# lab08

April 19, 2022

Name: Allan Gongora

Section: 0131

## 1 Lab 8: Inference and Capital Punishment, Part 1

Welcome to Lab 8! Over the next two labs, you will investigate the data relevant to a hotly debated social issue: the possible influence of capital punishment (the death penalty) on murder rates in the United States.

Lab 8 is Part 1 of the investigation. Lab 9 is Part 2 of the investigation. Lab 9 will be released next week.

By the end of Lab 9, you should know how to:

1. Test whether observed data appears to be a random sample from a distribution.
2. Analyze a natural experiment.
3. Implement and interpret a sign test.
4. Create a function to run a general hypothesis test.
5. Analyze visualizations and draw conclusions from them.

**Advice.** Develop your answers incrementally. To perform a complicated table manipulation, break it up into steps, perform each step on a different line, give a new name to each result, and check that each intermediate result is what you expect by displaying it. You can add additional names or functions to the provided cells in order to organize your work.

To get started, load `datascience`, `numpy`, `plots`, and `gofer`.

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests.

[1]:
```
pip install gofer-grader
```

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-
packages (1.1.0)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
```

```
packages (from jinja2->gofer-grader) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
from datascience import *
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

from gofer.ok import check
```

**Recommended Reading**: * Sampling and Empirical Distributions * Testing Hypotheses

1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include:
   A) Sentence reponses to questions that ask for an explanation
   B) Numeric responses to multiple choice questions
   C) Programming code
2) Moreover, throughout this lab and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Otherwise, you will fail tests that you thought you were passing previously!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

## 1.1 Overview

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 some 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: Does instituting a death penalty as a punishment 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.

Here is a road map for Part 1:

1. In Section 1, we'll visualize and explore the main dataset we'll be using.
2. In Section 2, we'll test a hypothesis.

### 1.1.1 Data

The main data source for this project comes from a paper by three researchers, Dezhbakhsh, Rubin, and Shepherd. The dataset contains rates of various violent crimes for every year between 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.)

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

```
[3]: 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)
```

## 1.2 1. Murder Rates

The `murder_rates` table isn't enough to demonstrate an *association* between crimes and punishments. We would like to check for an association between murder rates and the existence of capital punishment, for each pair of a state and a year.

**Question 1.1** What additional information will we need before we can check for that association? Assign `extra_info` to a Python list (i.e. [#] or [#, #, …]) containing the number(s) for all of the additional facts below that we *require* in order to check for an association.

1) What year(s) the death penalty was introduced in each state (if any).

2) Day to day data about when murders occurred.

3) What year(s) the death penalty was abolished in each state (if any).

4) Rates of other crimes in each state.

[4]:
```
extra_info = [1, 3]
```

[5]:
```
check("tests/q1_1.py")
```

[5]:
```
<gofer.ok.OKTestsResult at 0x7fa330065810>
```

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 the murder rate trends for a few states, just to see the variety.

**Question 1.2** 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 |

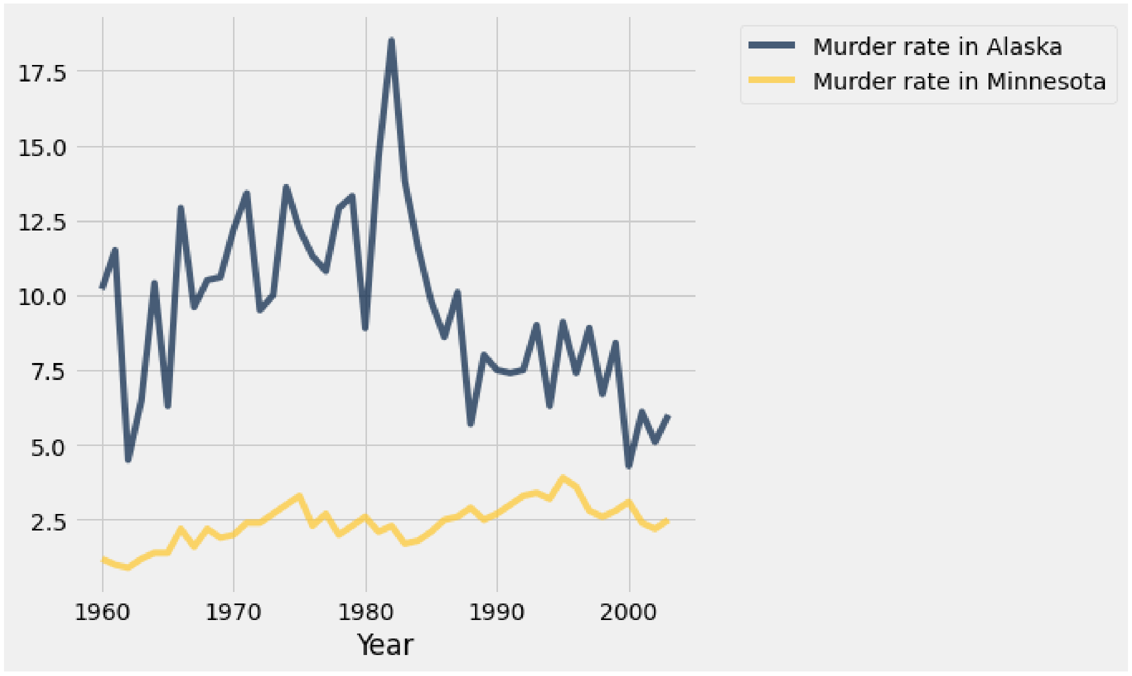
… (41 rows omitted)

[6]:
```
# 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)
ak_mn
```

[6]:
```
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)
```

[7]:
```python
# Draw your line plot here
ak_mn.plot("Year")
```



[8]:
```python
check("tests/q1_2.py")
```

[8]:
```
<gofer.ok.OKTestsResult at 0x7fa31edd7250>
```

What about the murder rates of other states? Say, for example, California and New York? Fill in the cell below to plot the murder rates of different pairs of states. **Note:** this should use similar code to Question 1.2, with only the variable names changed. The cell below will not be graded, but it creates a cool interactive module!

[9]:
```python
# 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')
    plots.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')
            )
```

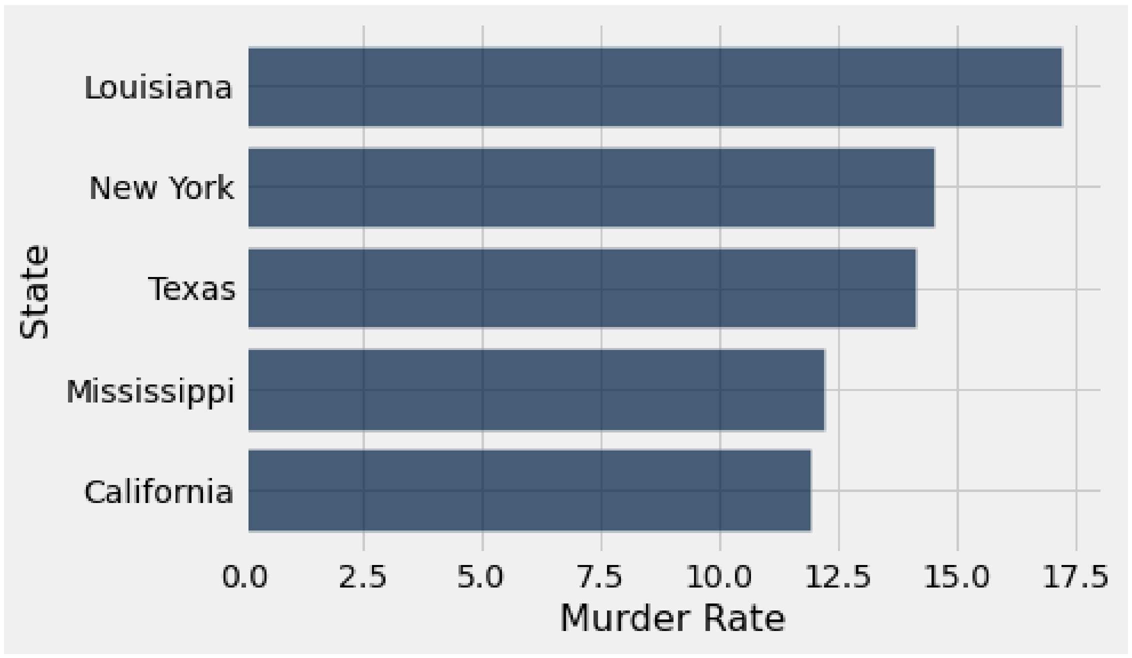
```
interactive(children=(Dropdown(description='state1', index=4,␣
↪options=('Alabama', 'Alaska', 'Arizona', 'Arkans…
```

**Question 1.3** Implement the function `most_murderous`, which takes a year (an integer) as its argument. It does two things: 1. It draws a horizontal bar chart of the 5 states that had the highest murder rates in that year. 2. It returns an array of the names of these states in order of *increasing* murder rate.

Assume that the argument is a year in `murder_rates`. You do not need to check that it is.

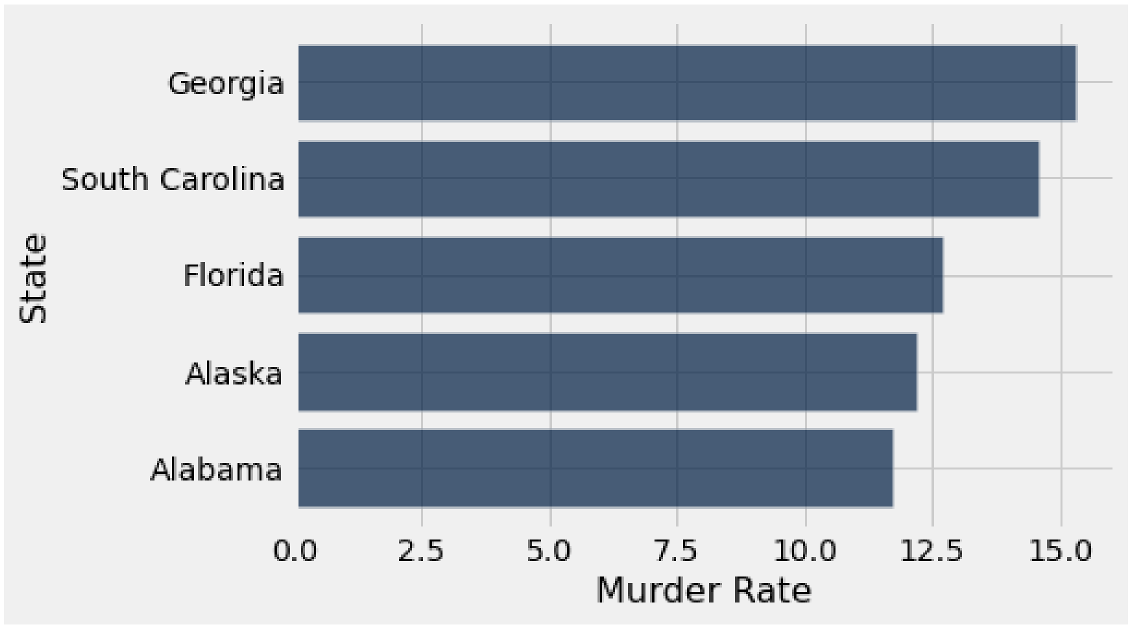
[10]:
```
def most_murderous(year):
    # Assign most to a table of the most murderous states this year in␣
↪ascending order.
    data_for_year = murder_rates.where("Year", year)
    sorted_data = data_for_year.sort("Murder Rate", descending=True)
    top_5 = sorted_data.take(range(5))
    top_5.barh('State', 'Murder Rate')
    return top_5.column('State')[::-1] # this is really dumb. why does it want␣
↪the names in reverse order?

