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

# These lines set up the plotting functionality and formatting.
import matplotlib
matplotlib.use('Agg', warn=False)
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('final_sp20.ok')
_ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Run this cell to set up the notebook, but please don't chan
ge it.
OK, version v1.14.19
=====================================================================
---------------------------------------------------------------------------
LoadingException                          Traceback (most recent call last)
<ipython-input-1-da4505131ac2> in <module>
     15 # These lines load the tests.
     16 from client.api.notebook import Notebook
---> 17 ok = Notebook('final_sp20.ok')
     18 _ = ok.auth(inline=True)

/opt/anaconda3/lib/python3.7/site-packages/client/api/notebook.py in __init__(self, filepath, cmd_args, debug, mode)
     13         ok_logger = logging.getLogger('client')   # Get top-level ok logger
     14         ok_logger.setLevel(logging.DEBUG if debug else logging.ERROR)
---> 15         self.assignment = load_assignment(filepath, cmd_args)
     16         # Attempt a login with enviornment based tokens
     17         login_with_env(self.assignment)

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in load_assignment(filepath, cmd_args)
     22     if cmd_args is None:
     23         cmd_args = Settings()
---> 24     return Assignment(cmd_args, **config)
     25 
     26 def _get_config(config):

/opt/anaconda3/lib/python3.7/site-packages/client/sources/common/core.py in __call__(cls, *args, **kargs)
    185                 raise ex.SerializeException('__init__() missing expected '
    186                                 'argument {}'.format(attr))
--> 187         obj.post_instantiation()
    188         return obj
    189 

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in post_instantiation(self)
    151     def post_instantiation(self):
    152         self._print_header()
--> 153         self._load_tests()
    154         self._load_protocols()
    155         self.specified_tests = self._resolve_specified_tests(

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in _load_tests(self)
    205 
    206         if not self.test_map:
--> 207             raise ex.LoadingException('No tests loaded')
    208 
    209     def dump_tests(self):

LoadingException: No tests loaded
```

# Data 8 Final Exam Spring 2020

- This exam is worth 150 points. You have until Friday, May 15 at 6PM to complete it. Late submissions will not be accepted.
- Do not use features of the Python language that have not been described in this course: we will not accept regrade requests for any issues caused by this.
- A few questions are marked *Just for fun*. These questions are optional and will not be graded.
- This exam is open notes: you may use any resources including the textbook, lecture, lab, homeworks, and projects, and old Piazza posts.
- Piazza will be closed to public posts. For clarifications, typos, and other similar issues with the exam *only*, you may make a private post.
- The collaboration policy for this exam is similar to the homeworks:
  - You may discuss the questions with other students, but any code, explanations, or text you write in this notebook must be your own.
  - You must list the names of any students you discussed the questions with below in Question 0.
- Any clarifications to the exam will be made in this Google doc (https://docs.google.com/document/d/1hWpU4IBBddMM7UsPoZfwKc8Nc61sI1d1L9nt8iSHhLo/edit?usp=sharing). In cases where we make a correction, we'll also make a Piazza announcement that you'll receive in your email. Please check the document and your email before you start or resume working on the exam.
- Unless otherwise stated, you may add cells or additional lines of code anywhere you want. Make sure you do *not* delete any existing cells. Also, do not reuse any variable names.
- Just like most assignments, unless otherwise stated, the public tests will *not* check for correctness.
- When budgeting your time, note that the later questions are longer and more difficult: make sure you leave enough time to attempt the entire exam.

## Useful functions

These are useful functions from lecture and the textbook. They've all been defined correctly. You may use any of them for any question in the exam,

**Warning**: Make sure you don't create any variables or functions that have the same names as any of these, or you may not receive credit on parts of the exam.

### Linear regression

These functions are described in lectures 30-34 (Least Squares through Regression Wrapup), and also in Chapter 15 of the textbook.

In [2]:
```
def standard_units(arr):
    return (arr - np.average(arr))/np.std(arr)

def correlation(t, x, y):
    x_standard = standard_units(t.column(x))
    y_standard = standard_units(t.column(y))
    return np.average(x_standard * y_standard)

def slope(t, x, y):
    r = correlation(t, x, y)
    y_sd = np.std(t.column(y))
    x_sd = np.std(t.column(x))
    return r * y_sd / x_sd

