Homework 7: Hypothesis Testing and A/B testing

Please complete this notebook by filling in the cells provided. When you're done:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells.
2. Select `Download as HTML (.html)` from the `File` menu
3. Inspect your file to make sure it looks like it should.
4. Upload your file to Moodle.

Run the cell below to prepare the notebook.

In [1]:
```
# Run this cell to set up the notebook, but please don't change it.
import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)
```

Hypothesis Testing Review

Games with Chantel

Our friend Chantel comes over and asks us to play a game with them. The game works like this:

> We will flip a fair coin 10 times, and if the number of heads is greater than or equal to 5, we win!
>
> Otherwise, Chantel wins.

We play the game once and we lose, observing 3 heads. We think the coin was rigged against us! Chantel is adamant, however, that the coin was fair.

Question 1

Suppose you'd like to perform an hypothesis test in this scenario. State the null and alternative hypotheses you would test.

Null: The probability for each face(head, tail) of the coin to appear is 1/2. p1=p2

Alternative: Each face of the coin appears with different chances on each flip, which is an unfair coin.

Question 2

Define the function `coin_test_statistic`, which, given an array of any length containing the strings 'Heads' and 'Tails', returns an appropriate test statistic for this hypothesis test.

In [2]:
```
def coin_test_statistic(heads_and_tails_array):
    '''Calculates the difference between the observed proportion of heads ar
    num_heads = 0
    array_size = len(heads_and_tails_array)
    for i in np.arange(array_size):
        if heads_and_tails_array[i] == "Heads":
            num_heads = num_heads + 1
    observed_statistic = abs ((num_heads/array_size) - 0.5)
    return observed_statistic

# We recommend trying some examples here to ensure your code works as you ir
# Here is one small example:
coin_test_statistic(make_array('Tails', 'Tails', 'Tails', 'Heads'))
```

Out[2]:
```
0.25
```

We have created the function which simulates under the null hypothesis for you below. Examine the code and make sure you know exactly what it does.

In [3]:
```
# Run this code and know what it does.  It may be helpful to
# write a docstring that documents the function.
def simulate_coin_under_null():
    statistics = make_array()
    for i in range(10000):
        sample = np.random.choice(make_array('Heads', 'Tails'), 10)
        statistic = coin_test_statistic(sample)
        statistics = np.append(statistics, statistic)
    return statistics
simulated_coin_statistics = simulate_coin_under_null()
simulated_coin_statistics
```

Out[3]:
```
array([ 0. ,  0.2,  0.2, ...,  0. ,  0. ,  0. ])
```

Question 3

Find the p-value of our observation without using a histogram. Think about how you can use the variable `simulated_coin_statistics` defined in the code cell above.

In [4]:
```
coin_p_val = np.count_nonzero(simulated_coin_statistics >= 0.5)/10000
coin_p_val
```

Out[4]:
```
0.0015
```

Chantel asks us to play another game. This time, the game involves rolling a die. After playing 100 rounds of the game, we begin to suspect Chantel's die isn't fair. Maybe we shouldn't play games with Chantel anymore...

We would like to use hypothesis testing to decide whether or not Chantel's die is fair.

Question 4

State the null and alternate hypotheses for this inference problem.

Null: The probability for each face of the die to appear is 1/6. p1=p2=p3=p4=p5=p6

Alternative: Each face of the die appears with different chances on each roll, which is an unfair die.

The table `rolls` contains the 100 die-roll outcomes from our games with Chantel.

In [5]:
```
rolls = Table.read_table('/srv/data/DS_113_S23/HW/HW_7/rolls.csv')
rolls
```

Out[5]:

roll
3
5
2
5
6
3
1
3
2
4

... (90 rows omitted)

Question 5

Compute the proportion of each value for the die in the sample. Your output should be a table called `observed_proportions` with two columns:

- `value` : The value of the die (1 through 6).
- `proportion` : The proportion of rolls in `rolls` that have that value.

