lab07_master

November 1, 2019

1 Lab 7: Crime and Penalty

Welcome to Lab 7!

[ ]:
```
# Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
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)

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('lab07.ok')
#_ = ok.auth(inline=True)
```

1.1 1. A/B Testing

A/B testing is a form of hypothesis testing that allows you to make comparisons between two distributions.

You'll almost never be explicitly asked to perform an A/B test. Make sure you can identify situations where the test is appropriate and know how to correctly implement each step.

Question 1.1: The following statements are the unordered steps of an A/B hypothesis test:

1. Choose a test statistic (typically the difference in means between two categories)
2. Shuffle the labels of the original sample, find your simulated test statistic, and repeat many times
3. Find the value of the observed test statistic

4. Calculate the p-value based off your observed and simulated test statistics
5. Define a null and alternate model
6. Use the p-value and p-value cutoff to draw a conclusion about the null hypothesis

Make an array called `ab_test_order` that contains the correct order of an A/B test, where the first item of the array is the first step of an A/B test and the last item of the array is the last step of an A/B test

```
BEGIN QUESTION
name: q1_1
```

[2]:
```
ab_test_order = make_array(5, 1, 3, 2, 4, 6) # SOLUTION
```

Question 1.2: If the null hypothesis of an A/B test is correct, should the order of labels affect the differences in means between each group? Why do we shuffle labels in an A/B test?

```
BEGIN QUESTION
name: q1_2
```

SOLUTION: Under the null model, there should be no statistically significant difference between the grouped means. We shuffle labels as a way to resample the original data, and understand the baseline differences between groups in the data.

1.2 2: Murder Rates

Punishment for crime has many philosophical justifications. An important one is that fear of punishment may *deter* people from committing crimes.

In the United States, some jurisdictions execute people who are convicted of particularly serious crimes, such as murder. This punishment is called the *death penalty* or *capital punishment*. The death penalty is controversial, and deterrence has been one focal point of the debate. There are other reasons to support or oppose the death penalty, but in this project we'll focus on deterrence.

The key question about deterrence is:

> Through our exploration, does instituting a death penalty for murder actually reduce the number of murders?

You might have a strong intuition in one direction, but the evidence turns out to be surprisingly complex. Different sides have variously argued that the death penalty has no deterrent effect and that each execution prevents 8 murders, all using statistical arguments! We'll try to come to our own conclusion.

The data The main data source for this lab comes from a paper by three researchers, Dezhbakhsh, Rubin, and Shepherd. The dataset contains rates of various violent crimes for every year 1960-2003 (44 years) in every US state. The researchers compiled the data from the FBI's Uniform Crime Reports.

Since crimes are committed by people, not states, we need to account for the number of people in each state when we're looking at state-level data. Murder rates are calculated as follows:

$$\text{murder rate for state X in year Y} = \frac{\text{number of murders in state X in year Y}}{\text{population in state X in year Y}} * 100000$$

(Murder is rare, so we multiply by 100,000 just to avoid dealing with tiny numbers.)

[4]:
```
murder_rates = Table.read_table('crime_rates.csv').select('State', 'Year',␣
↪'Population', 'Murder Rate')
murder_rates.set_format("Population", NumberFormatter)
```

[4]:
```
State  | Year | Population | Murder Rate
Alaska | 1960 | 226,167    | 10.2
Alaska | 1961 | 234,000    | 11.5
Alaska | 1962 | 246,000    | 4.5
Alaska | 1963 | 248,000    | 6.5
Alaska | 1964 | 250,000    | 10.4
Alaska | 1965 | 253,000    | 6.3
Alaska | 1966 | 272,000    | 12.9
Alaska | 1967 | 272,000    | 9.6
Alaska | 1968 | 277,000    | 10.5
Alaska | 1969 | 282,000    | 10.6
… (2190 rows omitted)
```

Murder rates vary over time, and different states exhibit different trends. The rates in some states change dramatically from year to year, while others are quite stable. Let's plot a couple, just to see the variety.

Question 2.1. Draw a line plot with years on the horizontal axis and murder rates on the vertical axis. Include two lines: one for Alaska murder rates and one for Minnesota murder rates. Create this plot using a single call, `ak_mn.plot('Year')`.

