and `count_arr_occurences(make_array("a", "b", "c"), "c")` should evaluate to `1`. Select all that apply.

☐
```
def count_arr_occurences(arr, value):
    count = 0
    for i in np.arange(value):
        if arr.item(i) == value:
            count = count + 1
    return count
```

√
```
def count_arr_occurences(arr, value):
    count = 0
    for x in arr:
        if x == value:
            count = count + 1
    return count
```

☐
```
def count_arr_occurences(arr, value):
    return arr == value
```

√
```
def count_arr_occurences(arr, value):
    return np.sum(arr == value)
```

7. In a game called September, players take turns selecting tokens and making moves based on the selected tokens. During a player's turn, they randomly select one token from a container and keep it; then randomly select another token from the container and keep it; make a play based on the two tokens; and then put all the tokens back in the container for the next player. The distribution of tokens is:

- Earth Token: 21 Tokens
- Wind Token: 12 Tokens
- Fire Token: 1 Token

(a) (3 points) What is the probability that a player will select no Wind tokens when it is their turn?

> **Sample Solution:** `(22 / 34) * (21 / 33)`

(b) (3 points) What is the probability that a player will select 2 of the same kind of tokens when it is their turn?

> **Sample Solution:** `(21 / 34) * (20 / 33) + (12 / 34) * (11 / 33)`

(c) (3 points) What is the probability that a player will select at least one Wind token when it is their turn?

> **Sample Solution:** `1 - (22 / 34) * (21 / 33)`

8. According to a recent survey, 28% of surveyed adults in the United States use LinkedIn. For the sake of this question, assume that the chance of a randomly sampled adult in the United States being a LinkedIn user is 28% (independently of all others).

(a) (2 points) For which sample size below is there a higher chance that a random sample of that size will contain a percent of LinkedIn users of more than 50%?

√ **20** ◯ 1,000

(b) (2 points) According to the Law of Large Numbers (Law of Averages), with a smaller sample size the percentage of surveyed adults in that sample that use LinkedIn is more likely to be closer to 28% than a larger sample size.

◯ True √ **False**

9. In the game of Wordle, a player guesses up to 6 words until they either correctly guess the secret word of the day or run out of guesses. Their guess count is either the number of guesses needed to guess the correct word (1 through 6) or X if all 6 guesses were incorrect. For all 1,000 students who played Wordle yesterday, we have collected the proportion of students with each guess count. These proportions appear in the table below and an array called `students`.

1	2	3	4	5	6	X
0.0	0.17	0.33	0.27	0.20	0.02	0.01

```
students = make_array(0.0, 0.17, 0.33, 0.27, 0.20, 0.02, 0.01)
```

Wordle's creator, Josh Wardle, sent us the proportion of guess counts for all players who tried to guess yesterday's word in an array called `everyone`.

1	2	3	4	5	6	X
0.0	0.09	0.25	0.32	0.28	0.03	0.03

```
everyone = make_array(0.0, 0.09, 0.25, 0.32, 0.28, 0.03, 0.03)
```

(a) (2 points) What best describes the table for the students? Choose one.

◯ Probability Distribution

√ **Empirical Distribution**

(b) (3 points) What is one way to simulate randomly selecting 1,000 individuals from the population of individuals that played Wordle yesterday? Choose one.

◯ `sample_proportions(1000, students)`

√ `sample_proportions(1000, everyone)`

◯ `sample_proportions(1000, make_array('1', '2', '3', '4', '5', '6', 'X'))`

◯ `sample_proportions(1000, make_array(1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7))`

(c) (2 points) If we assume the distribution provided by Josh Wardle is similar for tomorrow, what is the chance that a randomly selected Wordle player will guess the word in less than 4 guesses?

Sample Solution: `0.00 + 0.09 + 0.25`

10. (5 points) Create a function called `roll` with arguments `k`, `n`, and `trials` that simulates trials (the number of trials) rolls of n fair 6-sided dice, and each time counts how many of those dice show `k` or higher, and then displays an empirical histogram of those counts.