def intercept(t, x, y):
    x_mean = np.mean(t.column(x))
    y_mean = np.mean(t.column(y))
    return y_mean - slope(t, x, y)*x_mean

def fitted_values(t, x, y):
    a = slope(t, x, y)
    b = intercept(t, x, y)
    return a*t.column(x) + b

def fitted_value(t, x, y, new_x):
    a = slope(t, x, y)
    b = intercept(t, x, y)
    return a*new_x + b

def residuals(t, x, y):
    predictions = fitted_values(t, x, y)
    return t.column(y) - predictions
```

# Question 0

Please **list the names and email addresses of everyone you collaborated with** when taking this exam. As a reminder, you are allowed to discuss the questions with other students, but you must write all code and fill in all answers by yourself.

**SOLUTION:**

# Question 1

For each of the following cells, assign the specified variable so that the output of the entire cell is as specified. You may only change the first line of each cell: **don't change anything else**.

For example, if the cell said to assign the variable `x` to produce the output 10, and it looked like this:

```
x = ...
x + 7
```

then the correct answer would be to assign `x = 3`, so that the output of the last line would be 10.

### Question 1.1

(2 points) Assign the variable `a` so that the output of `a + b` is an array of two integers, whose first element is 2020 and whose second element is 8.

```
BEGIN QUESTION
name: q1_1
manual: false
```

```
In [3]: a = make_array(3, 8) # SOLUTION
        # Do not change anything below this line.
        b = make_array(2017, 0)
        a + b
```

```
Out[3]: array([2020,    8])
```

```
In [4]: # TEST
        # Defined `a` to be something
        all([type(a) != type(...), type(a) != None])
```

```
Out[4]: True
```

```
In [5]: # HIDDEN TEST
        all([a[0] == 3, a[1] == 8])
```

```
Out[5]: True
```

### Question 1.2

(2 points) Assign the variable `a2` so that the output of running the below cell is an array of integers whose first element is 2020 and whose second element is 8.

```
BEGIN QUESTION
name: q1_2
manual: false
```

```
In [6]: a2 = make_array(2, 0) # SOLUTION
        # Do not change anything below this line.
        t = Table().with_columns(
            'year', np.arange(2000, 2030, 10),
            'course', make_array(8, 100, 102),
        )
        first_col_index = a2.item(0)
        second_col_index = a2.item(1)
        year = t.column('year').item(first_col_index)
        course = t.column('course').item(second_col_index)
        make_array(year, course)

Out[6]: array([2020,    8])
```

```
In [7]: # TEST
        # Defined `a2` to be something
        all([type(a2) != type(...), type(a2) != None])

Out[7]: True
```

```
In [8]: # HIDDEN TEST
        import numpy as np
        all([type(a2) == np.ndarray, a2.item(0) == 2, a2.item(1) == 0])

Out[8]: True
```

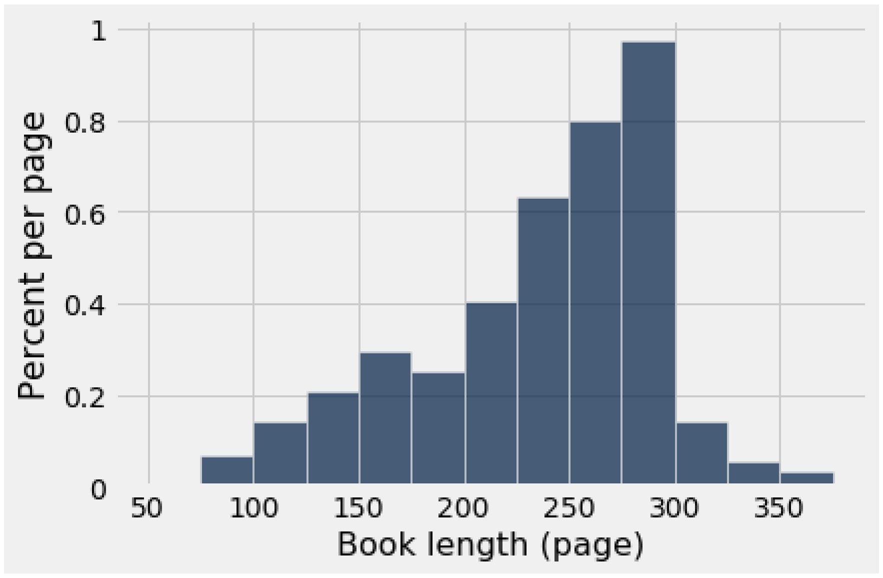
## Question 2: Books and Graphs

Zeynep goes through her family's entire collection of 370 books and records data for each one. She records her data in a table called `books` . The lengths of the books (in pages) are in a column labeled `Length` .

For this question, you won't be able to use the table `books` : you must answer based only on the information provided.

She notices that all of the books are between 50 and 400 pages, and uses the following line of code to generate a histogram showing the lengths:

```
books.hist('Length', unit='page', bins=np.arange(50, 400, 25))
```



**Question 2.1**

(3 points) What proportion of books are between 200 and 224 pages long (inclusive)? Assign the variable `q21_proportion_books_200_224` to your answer.

Your answer should be a number between 0 and 1, and should be correct to within .01.

```
BEGIN QUESTION
name: q2_1
manual: false
```

```
In [9]: q21_proportion_books_200_224 = 25 * 0.4 / 100 # SOLUTION
```

```
In [10]: # TEST
all([type(q21_proportion_books_200_224) != type(...), type(q21_proportio
n_books_200_224) != None])
```

```
Out[10]: True
```

```
In [11]: # TEST
0 <= q21_proportion_books_200_224 <= 1
```

```
Out[11]: True
```

In [12]:
```
# HIDDEN TEST
np.round(q21_proportion_books_200_224, 1) == 0.1
```

Out[12]:
```
True
```

### Question 2.2

(4 points) Based on the histogram and the information above, which must be true? Assign `book_interpretation_choices` to an array of your numbered answer(s).

1. There are more long books (200 pages and over) than short books (under 200 pages) in her family's collection.
2. The percentage of books between 125 and 149 pages (inclusive) is about 20% (that is, 19-21%).
3. The books in her family's collection are like a random sample from the population of all books.
4. In her family's collection, nonfiction books are longer on average than fiction books.
5. The mean length is less than the median length.

```
BEGIN QUESTION
name: q2_2
manual: true
```

In [13]:
```
book_interpretation_choices = make_array(1, 5) # SOLUTION
```

### Question 2.3

(2 points) Which of the following is closest to the median length? Assign the variable `median_length_choice` to either 1, 2, or 3 depending on your choice.

1. 200
2. 260
3. 315

```
BEGIN QUESTION
name: q2_3
manual: false
```

In [14]:
```
median_length_choice = 2 # SOLUTION
```

In [15]:
```
# TEST
type(median_length_choice) == int
```

Out[15]:
```
True
```

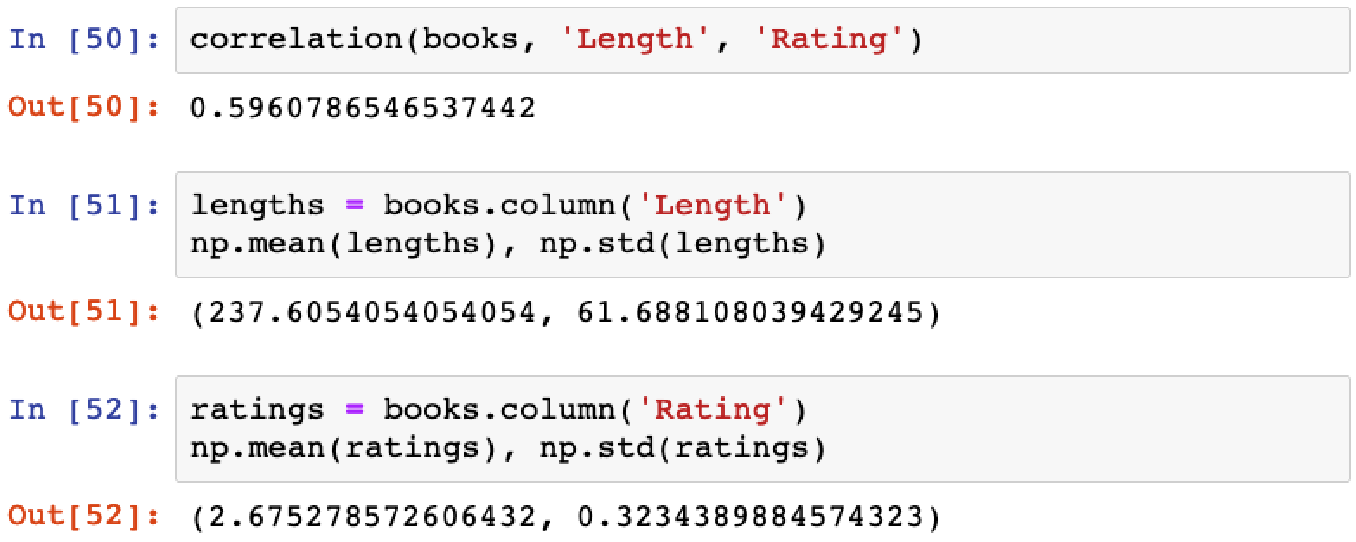
In [16]:
```
# TEST
median_length_choice in (1, 2, 3)
```

Out[16]:
```
True
```