Hint: To create two lines, you will need create the table `ak_mn` with two columns of murder rates, in addition to a column of years. This table will have the following structure:

Year	Murder rate in Alaska	Murder rate in Minnesota
1960	10.2	1.2
1961	11.5	1
1962	4.5	0.9

… (41 rows omitted)

```
BEGIN QUESTION
name: q2_1
```

[5]:
```
# The next lines are provided for you.  They create a table
# containing only the Alaska information and one containing
# only the Minnesota information.
ak = murder_rates.where('State', 'Alaska').drop('State', 'Population').
↪relabeled(1, 'Murder rate in Alaska')
```

```
mn = murder_rates.where('State', 'Minnesota').drop('State', 'Population').
↪relabeled(1, 'Murder rate in Minnesota')

# Fill in this line to make a table like the one pictured above.
ak_mn = ak.join('Year', mn) # SOLUTION
ak_mn
```

[5]:

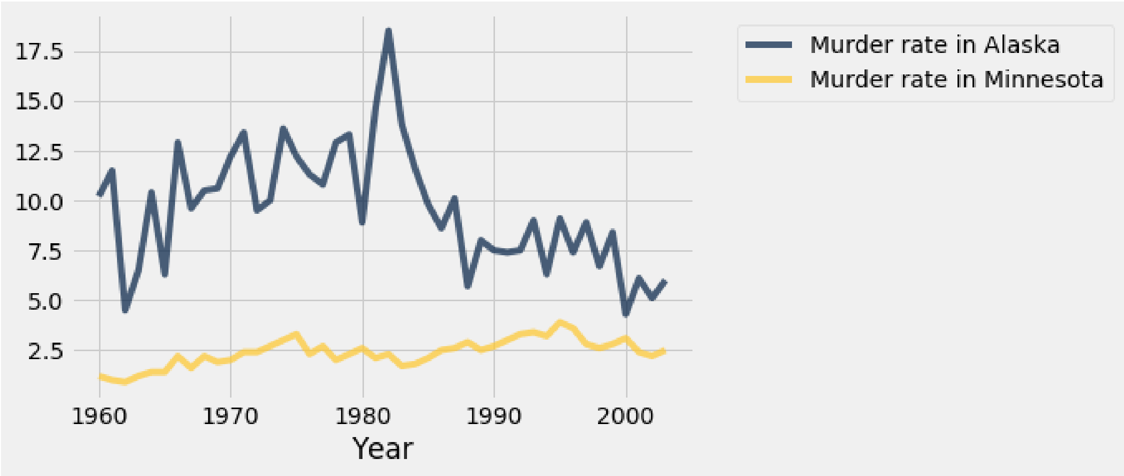
```
Year | Murder rate in Alaska | Murder rate in Minnesota
1960 | 10.2                  | 1.2
1961 | 11.5                  | 1
1962 | 4.5                   | 0.9
1963 | 6.5                   | 1.2
1964 | 10.4                  | 1.4
1965 | 6.3                   | 1.4
1966 | 12.9                  | 2.2
1967 | 9.6                   | 1.6
1968 | 10.5                  | 2.2
1969 | 10.6                  | 1.9
… (34 rows omitted)
```

Question 2.2: Using the table `ak_mn`, draw a line plot that compares the murder rate in Alaska and the murder rate in Minnesota over time.

```
BEGIN QUESTION
name: q2_2
```

[9]:

```
# Draw your line plot here
ak_mn.plot('Year') # SOLUTION
```



Now what about the murder rates of other states? Say, for example, California and New York? Run the cell below to plot the murder rates of different pairs of states.

```
[10]: # Compare the murder rates of any two states by filling in the blanks below

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

def state(state1, state2):
    state1_table = murder_rates.where('State', state1).drop('State',␣
 ↪'Population').relabeled(1, 'Murder rate in {}'.format(state1))
    state2_table = murder_rates.where('State', state2).drop('State',␣
 ↪'Population').relabeled(1, 'Murder rate in {}'.format(state2))
    s1_s2 = state1_table.join('Year', state2_table)
    s1_s2.plot('Year')
    plt.show()

states_array = murder_rates.group('State').column('State')

_ = interact(state,
             state1=widgets.
 ↪Dropdown(options=list(states_array),value='California'),
             state2=widgets.Dropdown(options=list(states_array),value='New␣
 ↪York')
            )
```

```
interactive(children=(Dropdown(description='state1', index=4, options=('Alabama', 'Alaska', 'A
```

1.3 3. The Death Penalty

Some US states have the death penalty, and others don't, and laws have changed over time. In addition to changes in murder rates, we will also consider whether the death penalty was in force in each state and each year.