In [6]:
```
roll_array = make_array(0,0,0,0,0,0)
for i in np.arange(100):
    if (rolls["roll"].item(i) == 1):
        roll_array[0] = roll_array[0] +1
    if (rolls["roll"].item(i) == 2):
        roll_array[1] = roll_array[1] +1
    if (rolls["roll"].item(i) == 3):
        roll_array[2] = roll_array[2] +1
    if (rolls["roll"].item(i) == 4):
        roll_array[3] = roll_array[3] +1
    if (rolls["roll"].item(i) == 5):
        roll_array[4] = roll_array[4] +1
    if (rolls["roll"].item(i) == 6):
        roll_array[5] = roll_array[5] +1
proportion_array = roll_array/100

observed_proportions = Table().with_columns("value", make_array(1,2,3,4,5,6)
                                                        "proportion", proportion_array)
observed_proportions
```

Out[6]:

value	proportion
1	0.13
2	0.26
3	0.21
4	0.16
5	0.09
6	0.15

Question 6

Define a test statistic that you can use to determine whether or not the observed die is fair. The function should take as an argument an array of observed proportions, and it should return a single number representing the test statistic.

Hint: We want to compare the observed distribution of die values to the expected distribution of values.

In [7]:
```
def die_test_statistic(sample):
    die = np.arange(1, 6+1, 1)
    avg_roll = sum(sample * die)
    avg_for_fair_die = np.mean(die)
    return abs(avg_roll - avg_for_fair_die)

observed_test_statistic = die_test_statistic(observed_proportions.column('pr
observed_test_statistic
```

Out[7]:
```
0.22999999999999998
```

The function below simulates making 100 rolls under the null hypothesis (that the die is fair).

It does this 5000 times and calculates the test statistic each time.

(It will take a bit to run.)

In [8]:
```
# Run this code and know what it does.
def simulate_die_under_null():
    statistics = make_array()
    for i in range(5000):
        num_rolls = 100
        rolls = Table().with_column("Face", np.random.choice(np.arange(1, 6+
        proportions = rolls.group("Face").sort("Face").column("count") / num
        statistic = die_test_statistic(proportions)
        statistics = np.append(statistics, statistic)
    return statistics
simulated_die_statistics = simulate_die_under_null()
simulated_die_statistics
```

Out[8]:
```
array([ 0.12,  0.08,  0.02, ...,  0.08,  0.02,  0.08])
```

Question 7

Find the p-value of our observation without using a histogram. Think about how you can use the array `simulated_die_statistics` defined in the code cell above.

Hint: Our observed test statistic is represented by the variable called `observed_test_statistic`.

```
In [9]: die_p_val = np.count_nonzero(simulated_die_statistics >= observed_test_stati
        die_p_val
```

```
Out[9]: 0.1864
```

Question 8

What do you conclude from your analysis?

Since the probability is large, we accept the null hypothesis. The probability to get each faces are based on randomness, which means that the die is fair.

5. Do Diet Drinks Cause Weight Gain?

Betteridge's Law notwithstanding, this is a serious question and a subject of much recent research. Though artificially-sweetened diet drinks (like Diet Pepsi or a cup of coffee with sucralose) contain no calories, it is theorized that drinking sweet diet drinks could increase cravings for other sweet food, or that the artificial sweeteners in diet drinks (like aspartame and sucralose) could directly cause weight gain. This article summarizes some of the recent research activity.

In this exercise we'll use bootstrap confidence intervals to replicate some of the analysis in this study. For simplicity (and because we couldn't get our hands on the data), we'll work with a synthetic dataset, not the dataset used in the actual study.