```
In [17]: # HIDDEN TEST
         median_length_choice == 2

Out[17]: True
```

Zeynep finds review data for each book online, and adds it to her `books` table in a column called `Rating`. The values in this column are floating point numbers between 1 and 5, indicating the number of "stars" the book got, on average.

She computes the following quantities, which are reproduced for you in the cell below. Remember, you can't use the `books` table, only the quantities defined for you.



```
In [50]: correlation(books, 'Length', 'Rating')

Out[50]: 0.5960786546537442

In [51]: lengths = books.column('Length')
         np.mean(lengths), np.std(lengths)

Out[51]: (237.6054054054054, 61.688108039429245)

In [52]: ratings = books.column('Rating')
         np.mean(ratings), np.std(ratings)

Out[52]: (2.675278572606432, 0.3234389884574323)
```

```
In [18]: r = 0.596
         length_mean = 237.605
         length_sd = 61.7
         rating_mean = 2.675
         rating_sd = 0.323
```

**Question 2.4**

(2 points) Using linear regression, what would she predict for the average rating (in stars) of a book with 300 pages? Assign the variable `book_with_300_pages_avg_rating` to your answer.

*Hint*: try to avoid names like `correlation`, `slope`, `intercept` for your variables, since those are the names of functions we've defined for you at the top of the notebook.

```
BEGIN QUESTION
name: q2_4
manual: false
```

```
In [19]: book_with_300_pages_avg_rating = ((300 - length_mean)/length_sd * r) * r
         ating_sd + rating_mean # SOLUTION
         book_with_300_pages_avg_rating

Out[19]: 2.869676445056726
```

```
In [20]: # TEST
         all([type(book_with_300_pages_avg_rating) != type(...), type(book_with_3
         00_pages_avg_rating) != None])

Out[20]: True
```

```
In [21]: # HIDDEN TEST
         np.round(book_with_300_pages_avg_rating, 2) == 2.87

Out[21]: True
```

**Question 2.5**

(2 points) Using linear regression, what would she predict for the length of a book (in pages) with an average rating of 4 stars? Assign the variable `book_with_4_stars_length` to your answer.

*Hint*: try to avoid names like `correlation`, `slope`, `intercept` for your variables, since those are the names of functions we've defined for you at the top of the notebook.

```
BEGIN QUESTION
name: q2_5
manual: false
```

```
In [22]: book_with_4_stars_length = ((4 - rating_mean)/rating_sd * r) * length_sd
         + length_mean # SOLUTION
         book_with_4_stars_length

Out[22]: 388.4548142414861
```

```
In [23]: # TEST
         all([type(book_with_4_stars_length) != type(...), type(book_with_4_stars
         _length) != None])

Out[23]: True
```

```
In [24]: # HIDDEN TEST
         np.round(book_with_4_stars_length, 2) == 388.45

Out[24]: True
```

### Question 2.6

(2 points) Fill in the blank with the smallest possible value that's guaranteed to be correct. Assign the variable `book_rating_blank` to your answer.

Without knowing anything else about the distribution of ratings, we can guarantee that 75% of the ratings will be between 2.029 stars and ___ stars.