Using this information, we would like to investigate how the presence of the death penalty affects the murder rate of a state.

Question 3.1. We want to know whether the death penalty *causes* a change in the murder rate. Why is it not sufficient to compare murder rates in places and times when the death penalty was in force with places and times when it wasn't?

```
BEGIN QUESTION
name: q3_1
```

SOLUTION: We didn't run a randomized controlled experiment, so we may be misled by confounding factors or reverse causation. For example, perhaps more traditionalist or conservative cultures are more likely to have lower murder rates and more likely to have the death penalty (a confounding factor); or perhaps higher murder rates lead politicians to institute the death penalty (reverse causation).

1.3.1 A Natural Experiment

In order to attempt to investigate the causal relationship between the death penalty and murder rates, we're going to take advantage of a *natural experiment*. A natural experiment happens when something other than experimental design applies a treatment to one group and not to another (control) group, and we have some hope that the treatment and control groups don't have any other systematic differences.

Our natural experiment is this: in 1972, a Supreme Court decision called *Furman v. Georgia* banned the death penalty throughout the US. Suddenly, many states went from having the death penalty to not having the death penalty.

As a first step, let's see how murder rates changed before and after the court decision. We'll define the test as follows:

> **Population:** All the states that had the death penalty before the 1972 abolition. (There is no control group for the states that already lacked the death penalty in 1972, so we must omit them.) This includes all US states **except** Alaska, Hawaii, Maine, Michigan, Wisconsin, and Minnesota.
>
> **Treatment group:** The states in that population, in 1973 (the year after 1972).
>
> **Control group:** The states in that population, in 1971 (the year before 1972).
>
> **Null hypothesis:** Murder rates in 1971 and 1973 come from the same distribution.
>
> **Alternative hypothesis:** Murder rates were higher in 1973 than they were in 1971.

Our alternative hypothesis is related to our suspicion that murder rates increase when the death penalty is eliminated.

Question 3.2: Should we use an A/B test to test these hypotheses? If yes, what is our "A" group and what is our "B" group?

BEGIN QUESTION
name: q3_2

SOLUTION: Yes, we should be using an A/B test. The "A" group is the "control" group - this consists of states with the death penalty in 1971. The "B" group is the "treatment" group - this consists of states after the death penalty was abolished in 1973.

The `death_penalty` table below describes whether each state allowed the death penalty in 1971.

[11]:
```
non_death_penalty_states = make_array('Alaska', 'Hawaii', 'Maine', 'Michigan',␣
↪'Wisconsin', 'Minnesota')

def had_death_penalty_in_1971(state):
    """Returns True if the argument is the name of a state that had the death␣
↪penalty in 1971."""
    # The implementation of this function uses a bit of syntax
    # we haven't seen before.  Just trust that it behaves as its
    # documentation claims.
    return state not in non_death_penalty_states
```

```
states = murder_rates.group('State').select('State')
death_penalty = states.with_column('Death Penalty', states.
 ↪apply(had_death_penalty_in_1971, 0))
death_penalty
```

```
[11]: State       | Death Penalty
Alabama     | True
Alaska      | False
Arizona     | True
Arkansas    | True
California  | True
Colorado    | True
Connecticut | True
Delaware    | True
Florida     | True
Georgia     | True
… (40 rows omitted)
```

Question 3.3: Use the `death_penalty` and `murder_rates` tables to find murder rates in 1971 for states with the death penalty before the abolition. Create a new table `preban_rates` that contains the name, year, population, murder rate, and death penalty status (`True` or `False`) for each of these states.