For example, if `k` is 5, `n` is 3, and rolling 3 dice results in a 6, a 4, and a 5, then 2 of the 3 dice are 5 or larger (the 6 and the 5). So, `roll(5, 3, 10_000)` would output a histogram created by repeating simulation 10,000 times.

```
def ___(a)____(__(b)__, __(c)__, __(d)__):
    """Repeatedly roll n dice and check how many results are k or larger."""

    outcomes = make_array()
    possible_results = np.arange(1, 7)

    for ______________(e)________________
        rolls = _____________(f)________________
        outcomes = _______(g)________(outcomes, np.count_nonzero(rolls >= ___(h)__))

    Table().with_column('Outcomes', ______(i)______).____(j)____(bins=np.arange(30))
```

Sample Solution:

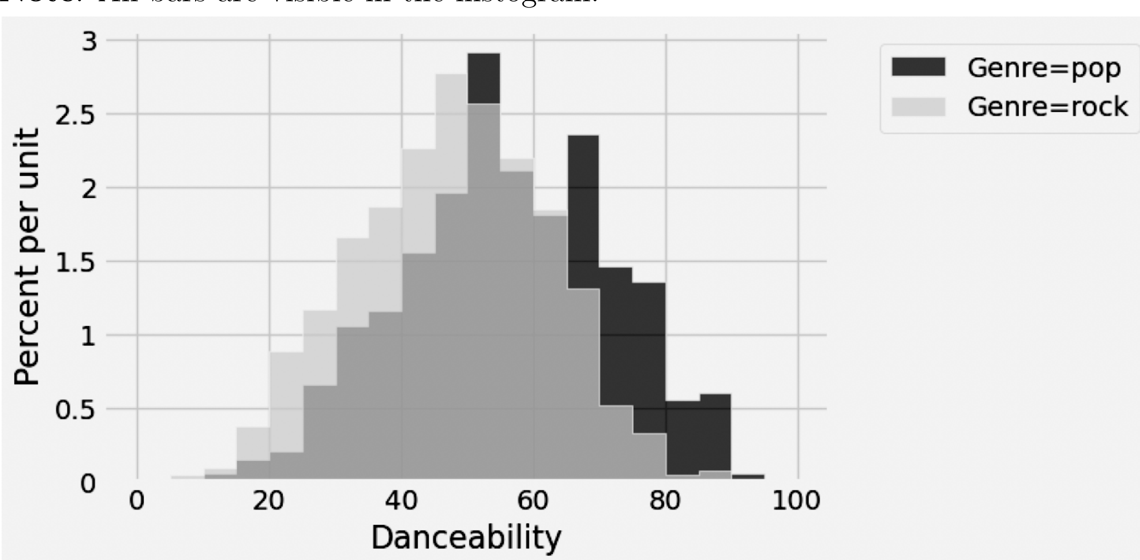
```
def roll(k, n, trials):
    """Repeatedly roll n dice and check how many results are k or larger."""

    outcomes = make_array()
    possible_results = np.arange(1, 7)

    for i in np.arange(trials):
        rolls = np.random.choice(possible_results, n)
        outcomes = np.append(outcomes, np.count_nonzero(rolls >= k))

    Table().with_column('Outcomes', outcomes).hist(bins=np.arange(30))
```

11. You want to investigate whether rock songs are less danceable than pop songs. A song's danceability is described as "... how easy it is to dance to, based on a combination of musical elements ..." and is on a scale of 0 to 100 with 100 being the most danceable. You collect a **random sample** of both rock and pop songs from Spotify's streaming platform and store the data in the `songs` table.

(a) (3 points) Suppose you visualize the danceability distribution for rock and pop songs by creating the following histograms using the line of code:

`songs.hist("Danceability", group="Genre", bins=np.arange(0, 101, 5))`

Note: All bars are visible in the histogram.



Which of the following statements are valid conclusions, just based on the histogram above? Select all that apply.

- √ **The most danceable song in the songs table was a pop song.**
- √ **Slightly more than 5% of pop songs had a danceability rating between 80 and 90.**
- ☐ Roughly the same number of pop and rock songs have a danceability rating between 60 and 65.
- √ **In this sample, rock and pop songs have different distributions of danceability ratings.**
- ☐ None of these.

(b) (2 points) Suppose you want to test whether rock songs have lower danceability ratings than pop songs, on average. Which of the following is the most appropriate null hypothesis?