most_murderous(1990) # California, Mississippi, ...,
```

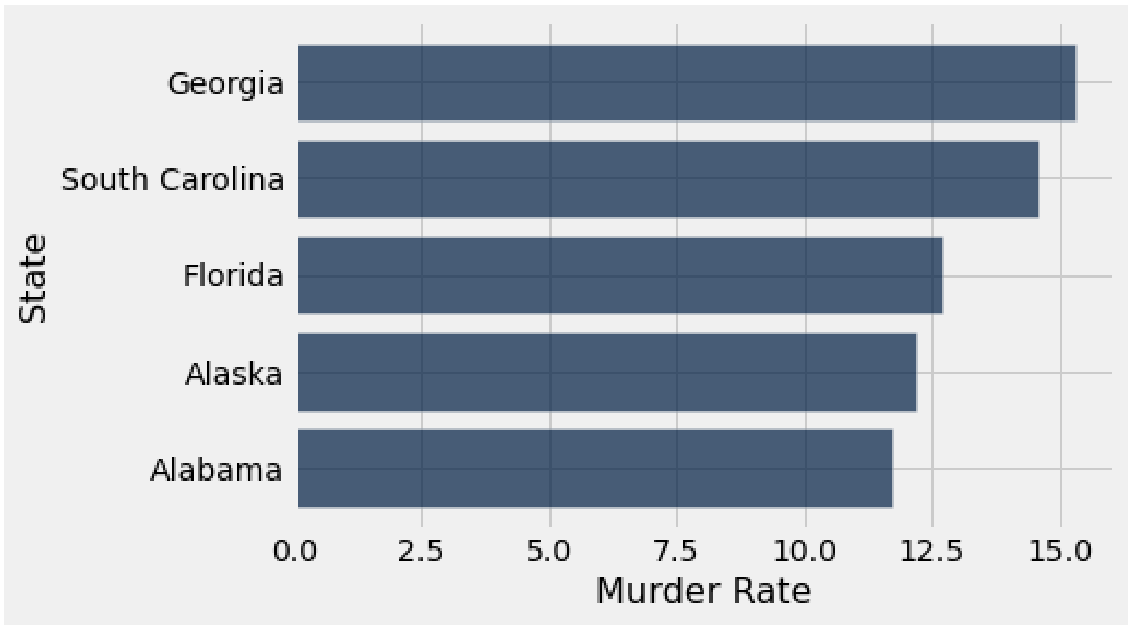
[10]:
```
array(['California', 'Mississippi', 'Texas', 'New York', 'Louisiana'],
      dtype='<U14')
```

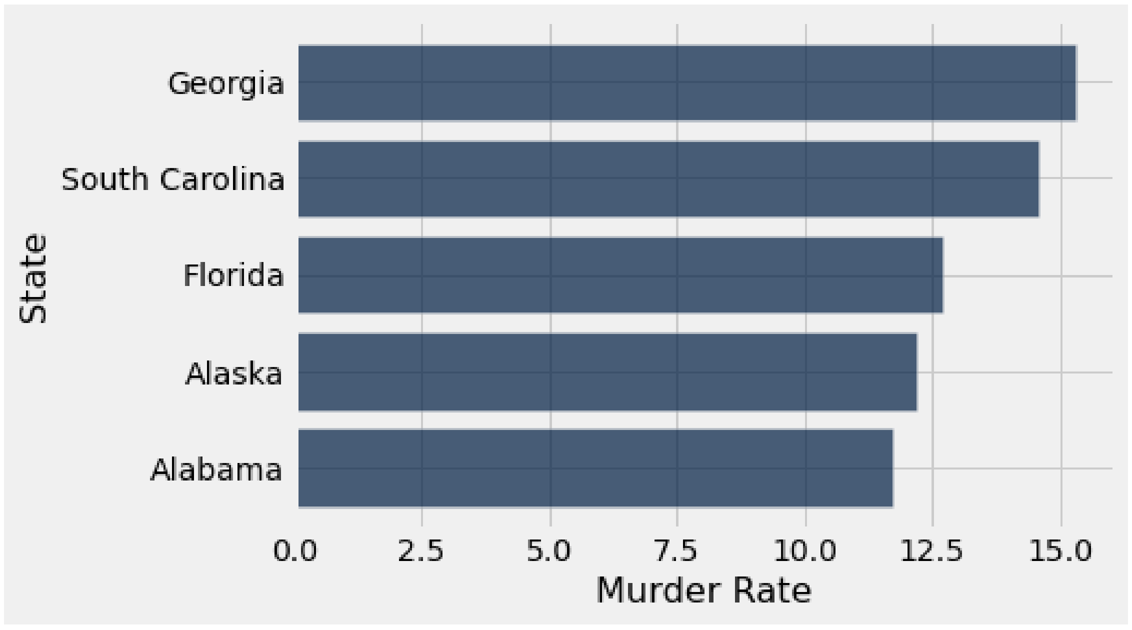


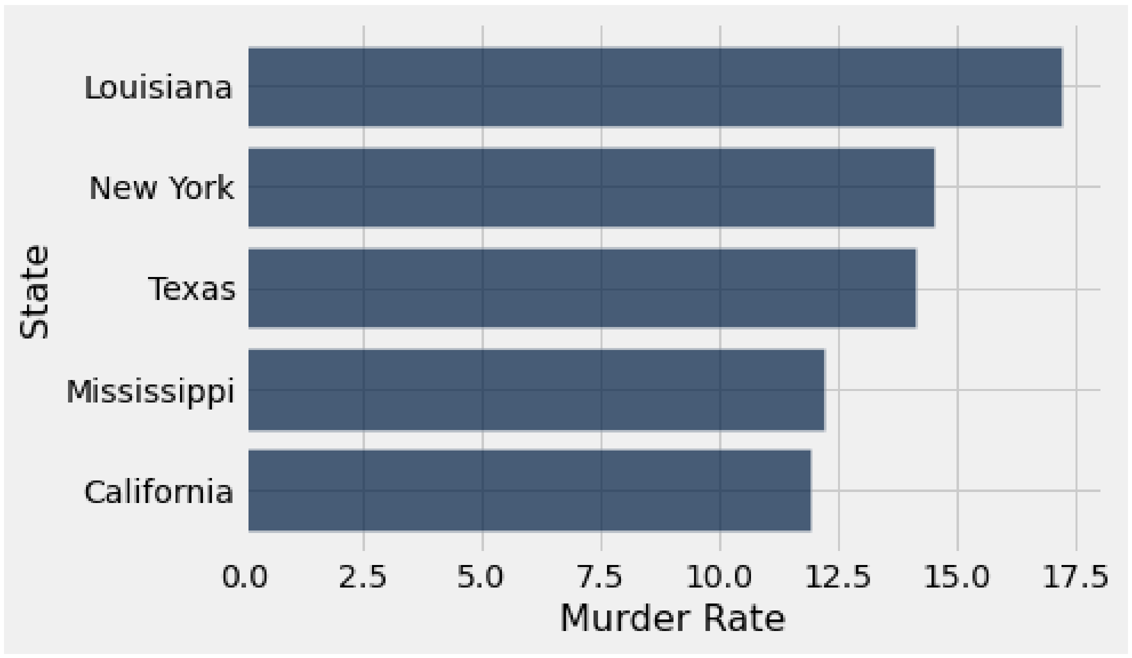
```
[11]: # When you run this cell, several bar charts will be displayed. You can ignore␣
      ↪them.
      check("tests/q1_3.py")
```