BEGIN QUESTION
name: q3_3

```
[12]: # States that had death penalty in 1971
preban_rates = murder_rates.join("State", death_penalty).where("Year", 1971).
 ↪where("Death Penalty", True) # SOLUTION
preban_rates
```

```
[12]: State       | Year | Population | Murder Rate | Death Penalty
Alabama     | 1971 | 3,479,000  | 15.1        | True
Arizona     | 1971 | 1,849,000  | 6.7         | True
Arkansas    | 1971 | 1,944,000  | 10.5        | True
California  | 1971 | 20,223,000 | 8.1         | True
Colorado    | 1971 | 2,283,000  | 6.5         | True
Connecticut | 1971 | 3,081,000  | 3.1         | True
Delaware    | 1971 | 558,000    | 6.1         | True
Florida     | 1971 | 7,041,000  | 13.3        | True
Georgia     | 1971 | 4,664,000  | 16          | True
Idaho       | 1971 | 732,000    | 3.3         | True
… (34 rows omitted)
```

Question 3.4: Create a table `postban_rates` that contains the same information as `preban_rates`, but for 1973 instead of 1971. `postban_rates` should only contain the states found in `preban_rates`.

BEGIN QUESTION
name: q3_4

[18]:
```
# BEGIN SOLUTION NO PROMPT
states_with_penalty = preban_rates.column("State")
postban_rates = murder_rates.where("Year", 1973).where("State", are.
 ↪contained_in(states_with_penalty))
# END SOLUTION
postban_rates = postban_rates.with_column("Death Penalty", False) # SOLUTION
postban_rates = postban_rates.sort("State")
postban_rates
```

[18]:
```
State       | Year | Population | Murder Rate | Death Penalty
Alabama     | 1973 | 3,539,000  | 13.2        | False
Arizona     | 1973 | 2,058,000  | 8.1         | False
Arkansas    | 1973 | 2,037,000  | 8.8         | False
California  | 1973 | 20,601,000 | 9           | False
Colorado    | 1973 | 2,437,000  | 7.9         | False
Connecticut | 1973 | 3,076,000  | 3.3         | False
Delaware    | 1973 | 576,000    | 5.9         | False
Florida     | 1973 | 7,678,000  | 15.4        | False
Georgia     | 1973 | 4,786,000  | 17.4        | False
Idaho       | 1973 | 770,000    | 2.6         | False
… (34 rows omitted)
```

Question 3.5: Use `preban_rates_copy` and `postban_rates` to create a table `change_in_death_rates` that contains each state's population, murder rate, and whether or not that state had the death penalty for both 1971 and 1973.

Hint: `tbl_1.append(tbl_2)` with create a new table that includes rows from both `tbl_1` and `tbl_2`. Both tables must have the exactly the same columns, in the same order.

BEGIN QUESTION
name: q3_5

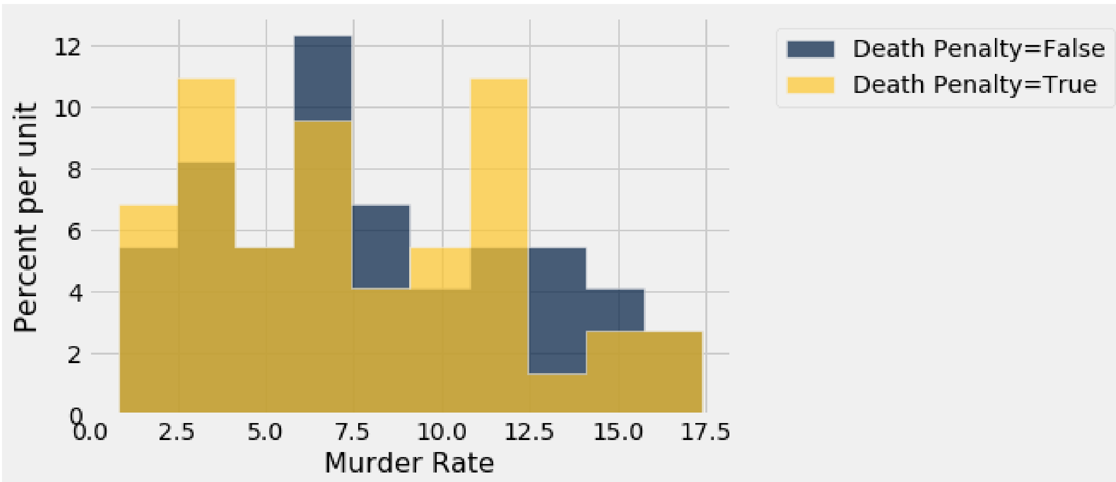
[24]:
```
preban_rates_copy = preban_rates.copy()
change_in_death_rates = preban_rates_copy.append(postban_rates)
change_in_death_rates
```