The original dataset is called the San Antonio Heart Study. It tracks 3,371 people living in San Antonio, Texas, over 7-8 years. For each person, it records (among many other things) how many diet drinks they reported drinking in a typical week, and the change in the person's Body Mass Index (BMI, a measure of weight adjusted for height) between the start and the end of the 7-8 year period. A change of 1 in BMI means that the person gained around 4-8 pounds, depending on their height.

```
In [10]: diet = Table.read_table("/srv/data/DS_113_S23/HW/HW_7/diet.csv")
         diet
```

Out[10]:

ID	Typical diet drinks per week	BMI change
0	6	-4.40065
1	0	0.952995
2	4	2.71019
3	6	-0.276764
4	0	6.12079
5	3	-0.158611
6	0	-2.94134
7	0	-2.64784
8	0	7.0943
9	7	1.69633

... (3361 rows omitted)

Question 1

We will crudely divide people into two categories: those who consume any diet drinks, and those who consume none. Create a table called `drink_or_not` that's a copy of `diet`, with an extra column called `"Drink"`. It should contain the value `True` for people who drank at least one drink per week and `False` otherwise.

In [11]:
```
drink_or_not = diet.with_column("Drink", diet[1] >= 1)
drink_or_not
```

Out[11]:

ID	Typical diet drinks per week	BMI change	Drink
0	6	-4.40065	True
1	0	0.952995	False
2	4	2.71019	True
3	6	-0.276764	True
4	0	6.12079	False
5	3	-0.158611	True
6	0	-2.94134	False
7	0	-2.64784	False
8	0	7.0943	False
9	7	1.69633	True

... (3361 rows omitted)

Question 2

Compute a table called `means` that looks like this, but with the `"BMI change mean"` column filled in according to its name:

|Drink|BMI change mean| |-|-| |False|?| |True|?|

In [12]:
```
means = drink_or_not.group("Drink", np.average).drop(1,2).relabel("BMI chang
means
```

Out[12]:

Drink	BMI change mean
False	1.01925
True	1.50449

You should find that diet drinkers have a higher average BMI change - they gained more weight on average. (The average for both groups is positive because most people gain a little weight as they get older.)

Question 3

Suppose our `diet` table is a random sample from the population of all people who lived during this 7-8 year period. We want to know whether drinking diet drinks really makes a difference in BMI change. Formulate appropriate null and alternative hypotheses for an hypothesis test, **or** (if appropriate) explain why no hypothesis test is needed.

Null hypothesis: There is no difference in BMI change mean between people who are drinking diet drinks and who are not. p1=p2

Alternative hypothesis: There is a difference in BMI change mean.

- if the null hypothesis is true, we can randomly suffle the people without considering they drink or not.
- to simulate the null, shuffle all the values and randomly give the number of True and number of False.

Question 4

Use a permutation test with 5000 repetitions to test your hypothesis. Report a p-value and the conclusion you reach based on the p-value.

Note: There are not equal numbers of participants in each category.

In [15]:
```
## make a function that returns the difference in average
def difference_in_average(table):
    averages = table.select('Drink', 'BMI change').group('Drink', np.average
    return averages.item(1) - averages.item(0)
```

In [18]:
```
## get the difference in average (before shuffling)
observed = difference_in_average(drink_or_not)
observed
```