- ◯ In the population of all rock and pop songs on Spotify, rock songs have lower danceability ratings than pop songs, on average.
- ◯ In the sample, rock songs have lower danceability ratings than pop songs, on average.
- ◯ In the population of all rock and pop songs on Spotify, danceability ratings for both pop and rock songs are drawn from a uniform distribution between 0 and 100.
- ◯ In the sample, the distribution of danceability ratings is the same for pop songs as for rock songs.
- √ **In the population of all rock and pop songs on Spotify, the distribution of danceability ratings is the same for pop songs as for rock songs.**

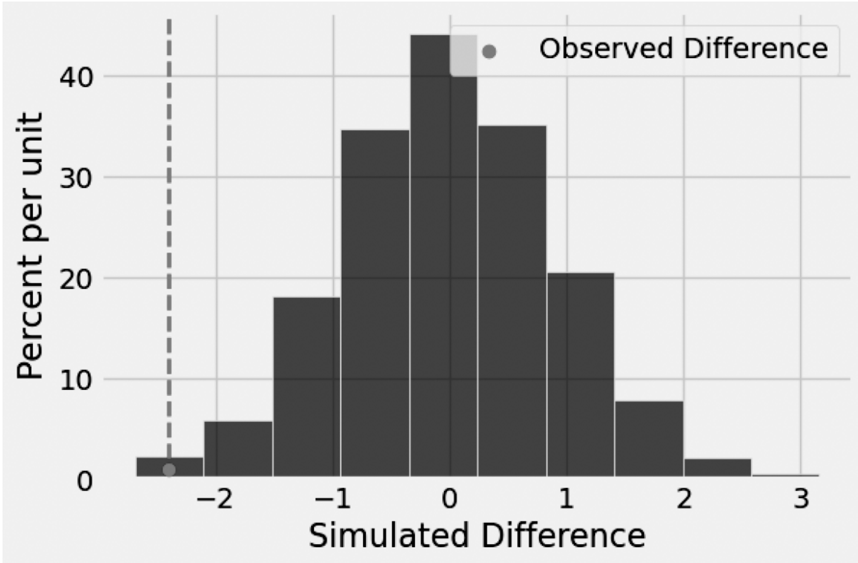
(c) (2 points) Suppose you want to test whether rock songs have lower danceability ratings than pop songs, on average. Which of the following is the best alternate hypothesis?

- √ **In the population of all rock and pop songs on Spotify, rock songs have lower danceability ratings than pop songs, on average.**
- ◯ In the sample, rock songs have lower danceability ratings than pop songs, on average.
- ◯ In the population of all rock and pop songs on Spotify, danceability ratings for both pop and rock songs are drawn from a uniform distribution between 0 and 100.
- ◯ In the sample, the distribution of danceability ratings is the same for pop songs as for rock songs.
- ◯ In the population of all rock and pop songs on Spotify, the distribution of danceability ratings is the same for pop songs as for rock songs.

(d) (3 points) Suppose you decide to use the difference of means between each group as your test statistic, defined as:

```
ave_danceability_rating_for_rock_songs - ave_danceability_rating_for_pop_songs
```

You first calculate your test statistic on your sample and save this as the `obs_stat` variable. Then, you simulate under the null hypothesis 10,000 times and record your simulated test statistics in an array called `sim_stats`. You plot both simulated and observed test statistics, as shown below:



Write a single line of code that evaluates to the empirical p-value of your test.

Sample Solution:

```
np.count_nonzero(sim_stats <= obs_stat) / len(sim_stats)
```

(e) (3 points) Suppose you calculate your empirical p-value to be 0.003. Using a 1% p-value cutoff, which of the following are valid conclusions you can make about your test? Select all that apply.

☐ The data are consistent with the null hypothesis.

√ **The data are consistent with the alternative hypothesis.**

☐ There is a 0.3% chance that the null hypothesis is true.

√ **If the null were true, there is a 1% chance that the null hypothesis would be incorrectly rejected.**

☐ None of these.

12. (2 points) When shuffling labels for a permutation test, sampling must be done without replacement.

√ **True** ◯ False

13. (2 points) Each person in a random sample of 1000 U.S. adults was asked if they agreed with the statement, "News organizations are growing in influence." Among the sampled men, 39% agreed. Among the sampled women, 43% agreed.