```
[11]: <gofer.ok.OKTestsResult at 0x7fa31eceaa90>
```




Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate


Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate



**Question 1.4** How many more people were murdered in California in 1988 than in 1975? Assign `ca_change` to the answer.

*Hint*: Make sure you understand how murder rate is calculated. Recall the formula given at the beginning of the project:

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

Feel free to define new variables and use additional lines to compute your answer. However, **make sure not to change any existing variable names**.

[12]:
```
def murder_rate_to_murders(year: int, state: str) -> int:
    state_year_stats = murder_rates.where('State', state).where("Year", year)
    murder_rate = state_year_stats["Murder Rate"][0]
    pop = state_year_stats["Population"][0]
    return (murder_rate * pop) / 100000
```

[13]:
```
ca = murder_rates.where('State', are.equal_to('California'))
ca_change = murder_rate_to_murders(1988, "California") -␣
 ↪murder_rate_to_murders(1975, "California")
np.round(ca_change)
```

[13]:
```
726.0
```

[14]:
```
check('tests/q1_4.py')
```

```
[14]: <gofer.ok.OKTestsResult at 0x7fa31e98b550>
```

### 1.3 2. Changes in Murder Rates

In this section, we'll see how to test this null hypothesis: "For a set of U.S. states, the murder rate was equally likely to go up or down each year."

Murder rates vary widely across states and years, presumably due to the vast array of differences among states and across US history. Rather than attempting to analyze rates themselves, here we will restrict our analysis to whether or not murder rates increased or decreased over certain time spans. **We will not concern ourselves with how much rates increased or decreased; only the direction of the changes** - whether they increased or decreased.

The `np.diff` function takes an array of values and computes the differences between adjacent items of a list or array as such:

```
[item 1 - item 0 , item 2 - item 1 , item 3 - item 2, ...]
```

Instead, we may wish to compute the difference between items that are two positions apart. For example, given a 5-element array, we may want:

```
[item 2 - item 0 , item 3 - item 1 , item 4 - item 2]
```

The `diff_n` function below computes this result. Don't worry if the implementation uses unfamiliar features of Python, as long as you understand its behavior.

```
[15]: def diff_n(values, n):
          return np.array(values)[n:] - np.array(values)[:-n]

      diff_n(make_array(1, 10, 100, 1000, 10000), 2)
```

```
[15]: array([  99,  990, 9900])
```

**Question 2.1** Implement the function `two_year_changes` that takes an array of murder rates for a state, ordered by increasing year. For all two-year periods (e.g., from 1960 to 1962), it computes and returns **the number of increases minus the number of decreases.**

For example, the array `r = make_array(10, 7, 12, 9, 13, 9, 11)` contains 3 increases (10 to 12, 7 to 9, and 12 to 13), 1 decrease (13 to 11), and 1 change that is neither an increase or decrease (9 to 9). Therefore, `two_year_changes(r)` would return 2, the difference between 3 increases and 1 decrease.

*Hint*: Consider using the `diff_n` function.

```
[16]: def two_year_changes(rates):
          "Return the number of increases minus the number of decreases after two␣
       ↪years."
          changes = diff_n(rates, 2)
          return len(changes[changes > 0]) - len(changes[changes < 0])

      print('Alaska:',    two_year_changes(ak.column('Murder rate in Alaska')))
      print('Minnesota:', two_year_changes(mn.column('Murder rate in Minnesota')))
```

```
Alaska: -5
Minnesota: 6
```

[17]:
```
check("tests/q2_1.py")
```

[17]:
```
<gofer.ok.OKTestsResult at 0x7fa31e961250>
```

We can use `two_year_changes` to summarize whether rates are mostly increasing or decreasing over time for some state or group of states. Let's see how it varies across the 50 US states.

**Question 2.2** Assign `changes_by_state` to a table with one row per state that has two columns: the `State` name and the `Murder Rate two_year_changes` statistic computed across all years in our data set for that state. Its first 2 rows should look like this:

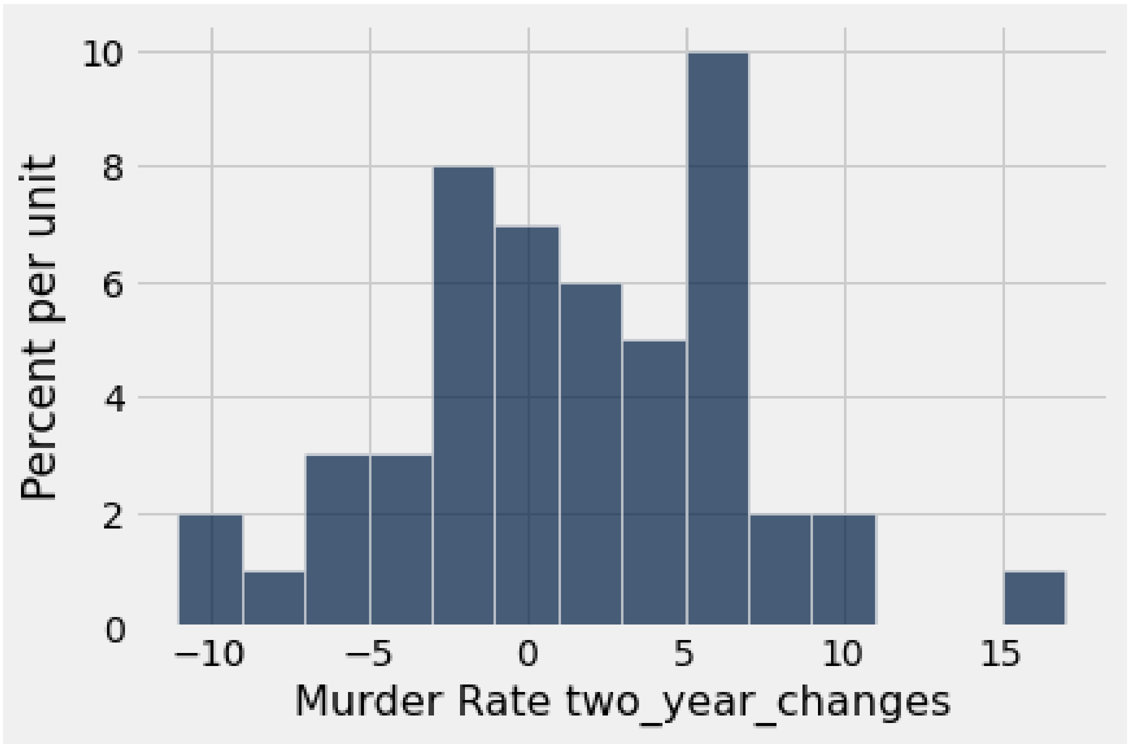
| State | Murder Rate two_year_changes |
|---|---|
| Alabama | -6 |
| Alaska | -5 |

… (48 rows omitted)

[18]:
```
unique = murder_rates.group("State", collect=lambda x: x) # this gets each␣
  ↪states name once and keeps a list of murder rates
# to be used later
changes_by_state = unique.select("State").with_column("Murder Rate␣
  ↪two_year_changes", unique.apply(two_year_changes, "Murder Rate"))
changes_by_state
```

[18]:
```
State       | Murder Rate two_year_changes
Alabama     | -6
Alaska      | -5
Arizona     | 1
Arkansas    | -1
California  | 17
Colorado    | -4
Connecticut | 4
Delaware    | -3
Florida     | -6
Georgia     | -3
… (40 rows omitted)
```

[19]:
```
# Here is a histogram of the two-year changes for the states.
# Since there are 50 states, each state contributes 2% to one bar.
changes_by_state.hist("Murder Rate two_year_changes", bins=np.arange(-11, 19,␣
  ↪2))
```



```
[20]: check("tests/q2_2.py")
```

```
[20]: <gofer.ok.OKTestsResult at 0x7fa31e99c050>
```

Some states have more increases than decreases (a positive change), while some have more decreases than increases (a negative change).

**Question 2.3** Assign `total_changes` to the total increases minus the total decreases for all two-year periods and all states in our data set. For example, if the murder rate in Ohio went up 23 times and fell 17 times, the total change for Ohio would be 6. We want the total value for all the states together.

```
[21]: total_changes = changes_by_state["Murder Rate two_year_changes"].sum() # this␣
        ↪one was very confusing
      print('Total increases minus total decreases, across all states and years:',␣
        ↪total_changes)
```