Out[18]:
```
0.4852397109235964
```

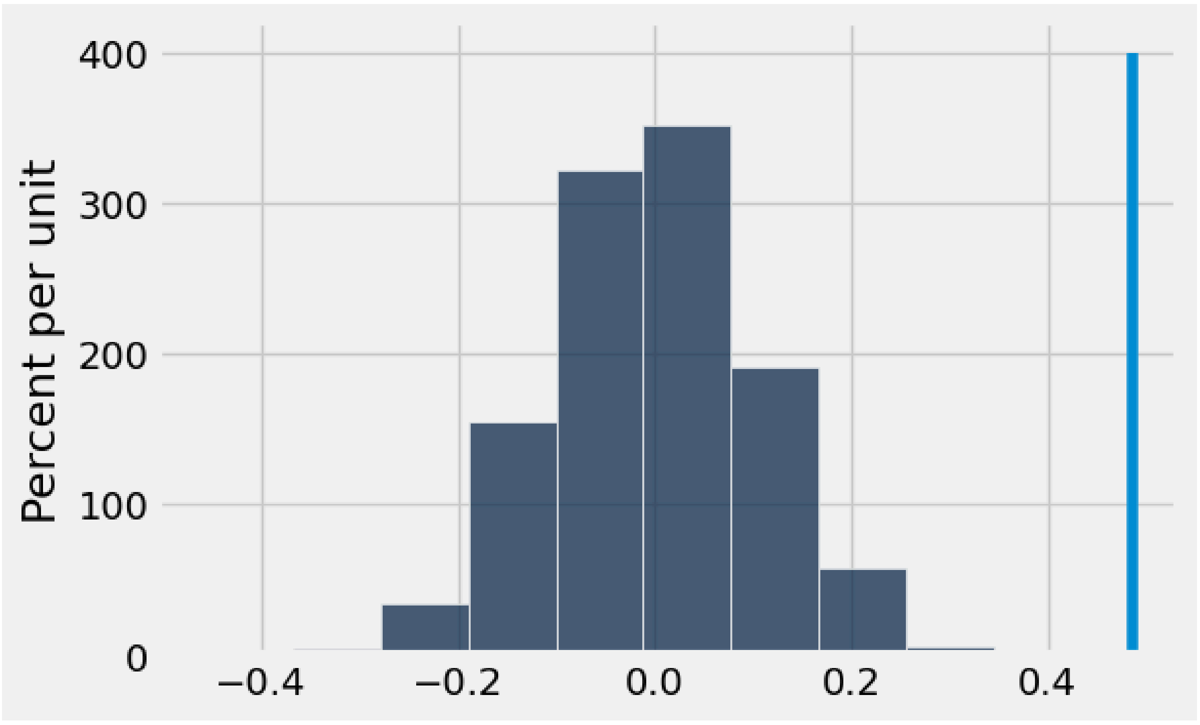
In [23]:
```
sampled_stats = make_array()

# repeat 5000 times
for i in np.arange(5000):
    shuffled = drink_or_not.select('BMI change').sample(len(drink_or_not[0])
    shuffled_table = drink_or_not.select('Drink').with_column('BMI change',
    sampled_stats = np.append(sampled_stats, difference_in_average(shuffled_

sampled_stats
```

Out[23]:
```
array([-0.09376613,  0.09233781,  0.17794224, ...,  0.11238606,
       -0.08387965,  0.01534192])
```

In [25]:
```
# calculate the p-value
p_value = np.count_nonzero(sampled_stats >= observed)/len(sampled_stats)
p_value
```

Out[25]:
```
0.0
```

In [27]:
```
Table().with_column("", sampled_stats).hist()
_ = plt.plot([observed, observed], [0, 4])
```



The empirical p-value is 0.0, meaning that no 5,000 permuted samples resulted in a difference of 0.485 or above. This is only an approximation. According to our simulation, we do have evidence to reject the null hypothesis.

Question 5

Now we are going to use the bootstrap to estimate the range of plausible values for the true difference in BMI change based on our sample. To begin, please write a function that does the following:

1. Takes a random sample **with replacement** from **within** each group that is the same size as the original group.
2. Calcuulates the difference in average BMI change between the two samples.