Data scientists have used an A/B test to see whether or not the observed difference is due to chance. The null hypothesis is one of the statements below. Pick the right one.

◯ In the sample, the percent of women who agree is the same as the percent of men who agree. The observed difference is due to chance.

◯ In the U.S., 39% of the men agree and 43% of the women agree, due to chance.

√ **In the U.S., the percent of men who agree is the same as the percent of women who agree. The difference in the sample is due to chance.**

◯ In the U.S., the percent of women who agree is different from the percent of men who agree, due to chance.

Table Reference

The table `building_data` contains 9 columns. The values in the columns `parcel_s`, `building_name`, `building_address`, `property_type`, and `energy_audit_due_date` have a `str` data type. The values in the rest of the columns `int` or `float` data types.

parcel_s	building_name	building_address	postal_code	floor_area	property_type	year_built	energy_audit_due_date	energy_use_2021
0010/001	2801 Leavenworth Street	2801 LEAVENWORTH ST	94109	133675	Commercial	1907	2024-04-01T00:00:00.000	6.21001e+06
0010/002	Argonaut Hotel-SV	495 JEFFERSON ST	94109	180000	Commercial	1907	2025-04-01T00:00:00.000	7.34107e+06
0011/008	Anchorage Garage	500 BEACH ST	94133	198525	Commercial	1974	2024-04-01T00:00:00.000	1.88699e+06

... (590 rows omitted)

The `zip_codes` table contains 4 columns. All the values in this table are either `float` or `int` data type.

zip	latitude	longitude	irs_estimated_population
94102	37.78	-122.42	21610
94103	37.77	-122.41	22940
94104	37.79	-122.4	1720

... (48 rows omitted)

At some point, you are asked to create the table `building_data_geo`. It should look like:

postal_code	parcel_s	building_name	building_address	floor_area	property_type	year_built	energy_audit_due_date	energy_use_2021	latitude	longitude	irs_estimated_population
94102	0296/001	449 Powell Street	449 POWELL ST	34173	Commercial	1913	2024-04-01T00:00:00.000	2.08193e+06	37.78	-122.42	21610
94102	0296/005	Chancellor Hotel	433 POWELL ST	46800	Commercial	1914	2021-04-01T00:00:00.000	3.01398e+06	37.78	-122.42	21610
94102	0296/006	400 POST ST	400 POST ST	61807	Commercial	1909	2020-04-01T00:00:00.000	9.32405e+06	37.78	-122.42	21610

... (590 rows omitted)

The `songs` table is shown below:

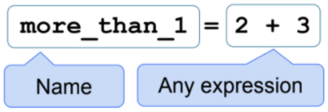
Title	Genre	Danceability
We Will Rock You	rock	42.4
Uptown Funk	pop	88.5

. . . (1994 rows omitted)

The table has 3 columns:

- `Title`: (string) the name of the song
- `Genre`: (string) the genre of the song which is either pop or rock
- `Danceability`: (float) a danceability rating between 0 and 100 (higher means more danceable)

Statements



- Statements don't have a value; they perform an action.
- An assignment statement changes the meaning of the name to the left of the = symbol.
- The name is bound to the value of the right hand side

Comparisons

- < and > mean what you expect (less than, greater than)
- <= means "less than or equal; likewise for >=
- == means "equal to"; != means "not equal to"
- Comparing strings compares their alphabetical order

Arrays: sequences of the same type that can be manipulated

- Arithmetic and comparisons are applied to each element individually
  - `make_array(1, 2, 3) > 2 # array([False, False, True])`
- Elementwise operations can be done on arrays of the same size
  - `make_array(1, 2)* make_array(2, 3) # array([2, 6])`

Defining a Function

```
def function(arg1, arg2, ...):
    # Body can contain any code
```

Note: Many functions `return` a value

for Statements

```
for i in np.arange(12):
    total = total + i
```

- The body is executed **for** every item in a sequence
- The body of the statement can have multiple lines
- The body should do something: assign, sample, etc

Conditional Statements

```
if <if_expression>:
    <if_body>
elif <elif_expression_0>:
    <elif_body_0>
elif <elif_expression_1>:
    <elif_body_1>
...
else:
    <else_body>
```