```
Total increases minus total decreases, across all states and years: 45
```

```
[22]: check("tests/q2_3.py")
```

```
[22]: <gofer.ok.OKTestsResult at 0x7fa31e9bb950>
```

"More increases than decreases," one person exclaims, "Murder rates tend to go up across two-year periods. What dire times we live in."

“Not so fast,” another person replies, “Even if murder rates just moved up and down uniformly at random, there would be some difference between the increases and decreases. There were a lot of states and a lot of years, so there were many chances for changes to happen. If state murder rates increase and decrease at random with equal probability, perhaps this difference was simply due to chance!”

**Question 2.4** What is the total number of distinct pairs of a state and a two-year period? Assign `num_changes` to this value.

For example, Alaska during 1968 to 1970 would count as one distinct pair. Considering all states and all possible two-year periods, how many such pairs are there in total?

[23]:
```
num_changes = 50 * ((unique["Year"][0][-1] - unique["Year"][0][0]) - 1) #␣
 ↪wording here also confusing
# i still dont know what this question is asking. i thought it was just state *␣
 ↪num 2 periods
# but that didnt get me the correct answer it was actually about 2 times bigger␣
 ↪than it was supposed to be
# so instead i just multiplied it by the num years but that was still a bit too␣
 ↪big so i subtracted one (and therfore 50)
# and i got it. If it wasn't because I can cheat by looking at the tests I␣
 ↪wouldn't have gotten to that point.
num_changes
```

[23]:
```
2100
```

[24]:
```
check("tests/q2_4.py")
```

[24]:
```
<gofer.ok.OKTestsResult at 0x7fa31e81fb50>
```

We now have enough information to perform a hypothesis test.

> **Null Hypothesis**: State murder rates increase and decrease over two-year periods as if “increase” or “decrease” were sampled at random from a uniform distribution, like a fair coin flip.

Murder rates can be more likely to go up or more likely to go down. Since we observed 45 more increases than decreases for all two year periods in our dataset, we formulate an alternative hypothesis in accordance with our suspicion:

> **Alternative Hypothesis**: State murder rates are more likely to increase over two-year periods.

If we had observed more decreases than increases, our alternative hypothesis would have been defined accordingly (that state murder rates are more likely to *decrease*). This is typical in statistical testing - we first observe a trend in the data and then run a hypothesis test to confirm or reject that trend.

*Technical note*: These changes in murder rates are not random samples from any population. They describe all murders in all states over all recent years. However, we can imagine that history could have been different, and that the observed changes are the values observed in only one possible world: the one that happened to occur. In this sense, we can evaluate whether the observed "total increases minus total decreases" is consistent with a hypothesis that increases and decreases are drawn at random from a uniform distribution.

*Important requirements for our test statistic:* We want to choose a test statistic for which large positive values are evidence in favor of the alternative hypothesis, and other values are evidence in favor of the null hypothesis. This is because once we've determined the direction of our alternative hypothesis, we only care about the tail in that direction. If, for example, our P-value cutoff was 5%, we'd check to see if our observed test statistic fell within the largest 5% of values in our null hypothesis distribution.

Our test statistic should depend only on whether murder rates increased or decreased, not on the size of any change. Thus we choose:

> **Test Statistic**: The number of increases minus the number of decreases.

The cell below samples increases and decreases at random from a uniform distribution 100 times. The final column of the resulting table gives the number of increases and decreases that resulted from sampling in this way. **Using `sample_from_distribution` is faster than using `sample` followed by `group` to compute the same result.**

[25]:
```
uniform = Table().with_columns(
    "Change", make_array('Increase', 'Decrease'),
    "Chance", make_array(0.5,        0.5))
uniform.sample_from_distribution('Chance', 100)
```

[25]:
```
Change   | Chance | Chance sample
Increase | 0.5    | 55
Decrease | 0.5    | 45
```

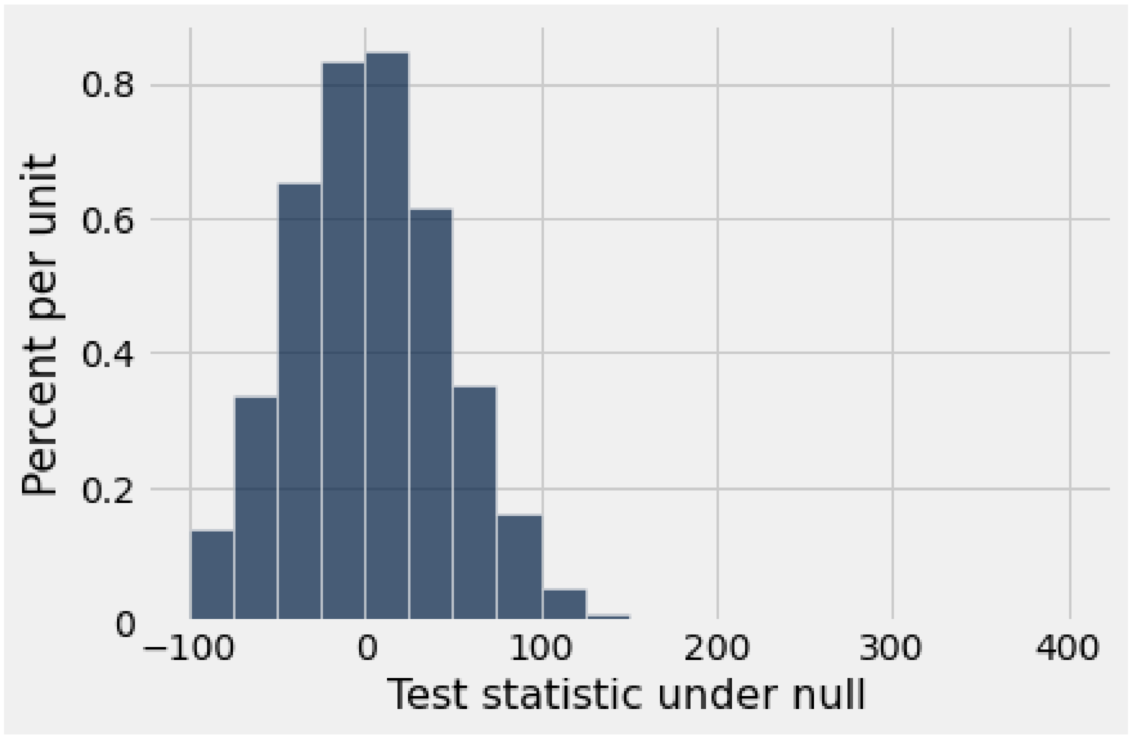
**Question 2.5** Complete the simulation below, which samples `num_changes` increases/decreases at random many times and forms an empirical distribution of your test statistic under the null hypothesis. Your job is to: * fill in the function `simulate_under_null`, which simulates a single sample under the null hypothesis, and * fill in its argument when it's called below.

[26]:
```
def simulate_under_null(num_chances_to_change):
    """Simulates some number changing several times, with an equal
    chance to increase or decrease.  Returns the value of our
    test statistic for these simulated changes.

    num_chances_to_change is the number of times the number changes.
    """
    sample = uniform.sample_from_distribution('Chance', num_chances_to_change)
    return sample["Chance sample"][0] - sample["Chance sample"][1]

uniform_samples = make_array()
for i in np.arange(5000):
    uniform_samples = np.append(uniform_samples,␣
 ↪simulate_under_null(num_changes))
```