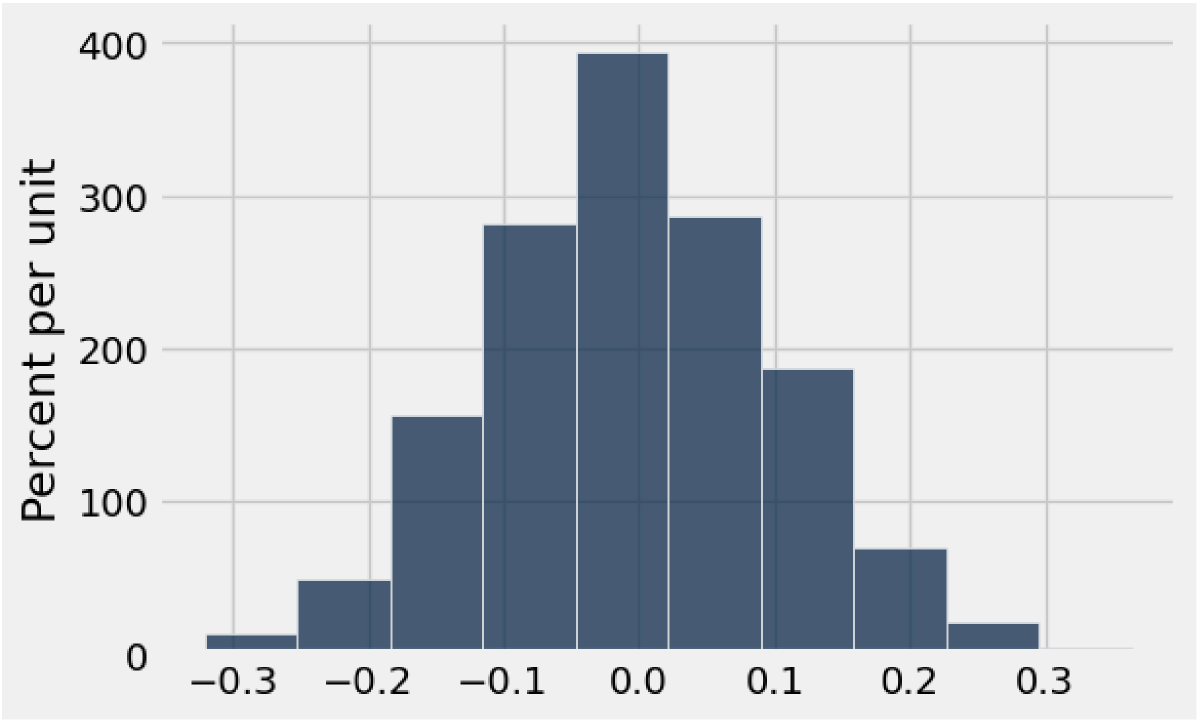
In [31]:
```
def rand_diff():
    '''take a random sample with replacement and return the difference in av
    random_suffle = drink_or_not.select('BMI change').sample(len(drink_or_nc
    random_sample = drink_or_not.select('Drink').with_column('BMI change', r
    difference = difference_in_average(random_sample)
    return difference

rand_diff()
```

Out[31]:
```
-0.01026373529180269
```

Question 6

Use your function to produce 1000 bootstrapped samples and calculate the difference in means for each sample. Produce a histogram of these differences.

In [33]:
```
sampled_stats = make_array()

# repeat 1000 times
for i in np.arange(1000):
    sample_diff = rand_diff()
    sampled_stats = np.append(sampled_stats, sample_diff)

Table().with_column("", sampled_stats).hist()
```