Simulating a Statistic

- Define a function to simulate one value of the statistic
- Create an empty collection array
- For each repetition of the process:
  - Call the function to simulate one value
  - Append this value to the collection array
- At the end, all simulated values will be in the collection array

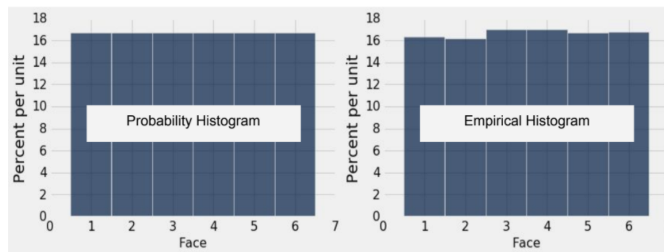
Total Variation Distance between two categorical distributions

- For each category, find the difference between the proportions in the two distributions
- Take the absolute value of the differences
- Sum all the absolute values, then divide by 2

A **histogram** has three defining properties:

- Bins are contiguous (though some may be empty) and drawn to scale.
- The area of each bar is the percent of entries in the bin.
- The total area of the histogram is 100%.

- The histogram on the left displays the theoretical probabilities of the number of spots on one roll of a fair die.
- The histogram on the right represents the empirical (or observed) distribution of the numbers of spots on many rolls of a fair die.
- Example of the *Law of Averages*: The more we roll, the more the histogram on the right is likely to resemble the one on the left.



Finding Probabilities

Complement Rule:
P(event happens) = 1 – P(event does not happen)

Multiplication Rule:
P(two events both happen) = P(first event happens) * P(second event happens given that the first event happened)

Addition Rule: If an event can happen in only one of two ways, then:
P(event happens) = P(happens in the first way) + P(happens in the second way)

p-values

- The *observed significance level* (or *p-value*) of a test is the chance, calculated under the null hypothesis, that the test statistic is equal to the one observed in the sample or more in the direction of the alternative.
- The result of a test of hypotheses is called *statistically significant* if the p-value is less than 5%; *highly statistically significant* if the p-value is less than 1%.

Table Properties and Methods

In the examples in the left column, np refers to the NumPy module, as usual. Everything else is a function, a method, an example of an argument to a function or method, or an example of an object we might call the method on. For example, tbl refers to a table, array refers to an array, and num refers to a number. array.item(0) is an example call for the method item, and in that example, array is the name previously given to some array.

Name	Description	Input	Output
`Table()`	Create an empty table, usually to extend with data	None	An empty `Table`
`tbl.with columns(name, values)` `tbl.with columns(n1, v1, n2, v2,...)`	A table with an additional or replaced column or columns. `name` is a string for the name of a column, `values` is an array	1. `string`: the name of the new column; 2. `array`: the values in that column	`Table`: a copy of the original Table with the new columns added
`tbl.column(column_name or index)`	The values of a column (an array)	`string` or `int`: the column_name or index	`array`: the values in that column
`tbl.num_rows`	Compute the number of rows in a table	None	`int`: the number of rows in the table
`tbl.num_columns`	Compute the number of columns in a table	None	`int`: the number of columns in the table
`tbl.labels`	Lists the column labels in a table	None	`array`: the names of each column (as strings) in the table
`tbl.select(col1, col2, ...)`	Create a copy of a table with only some of the columns.	`string` or `int`: column name(s) or index(es)	`Table` with the selected columns
`tbl.drop(col1, col2, ...)`	Create a copy of a table without some of the columns.	`string` or `int`: column name(s) or index(es)	`Table` without the selected columns
`tbl.relabeled(old+label, new_label)`	Creates a new table, changing the column name specified by the old label to the new label, and leaves the original table unchanged.	1. `string`: the old column name 2. `string`: the new column name	`Table`: a new table
`tbl.show(n)`	Display n rows of a table. If no argument is specified, defaults to displaying the entire table.	(Optional) `int`: number of rows you want to display	None: Displays a table with n rows
`tbl.sort(column_name or index)`	Create a copy of a table sorted by the values in a column. Defaults to ascending order unless `descending=True` is included.	1. `string` or `int`t: column index or name 2. (Optional) `boolean`: `descending=True`	`Table`: a copy of the original table with the column sorted
`tbl.where(column, predicate)`	Create a copy of a table with only the rows that match some predicate See `Table.where` predicates table.	1. `string` or `int`: column index or name 2. `are.(...)` predicate	`Table`: a copy of the original table with only the rows that match the predicate
`tbl.take(row_indices)`	A table with only the rows at the given indices. `row_indices` is either an array of indices or an integer corresponding to one index.	`array` of ints: the indices of the rows to be included in the Table OR `int`: the index of the row to be included	`Table`: a copy of the original table with only the rows at the given indices
`tbl.scatter(x_column, y_column)`	Draws a scatter plot consisting of one point for each row of the table. Note that `x_column` and `y_column` must be strings specifying column names.	1. `string` or `int`: name or index of the column on x-axis 2. `string` or `int`: name or index of the column on y-axis 3. (Optional) `fit line=True`	None: Draws a scatter plot
`tbl.plot(x_column, y_column)` `tbl.plot(x_column)`	Draws a line graph consisting of one point for each row of the table. If you only specify one column, it will plot the rest of the columns on the y-axis as different colored lines.	1. `string` or `int`: name or index of the column on x-axis 2. `string` or `int`: name or index of the column on y-axis	None: Draws a line graph
`tbl.barh(categories)` `tbl.barh(categories, values)`	Displays a bar chart with bars for each category in a column, with height proportional to the corresponding frequency. `values` argument unnecessary if table has only a column of categories and a column of values.	1. `string` or `int`: name or index of the column with categories 2. (Optional) `string` or `int`: name or index of the column with values for corresponding categories	None: Draws a bar chart