[24]:
```
State       | Year | Population | Murder Rate | Death Penalty
Alabama     | 1971 | 3,479,000  | 15.1        | True
Arizona     | 1971 | 1,849,000  | 6.7         | True
Arkansas    | 1971 | 1,944,000  | 10.5        | True
California  | 1971 | 20,223,000 | 8.1         | True
Colorado    | 1971 | 2,283,000  | 6.5         | True
Connecticut | 1971 | 3,081,000  | 3.1         | True
Delaware    | 1971 | 558,000    | 6.1         | True
Florida     | 1971 | 7,041,000  | 13.3        | True
Georgia     | 1971 | 4,664,000  | 16          | True
```

```
Idaho        | 1971 | 732,000     | 3.3          | True
… (78 rows omitted)
```

Run the cell below to view the distribution of death rates during the pre-ban and post-ban time periods.

[25]:
```python
change_in_death_rates.hist('Murder Rate', group = 'Death Penalty')
```



Question 3.6: Create a table `rate_means` that contains the average murder rates for the states that had the death penalty and the states that didn't have the death penalty. It should have two columns: one indicating if the penalty was in place, and one that contains the average murder rate for each group.

BEGIN QUESTION
name: q3_6

[26]:
```python
rate_means = change_in_death_rates.select("Death Penalty", "Murder Rate").
↪group('Death Penalty', np.average) # SOLUTION
rate_means
```

[26]:
```
Death Penalty | Murder Rate average
False         | 8.12045
True          | 7.51364
```

Question 3.7: We want to figure out if there is a difference between the distribution of death rates in 1971 and 1973. What should the test statistic be? How does it help us differentiate whether the data supports the null and alternative?

If you are in lab, confirm your answer with a lab TA/LA before moving on.

BEGIN QUESTION
name: q3_7

SOLUTION: We want to find the difference between the mean death rates in the pre-ban and post-ban states. Low values of this difference support the null, which high values of this different support the alternative

Question 3.8: Set `observed_difference` to the observed test statistic using the `rate_means` table

```
BEGIN QUESTION
name: q3_8
```

[123]:
```
# BEGIN SOLUTION NO PROMPT
means = rate_means.column(1)
# END SOLUTION
observed_difference = means.item(0) - means.item(1) #SOLUTION
observed_difference
```

[123]: 0.6068181600659095

Question 3.9: Given a table like `change_in_death_rates`, a value column `label`, and a group column `group_label`, write a function that calculates the appropriate test statistic.

```
BEGIN QUESTION
name: q3_9
```

[126]:
```
def find_test_stat(table, label, group_label):
    # BEGIN SOLUTION
    reduced = table.select(label, group_label)
    means_table = reduced.group(group_label, np.average)
    means = means_table.column(1)
    return means.item(0) - means.item(1)
    # END SOLUTION

find_test_stat(change_in_death_rates, "Murder Rate", "Death Penalty")
```

[126]: 0.6068181600659095

When we run a simulation for A/B testing, we resample by shuffling the labels of the original sample. If the null hypothesis is true and the murder rate distributions are the same, we expect that the difference in mean death rates will be not change when "Death Penalty" labels are changed.

Question 3.10: Write a function `simulate_and_test_statistic` to compute one trial of our A/B test. Your function should run a simulation and return a test statistic.

```
BEGIN QUESTION
name: q3_10
```

[128]:
```
def simulate_and_test_statistic(table, label, group_label):
    # BEGIN SOLUTION
    shuffled_labels = table.sample(with_replacement = False).column("Death␣
↪Penalty")
```

```
    original_and_shuffled = change_in_death_rates.with_column('Death Penalty',␣
↪shuffled_labels)
    return find_test_stat(original_and_shuffled, label, group_label)
    # END SOLUTION

simulate_and_test_statistic(change_in_death_rates, "Murder Rate", "Death␣
↪Penalty")
```

[128]: 0.7840908780250002

Question 3.11: Simulate 5000 trials of our A/B test and store the test statistics in an array called `differences`