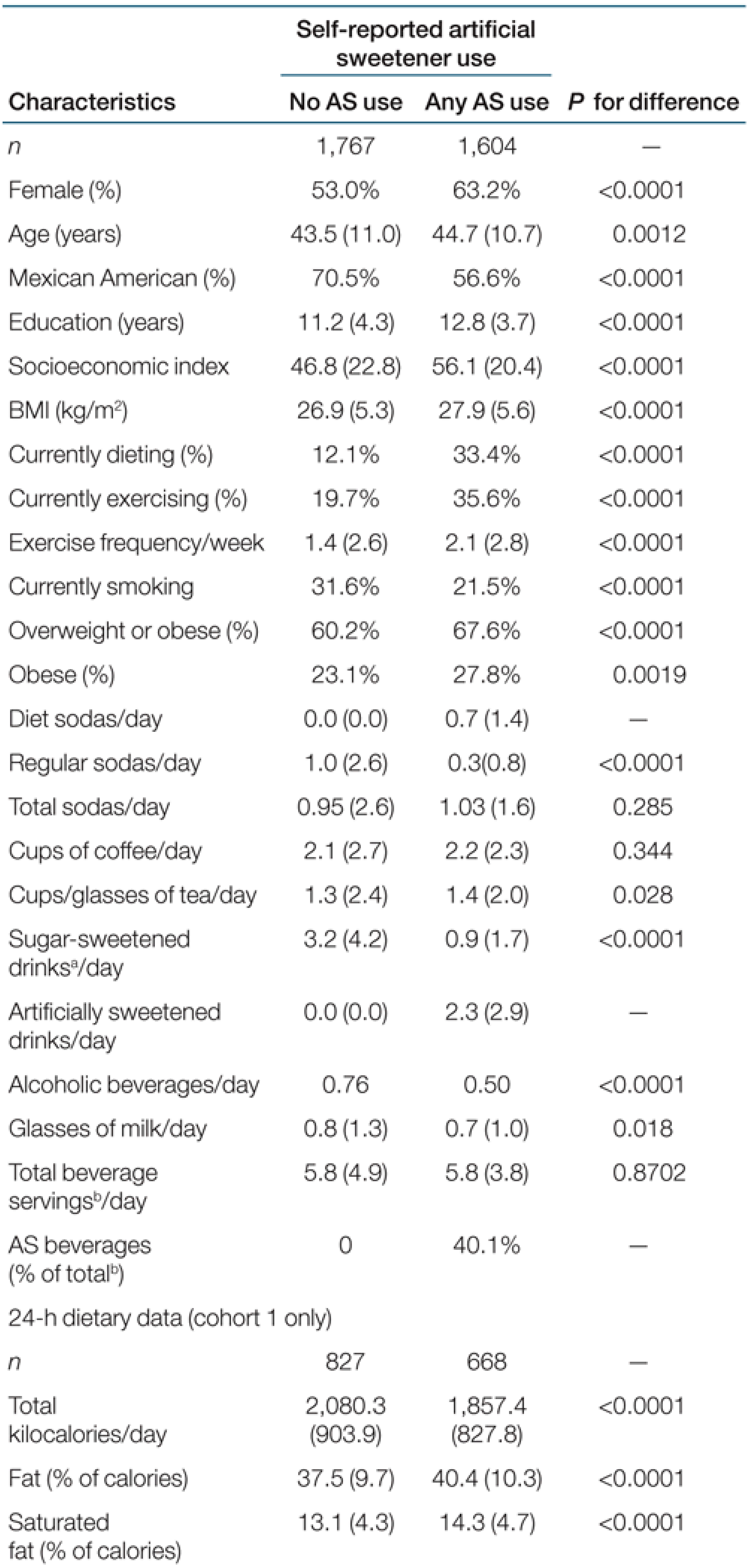


Question 7

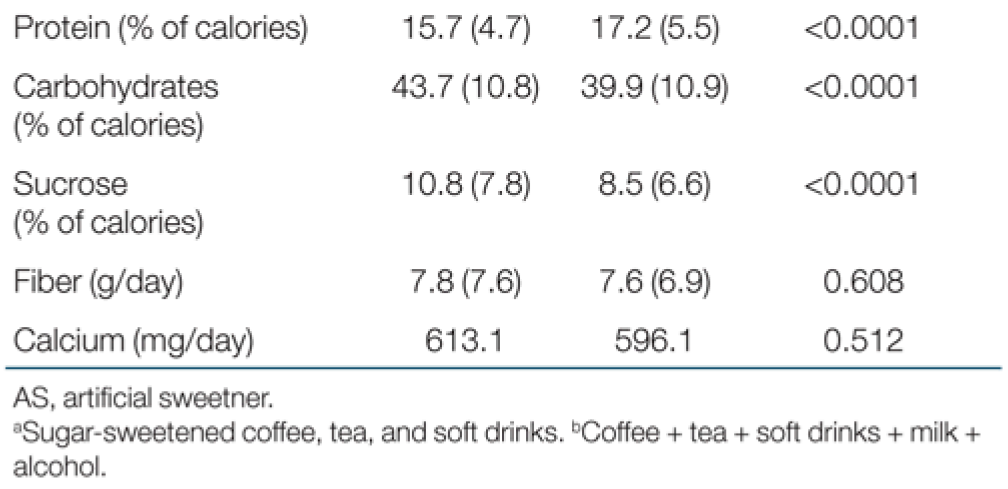
Based on your histogram, report a plausible interval for the true value of the difference in BMI change.