```
simulated_changes = Table().with_column('Test statistic under null',␣
 ↪uniform_samples)
simulated_changes
```

[26]:
```
Test statistic under null
42
-20
14
-10
-8
64
-16
-4
44
-40
… (4990 rows omitted)
```

[27]:
```
# Run this cell to visualize the empirical distribution of
# the test statistic under the null hypothesis.
simulated_changes.hist(0, bins=np.arange(-100, 400+25, 25))
```



**Question 2.6** Looking at this histogram, draw a conclusion about whether murder rates basically increase as often as they decrease. (Remember that we're only concerned with the *positive direction* because it supports our alternative hypothesis). You **do not** need to compute a P-value for this

question.

First, set `which_side` to `"Right"` or `"Left"` depending on which side of the histogram you need to look at to make your conclusion.

Then, set `reject_null` to `True` if rates increase more than they decrease, and we can reject the null hypothesis. Set `reject_null` to `False` if the observed difference is typical under the null hypothesis.

[28]:
```
which_side = "Right"
reject_null = False
```

[29]:
```
check("tests/q2_6.py")
```

[29]:
```
<gofer.ok.OKTestsResult at 0x7fa31e853110>
```

## 1.4 3. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

**Double check that you have completed all of the free response questions as the autograder does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this lab.** When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

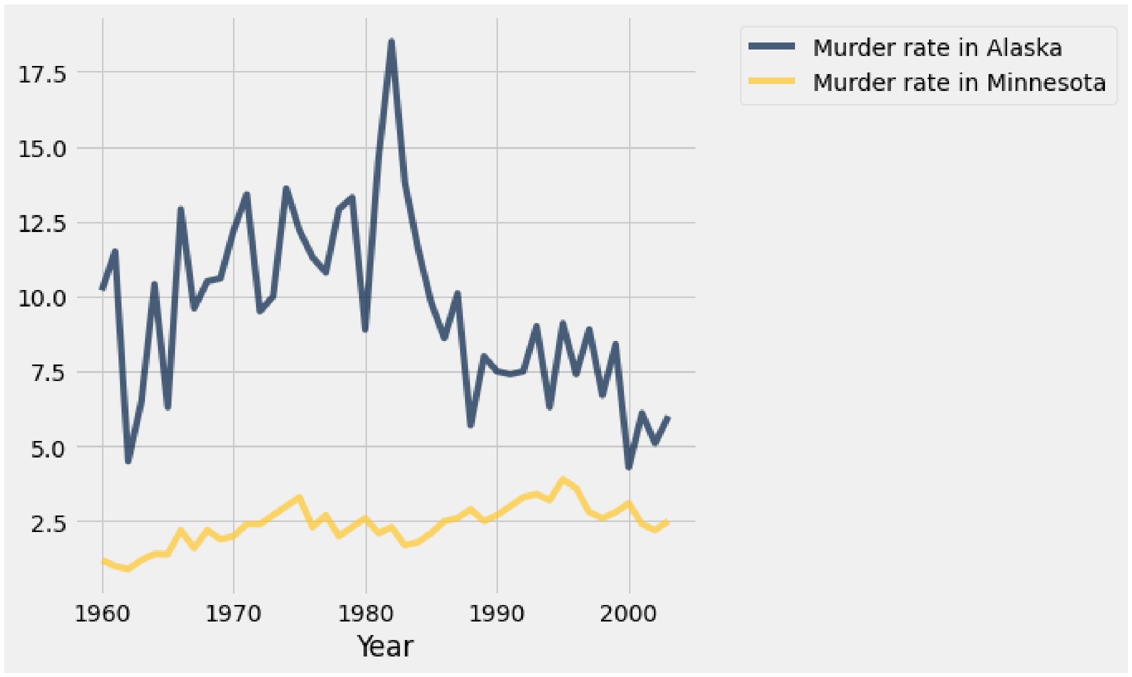
The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

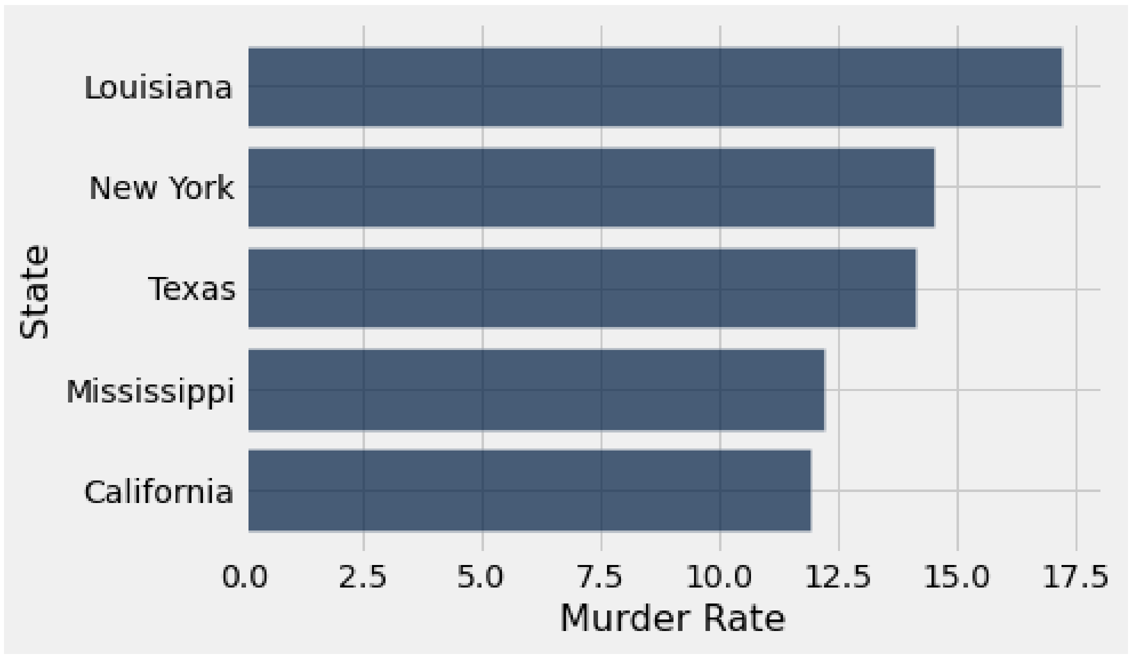
Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

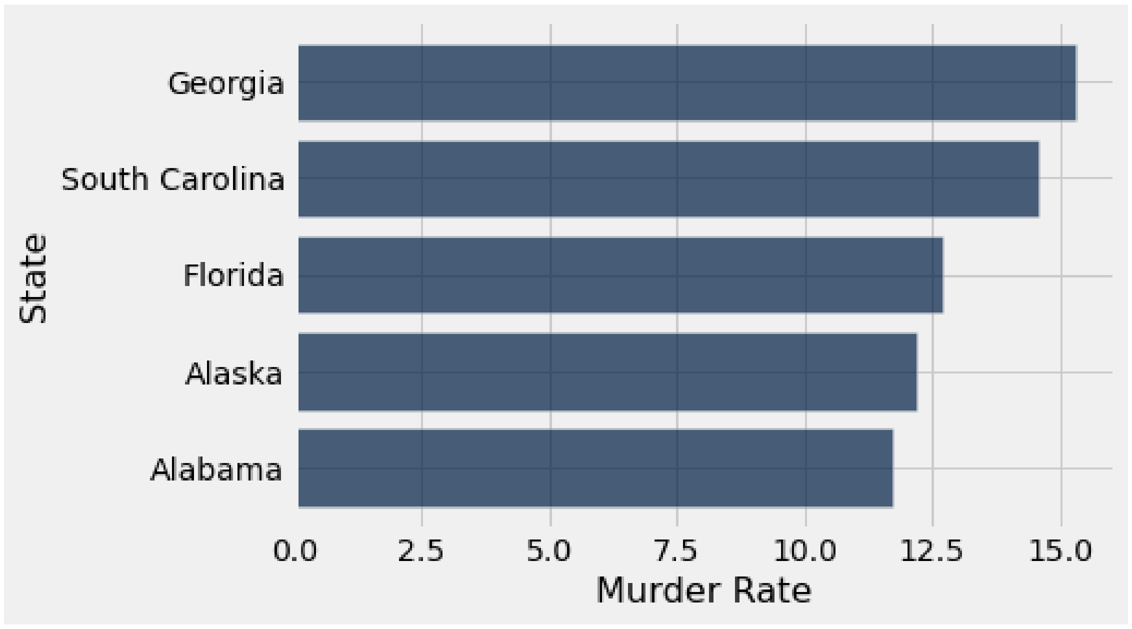
[30]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('lab08.ipynb', sorted(glob.glob('tests/q*.py'))))
```

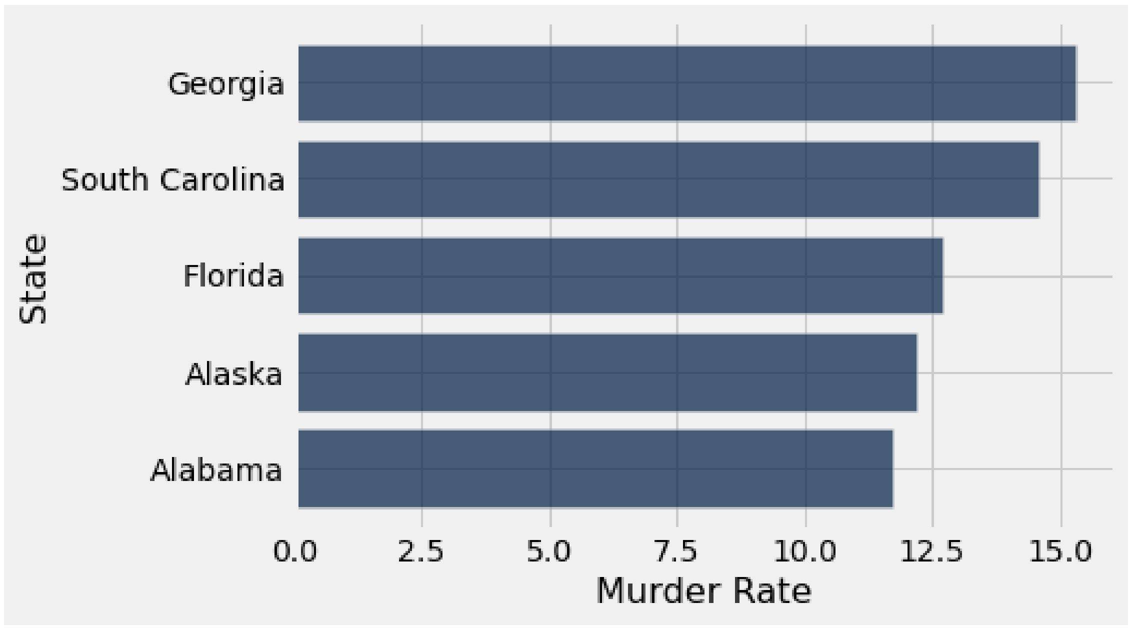
```
Alaska: -5
Minnesota: 6
Total increases minus total decreases, across all states and years: 45
['tests/q1_1.py', 'tests/q1_2.py', 'tests/q1_3.py', 'tests/q1_4.py',
'tests/q2_1.py', 'tests/q2_2.py', 'tests/q2_3.py', 'tests/q2_4.py',
'tests/q2_6.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7fa31e405c50>
```

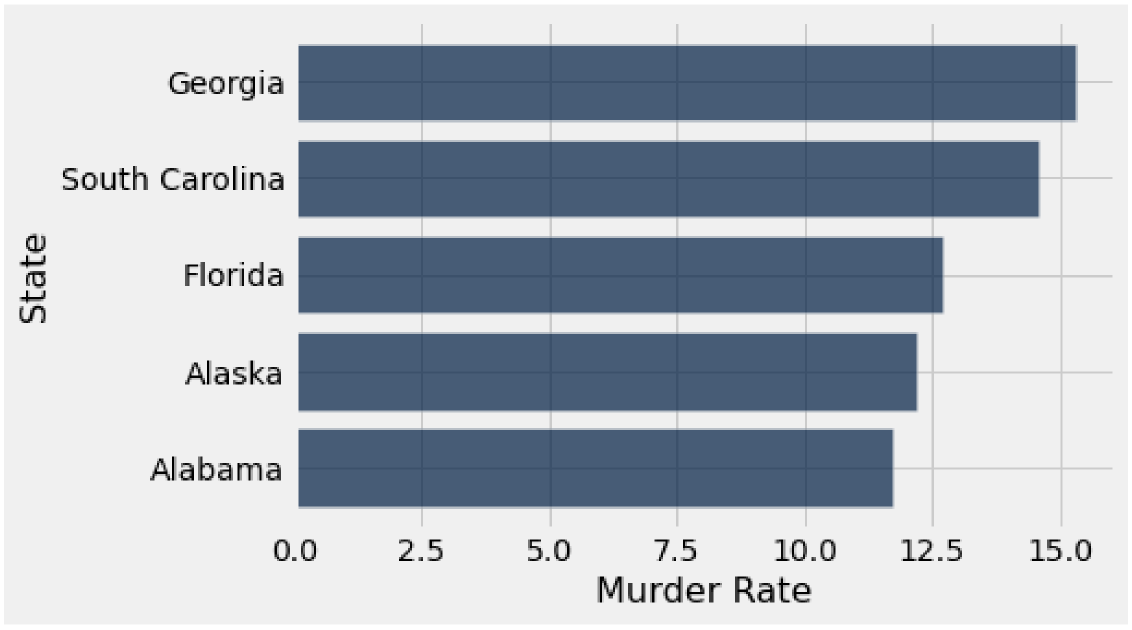
```
Question 2:

<gofer.ok.OKTestsResult at 0x7fa320f9e450>

Question 3:

<gofer.ok.OKTestsResult at 0x7fa31e374690>

Question 4:

<gofer.ok.OKTestsResult at 0x7fa31e1ad1d0>

Question 5:

<gofer.ok.OKTestsResult at 0x7fa31e22c090>

Question 6:

<gofer.ok.OKTestsResult at 0x7fa31e36a7d0>

Question 7:

<gofer.ok.OKTestsResult at 0x7fa31e6d5fd0>

Question 8:

<gofer.ok.OKTestsResult at 0x7fa31e8486d0>

Question 9:

<gofer.ok.OKTestsResult at 0x7fa31e79da10>

1.0
```