`tbl.hist(column, unit, bins, group)`	Generates a histogram of the numerical values in a column. `unit`, `bins` and `group` are optional arguments, used to label the axes, specify the intervals (bins) and plot separate histograms per group, respectively	1. `string` or `int`: name or index of the column with categories 2. (Optional) `string`: units of x-axis 3. (Optional) `array`: of ints/floats denoting bin boundaries 4. (Optional) `str`: name of column to group by	None: Draws a histogram
`tbl.bin(column_name or index)` `tbl.bin(column_name or index, bins)`	Groups values into intervals, known as bins. Results in a two-column table that contains the number of rows in each bin. The first column lists the left endpoints of the bins, except in the last row.	1. `string` or `int`: column name(s) or index(es) 2. (Optional) `array` of ints/floats denoting bin boundaries or an `int` of the number of bins you want	`Table`: A new table
`tbl.apply(function)` `tbl.apply(function, col1, col2, ...)`	Returns an array of values resulting from applying a function to each item in a column.	1. `function`: function to apply to column 2. (Optional) `string` or `int`: name or index of the column to apply function to (if you have multiple columns, the respective column's values will be passed as the corresponding argument to the function), and if there is no argument, your function will be applied to every row (Row object) in tbl	`array`: contains an element for each value in the original column after applying the function to it.
`tbl.group(column_or_columns, collect)`	Group rows by unique values or combinations of values in a column(s). Multiple columns must be entered in array or list form. Other values aggregated by count (default) or optional argument `collect`.	1. `string/int` or `array` of strings/ints: column(s) on which to group 2. (Optional) `collect`: function to aggregate values in cells (defaults to count)	`Table`: A new table
`tbl.pivot(col1, col2, values, collect)` `tbl.pivot(col1, col2)`	A pivot table where each unique value in `col1` has its own column and each unique value in `col2` has its own row. Count or aggregate values from a third column, collect with some function. Default `values` and `collect` return counts in cells.	1. `string` or `int`: name or index of column whose unique values will make up columns of the pivot table 2. `string` or `int`: name or index of column whose unique values will make up rows of the pivot table 3. (Optional) `string` or `int`: name or index of column containing the values of cell 4. (Optional) function: how the values are collected; e.g. np.mean	`Table`: A new table
`tblA.join(colA, tblB, colB)` `tblA.join(colA, tblB)`	Generate a table with the columns of `tblA` and `tblB`, containing rows for all values of a column that appear in both tables. Default `colB` is `colA`. `colA` and `colB` must be strings specifying column names.	1. `string`: name of column in tblA with values to join on 2. Table: other Table 3. (Optional) `string`: if column names are different between Tables, the name of the shared column in tblB	`Table`: A new table
`tbl.sample(n)` `tbl.sample(n, with_replacement)`	A new table where n rows are randomly sampled from the original table; by default, `n=tbl.num_rows`. Default is with replacement. For sampling without replacement, use argument `with_replacement=False`. For a non-uniform sample, provide a third argument `weights=distribution` where `distribution` is an array or list containing the probability of each row.	1. `int`: sample size 2. (Optional) `with replacement=True`	`Table`: A new table with n rows
`tbl.row(row_index)`	Accesses the row of a table by taking the index of the row as its argument. Note that rows are in general not arrays, as their elements can be of different types. However, you can use `.item` to access a particular element of a row using `row.item(label)`.	`int`: row index	`Row` object with the values of the row and labels of the corresponding columns
`tbl.rows`	Can use to access all of the rows of a table.	None	`Row` object made up of all rows as individual row objects