Answer: The true difference in BMI change between those who drink artifically sweet drinks and those that do not is between *-0.32* and *0.29*.

The study tracked many pieces of information about each individual. The authors include the following table in their report, comparing diet-drinkers and non-diet-drinkers on various traits. The traits were measured at the *start* of the 7-8 year observational period.



	Self-reported artificial sweetener use		
Characteristics	No AS use	Any AS use	*P* for difference
n	1,767	1,604	—
Female (%)	53.0%	63.2%	<0.0001
Age (years)	43.5 (11.0)	44.7 (10.7)	0.0012
Mexican American (%)	70.5%	56.6%	<0.0001
Education (years)	11.2 (4.3)	12.8 (3.7)	<0.0001
Socioeconomic index	46.8 (22.8)	56.1 (20.4)	<0.0001
BMI (kg/m^2)	26.9 (5.3)	27.9 (5.6)	<0.0001
Currently dieting (%)	12.1%	33.4%	<0.0001
Currently exercising (%)	19.7%	35.6%	<0.0001
Exercise frequency/week	1.4 (2.6)	2.1 (2.8)	<0.0001
Currently smoking	31.6%	21.5%	<0.0001
Overweight or obese (%)	60.2%	67.6%	<0.0001
Obese (%)	23.1%	27.8%	0.0019
Diet sodas/day	0.0 (0.0)	0.7 (1.4)	—
Regular sodas/day	1.0 (2.6)	0.3(0.8)	<0.0001
Total sodas/day	0.95 (2.6)	1.03 (1.6)	0.285
Cups of coffee/day	2.1 (2.7)	2.2 (2.3)	0.344
Cups/glasses of tea/day	1.3 (2.4)	1.4 (2.0)	0.028
Sugar-sweetened drinks[a]/day	3.2 (4.2)	0.9 (1.7)	<0.0001
Artificially sweetened drinks/day	0.0 (0.0)	2.3 (2.9)	—
Alcoholic beverages/day	0.76	0.50	<0.0001
Glasses of milk/day	0.8 (1.3)	0.7 (1.0)	0.018
Total beverage servings[b]/day	5.8 (4.9)	5.8 (3.8)	0.8702
AS beverages (% of total[b])	0	40.1%	—
24-h dietary data (cohort 1 only)			
n	827	668	—
Total kilocalories/day	2,080.3 (903.9)	1,857.4 (827.8)	<0.0001
Fat (% of calories)	37.5 (9.7)	40.4 (10.3)	<0.0001
Saturated fat (% of calories)	13.1 (4.3)	14.3 (4.7)	<0.0001



Protein (% of calories)	15.7 (4.7)	17.2 (5.5)	<0.0001
Carbohydrates (% of calories)	43.7 (10.8)	39.9 (10.9)	<0.0001
Sucrose (% of calories)	10.8 (7.8)	8.5 (6.6)	<0.0001
Fiber (g/day)	7.8 (7.6)	7.6 (6.9)	0.608
Calcium (mg/day)	613.1	596.1	0.512

AS, artificial sweetner.
[a]Sugar-sweetened coffee, tea, and soft drinks. [b]Coffee + tea + soft drinks + milk + alcohol.

Question 8

Using this table, Steve the Scientist makes the following argument:

> "People who drank diet drinks were much more likely (12.1% versus 33.4%) to say they were dieting at the start of the observational period. So perhaps drinking diet drinks does not directly cause weight gain. Instead, the association we observed in question 5 could be caused entirely by this confounding factor."

Is this a valid argument?

Our null hypothesis was that there is no difference in the average BMI changes, which means change in weight, for people who drinks diet drinks and who does not. Since the p-value for the difference in mean is less than 0.0001, we have an evidence to reject the null hypothesis. Therefore, we can argue that there is a chance that drinking diet drinks would have an effect on weight gain.