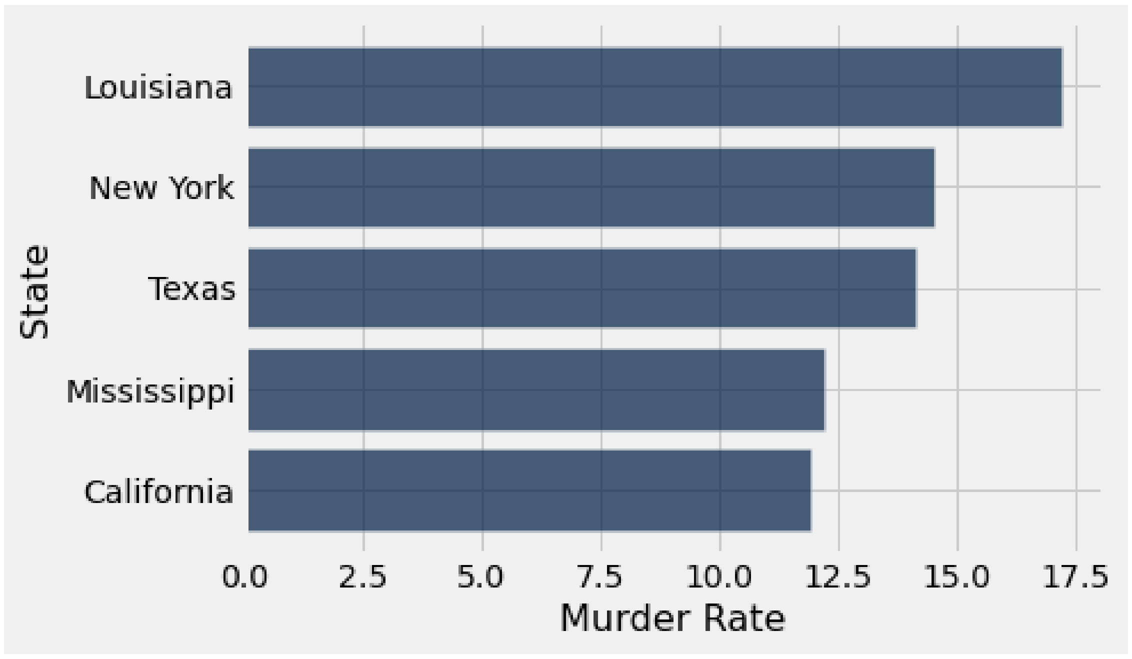


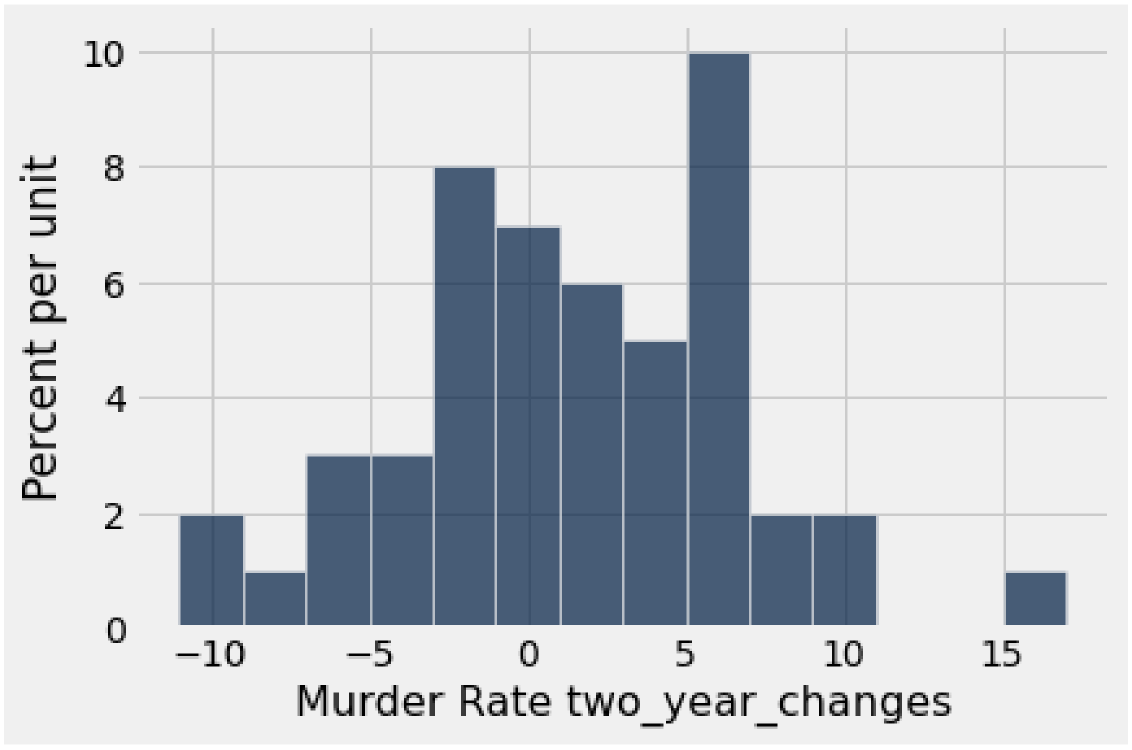

State
Louisiana
New York
Texas
Mississippi
California
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Murder Rate

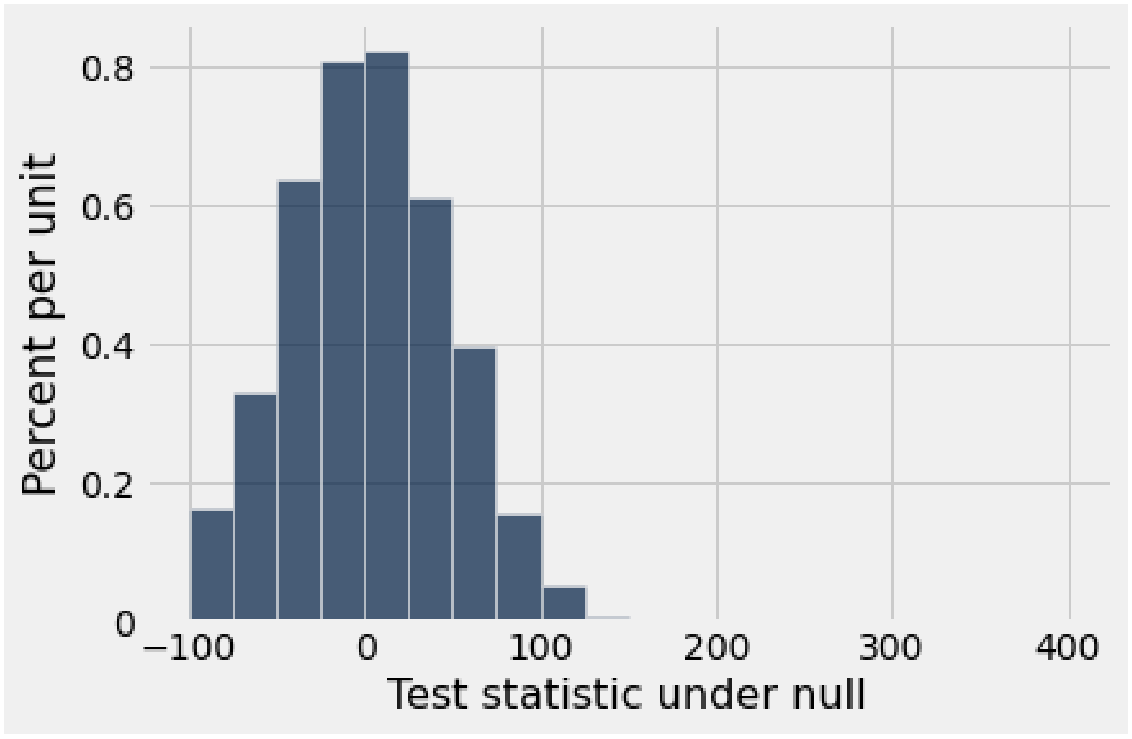

State
Georgia
South Carolina
Florida
Alaska
Alabama
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate

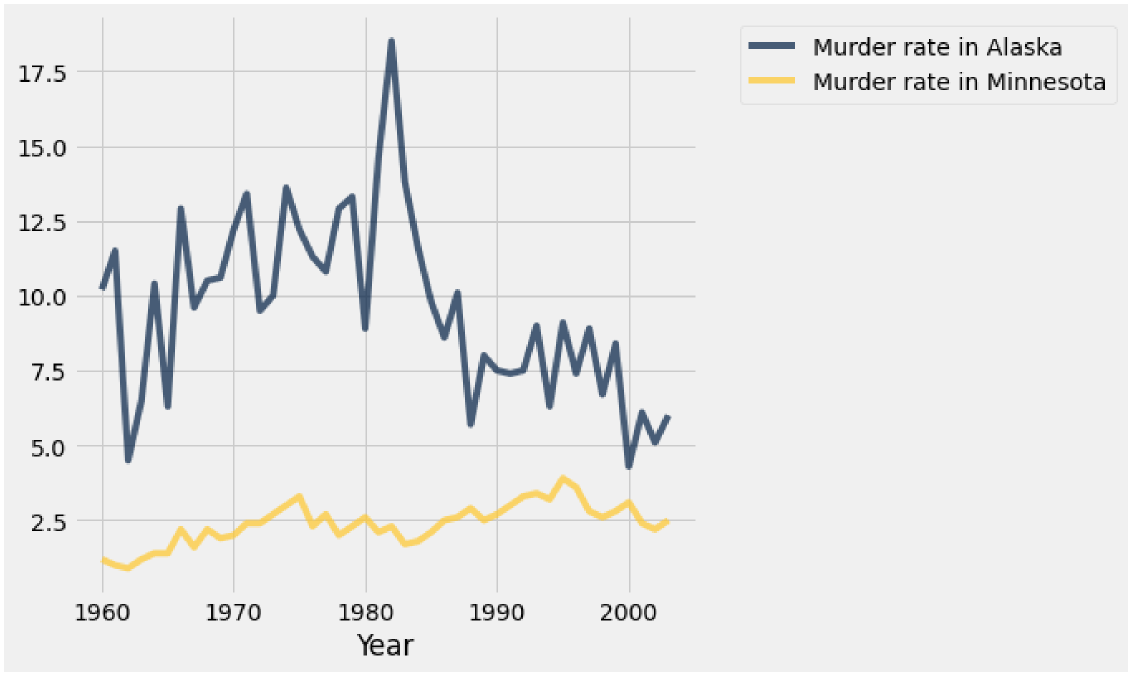

Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate

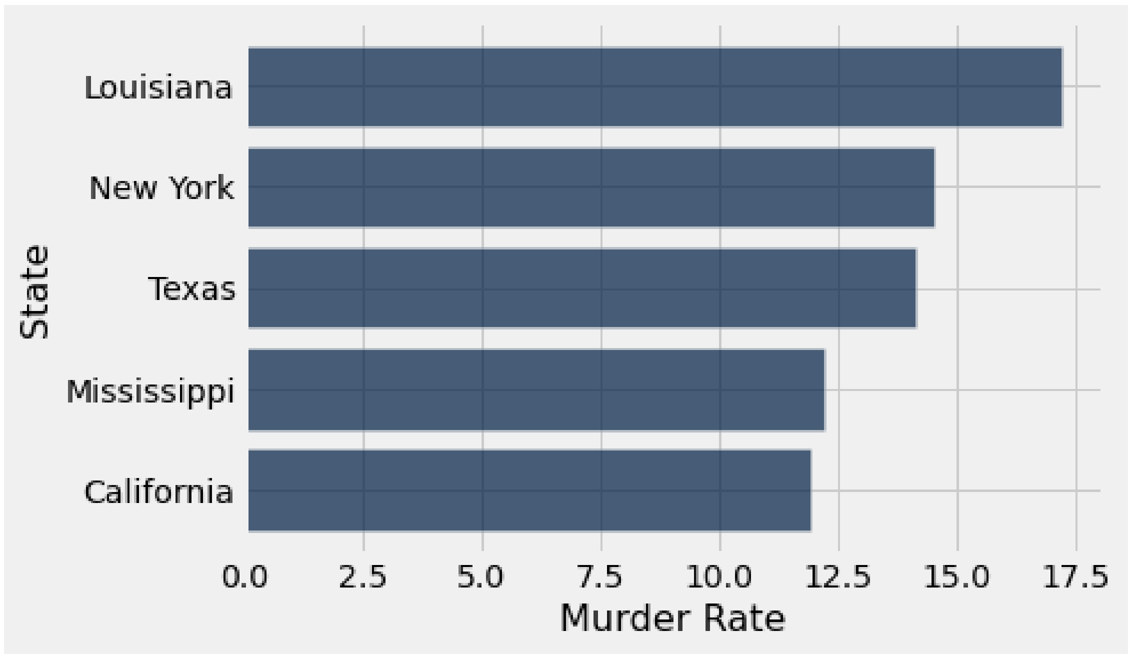

Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate

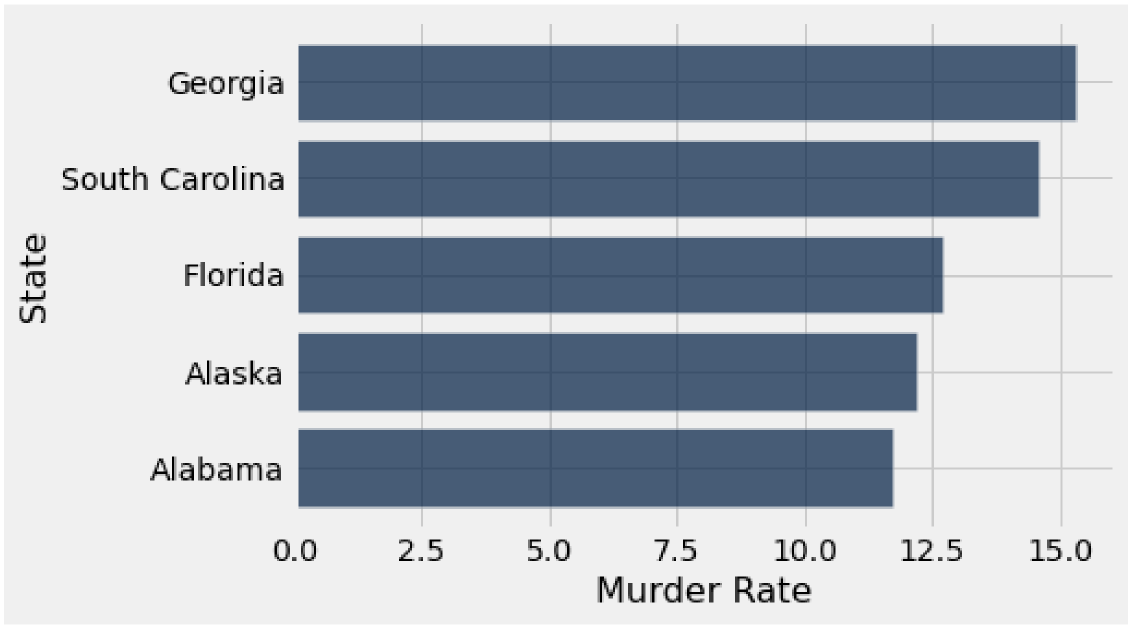

Louisiana
New York
Texas
Mississippi
California
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Murder Rate

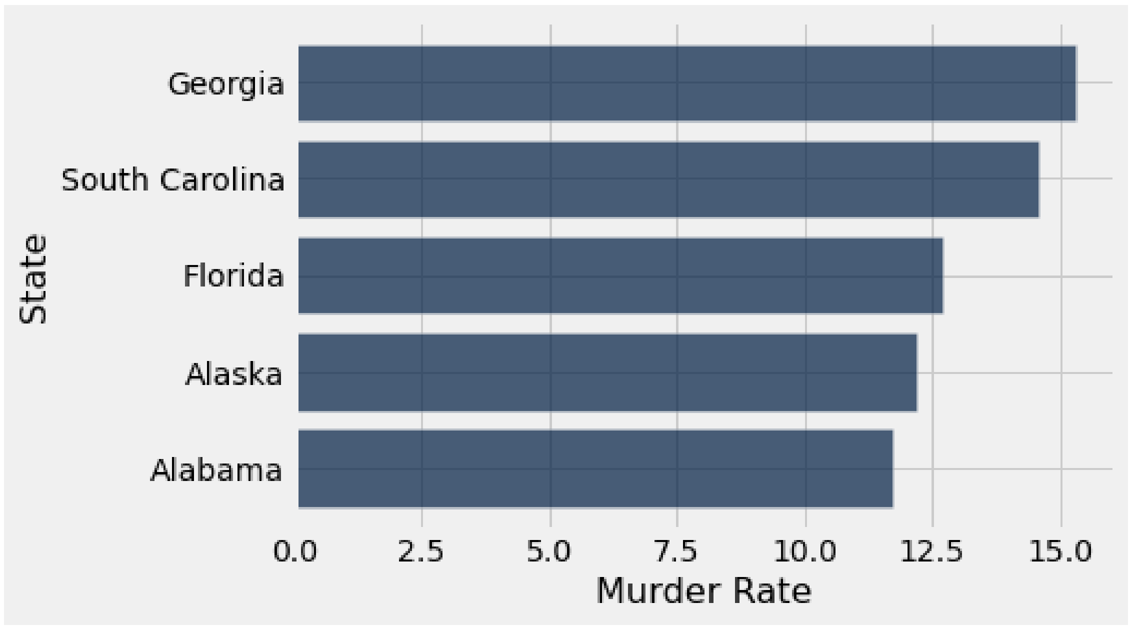

10
8
6
4
2
0
Percent per unit
−10
−5
0
5
10
15
Murder Rate two_year_changes

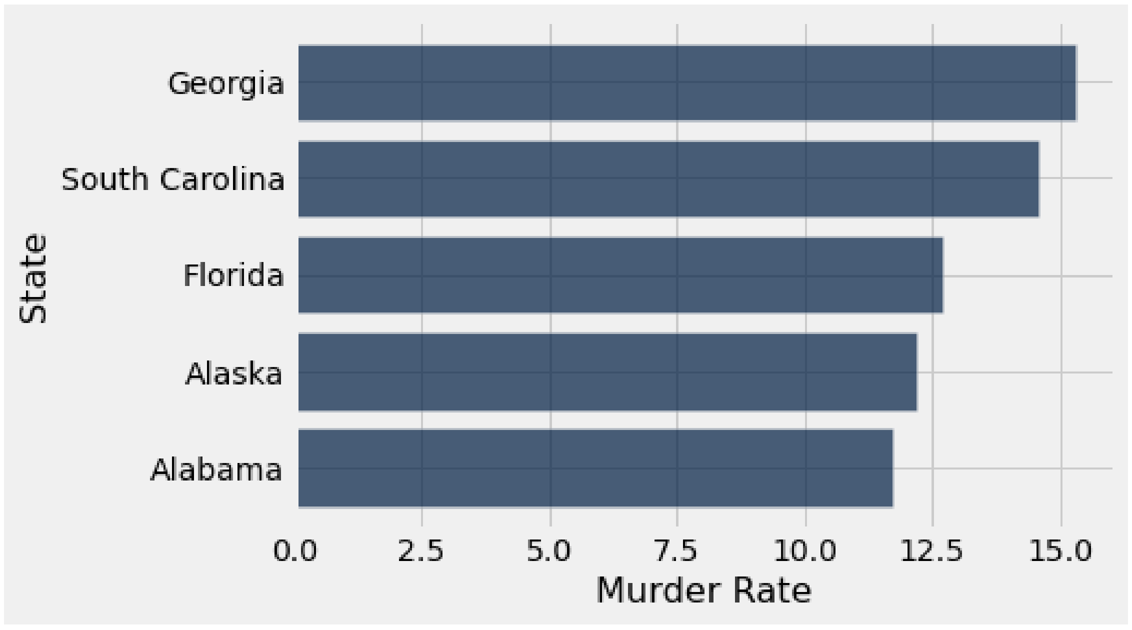

Percent per unit
0.8
0.6
0.4
0.2
0
−100
0
100
200
300
400
Test statistic under null