Miscellaneous Functions

These are functions in the datascience library that are used in the course that don't fall into any of the categories above.

Name	Description	Input	Output
`sample_proportions(sample_size, model_proportions)`	sample size should be an integer, model proportions an array of probabilities that sum up to 1. The function samples sample size objects from the distribution specified by model proportions. It returns an array with the same size as model proportions. Each item in the array corresponds to the proportion of times it was sampled out of the sample size times.	1. int: sample size 2. array: an array of proportions that should sum to 1	array: each item corresponds to the proportion of times that corresponding item was sampled from model proportions in sample size draws. Should sum to 1.
`minimize(function)`	Returns an array of values such that if each value in the array was passed into function as arguments, it would minimize the output value of function.	function: name of a function that will be minimized	array: An array in which each element corresponds to an argument that minimizes the output of the function. Values in the array are listed based on the order they are passed into the function; the first element in the array is also going to be the first value passed into the function.

Array Functions and Methods

Function	Description
`max(array)`	Returns the maximum value of an array
`min(array)`	Returns the minimum value of an array
`sum(array), np.sum(array)`	Returns the sum of the values in an array
`abs(num), np.abs(array)`	Take the absolute value of a number or each number in an array
`np.round(num), np.round(array)`	Round number or array of numbers to the nearest integer
`len(array)`	Returns the length (number of elements) of an array
`make_array(val1, val2, ...)`	Makes a numpy array with the values passed in
`np.average(array), np.mean(array)`	Returns the mean value of an array
`np.std(array)`	Returns the standard deviation of an array
`np.diff(array)`	Returns a new array of size len(arr)-1 with elements equal to the difference between adjacent elements; val 2 - val 1, val 3 - val 2, etc.
`np.sqrt(array)`	Returns an array with the square root of each element

`np.arange(start, stop, step)`
`np.arange(start, stop)`
`np.arange(stop)`
	An array of numbers starting with start, going up in increments of step, and going up to but excluding stop. When start and/or step are left out, default values are used in their place. Default step is 1; default start is 0

`array.item(index)`	Returns the (index-1)-th item in an array (remember Python indices start at 0!)

`np.random.choice(array, n)`
`np.random.choice(array)`
`np.random.choice(array, n, replace)`
	Picks one (by default) or some number 'n' of items from an array at random. Default is with replacement. For sampling without replacement, use argument replace=False.

`np.count_nonzero(array)`	Returns the number of non-zero (or True) elements in an array
`np.append(array, item)`	Returns a copy of the input array with item (must be the same type as the other entries in in the array) appended to the end
`percentile(p, array)`	Returns the pth percentile of an array

Table.where Predicates

Any of these predicates can be negated by adding not in front of them, e.g. `are.not_equal_to(Z)` or `are.not_containing(S)`.

Function	Description
`are.equal_to(Z)`	Equal to Z
`are.above(x)`	Greater than x
`are.above_or_equal_to(x)`	Greater than or equal to x
`are.below(x)`	Less than x
`are.below_or_equal_to(x)`	Less than or equal to x
`are.between(x,y)`	Greater than or equal to x and less than y
`are.between_or_equal_to(x,y)`	Greater than or equal to x and less than or equal to y
`are.contained_in(A)`	Is a substring of A (if A is a string) or an element of A (if A is a list/array)
`are.containing(S)`	Contains the string S
`are.strictly_between(x,y)`	Greater than x and less than y