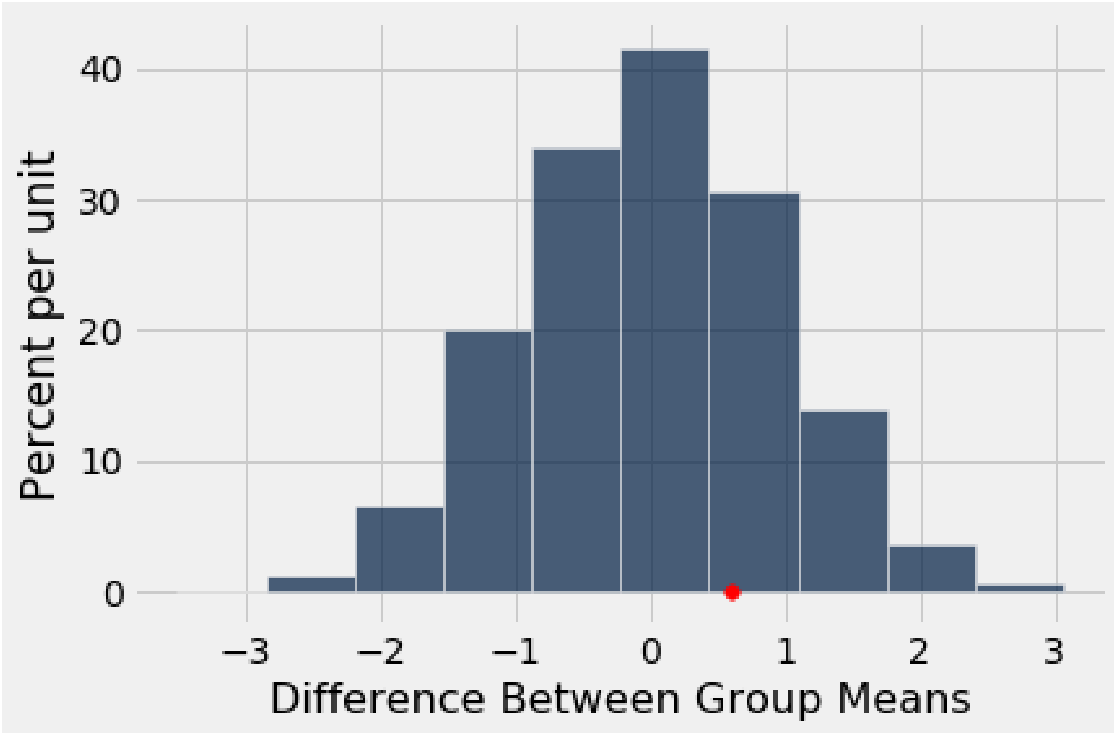
```
BEGIN QUESTION
name: q3_11
```

[103]:
```
# This cell might take a couple seconds to run
differences = make_array()

# BEGIN SOLUTION
repetitions = 5000
for i in np.arange(repetitions):
    new_difference = simulate_and_test_statistic(change_in_death_rates, "Murder␣
↪Rate", "Death Penalty")
    differences = np.append(differences, new_difference)
# END SOLUTION

differences
```

[103]:
```
array([ 0.71136371, -0.50227278,  0.0840909 , …,  1.28409098,
        1.16136365,  0.04772724])
```

Run the cell below to view a histogram of your simulated test statistics plotted with your observed test statistic

[107]:
```
Table().with_column('Difference Between Group Means', differences).hist()
plt.scatter(observed_difference, 0, color='red', s=30, zorder=2);
```



Question 3.12: Find the p-value for your test and assign it to `empirical_P`

BEGIN QUESTION
name: q3_12

```
[97]: empirical_P = np.count_nonzero(differences >= observed_difference) /␣
       ↪repetitions # SOLUTION
      empirical_P
```

```
[97]: 0.2644
```

Question 3.13: Using a 5% P-value cutoff, draw a conclusion about the null and alternative hypotheses. Describe your findings using simple, non-technical language. What does your analysis tell you about murder rates after the death penalty was suspended? What can you claim about causation from your statistical analysis?

BEGIN QUESTION
name: q3_13

SOLUTION: Our p-value (staff solutions had a p-value of 0.256) is greater than the cutoff of 0.05. Our data supports the null hypothesis that murder rates don't significantly change due to the ban of the death penalty. Any observed different was due to chance.

You're done! Congratulations. Run the cells below to check your work and submit to okpy.

[ ]:
```python
# For your convenience, you can run this cell to run all the tests at once!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]
print("Finished running all tests.")
```

[ ]:
```python
# Submit the project!
_ = ok.submit()
```