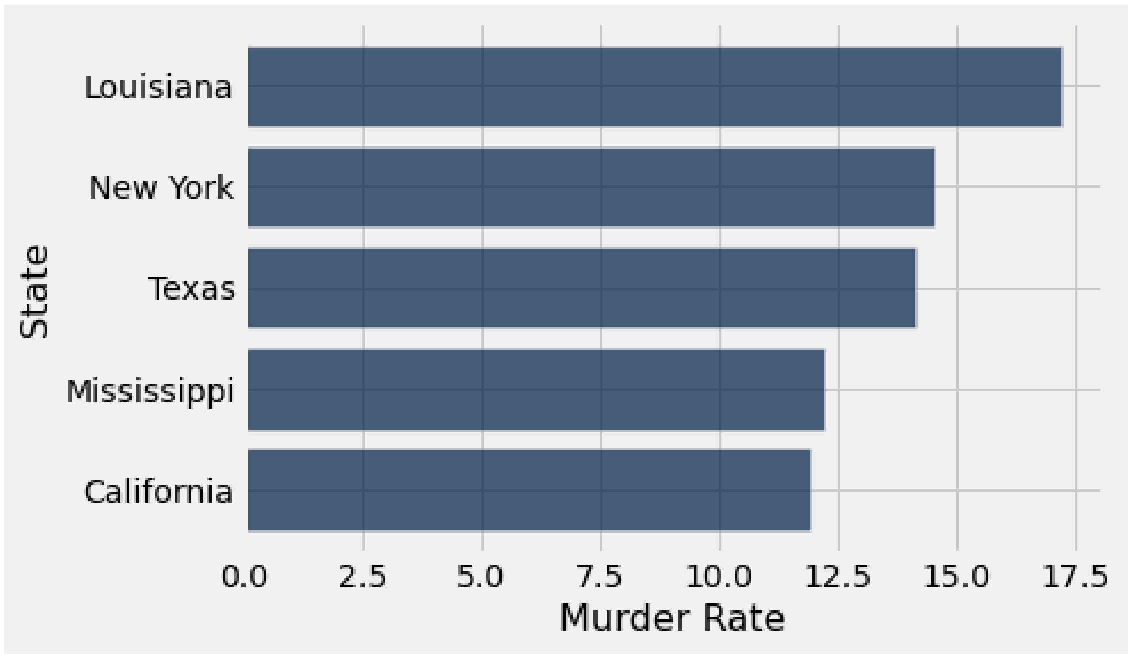

Murder rate in Alaska
Murder rate in Minnesota
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1960
1970
1980
1990
2000
Year

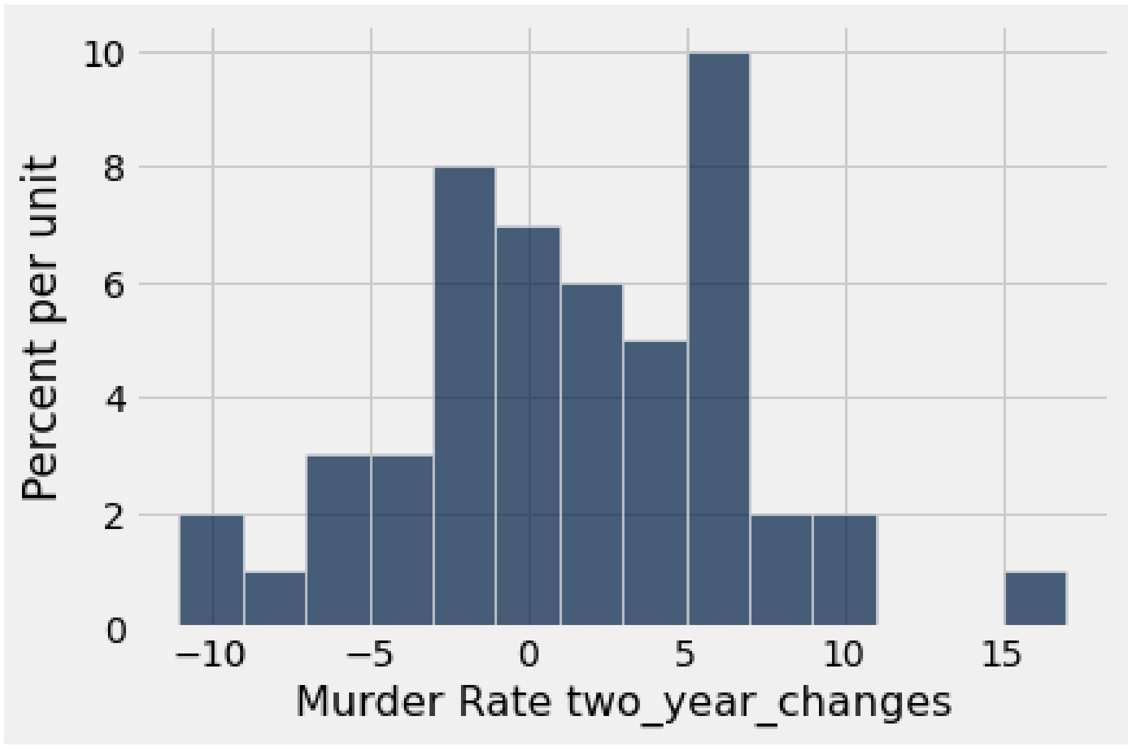

Louisiana
New York
Texas
Mississippi
California
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Murder Rate

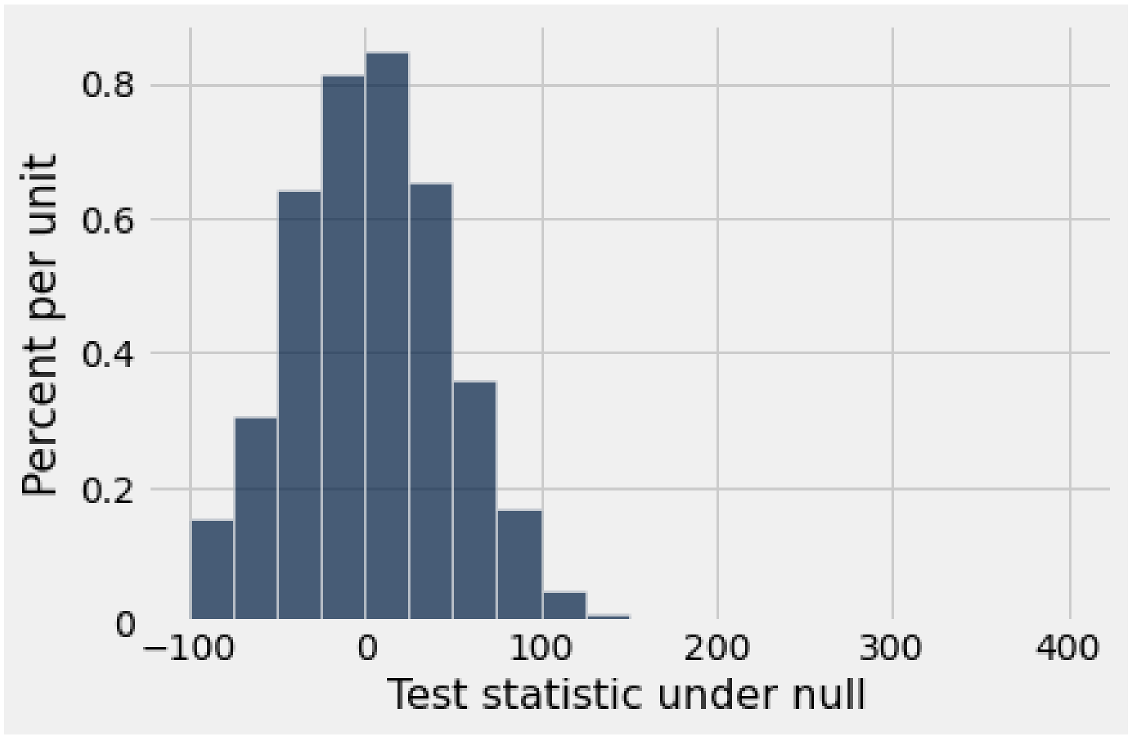

Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate


Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate


Georgia
South Carolina
Florida
Alaska
Alabama
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Murder Rate


Louisiana
New York
Texas
Mississippi
California
State
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Murder Rate


10
8
6
4
2
0
Percent per unit
−10
−5
0
5
10
15
Murder Rate two_year_changes



Name: Allan Gongora

Section: 0131