```
BEGIN QUESTION
name: q2_6
manual: false
```

```
In [25]: book_rating_blank = rating_mean + 2 * rating_sd # SOLUTION
         book_rating_blank
```

```
Out[25]: 3.3209999999999997
```

```
In [26]: # TEST
         all([type(book_rating_blank) != type(...), type(book_rating_blank) != No
         ne])
```

```
Out[26]: True
```

```
In [27]: # HIDDEN TEST
         np.round(book_rating_blank, 2) == 3.32
```

```
Out[27]: True
```

### Question 2.7

(4 points) Fill in the blank with the smallest value that's guaranteed to be correct. Assign the variable `range_of_accurate_book_predictions` to your answer.

When predicting average rating from book length, at least 93.75% of the predictions will be correct to within ___ stars of the true value.

*Hint*: $93.75\%$ is the same as $1 - \frac{1}{16}$.

```
BEGIN QUESTION
name: q2_7
manual: false
```

```
In [28]: range_of_accurate_book_predictions = 4 * np.sqrt(1 - r**2) * rating_sd #
         SOLUTION
         range_of_accurate_book_predictions

Out[28]: 1.037455887725353
```

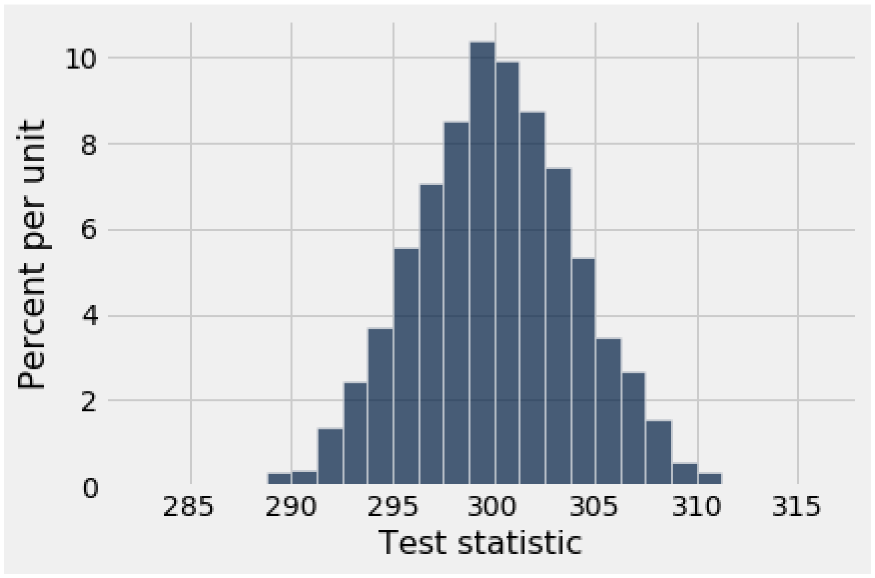
```
In [29]: # TEST
         all([type(range_of_accurate_book_predictions) != type(...), type(range_o
         f_accurate_book_predictions) != None])

Out[29]: True
```

```
In [30]: # HIDDEN TEST
         np.round(range_of_accurate_book_predictions, 2) == 1.04

Out[30]: True
```

### Question 2.8

(3 points) Zeynep uses her data to conduct a hypothesis test, but refuses to tell you any of the details. All she tells you is:

- Larger values of the test statistic favor the alternative hypothesis.
- Her $p$-value cutoff is 0.05.
- Her histogram of 5,000 simulated values of her test statistic under the null hypothesis looks like this (assume all values are shown in the histogram):



Based only on this information, which of the following must be true? Assign `mystery_hypothesis_test_conclusions` to an array with your numbered answer(s).

1. If the test statistic in her data is 310, then the data are more consistent with the alternative hypothesis than the null hypothesis.
2. If the test statistic in her data is 270, then we should conclude the data are more consistent with the null hypothesis than the alternative hypothesis.
3. If this was an A/B test where the test statistic was the difference in the means of two groups, and the test statistic in her data is 312, then she can conclude that there's a causal link between the groups and the value she's measuring for her test statistic.

```
BEGIN QUESTION
name: q2_8
manual: true
```

```
In [31]: mystery_hypothesis_test_conclusions = make_array(1, 2) # SOLUTION
```

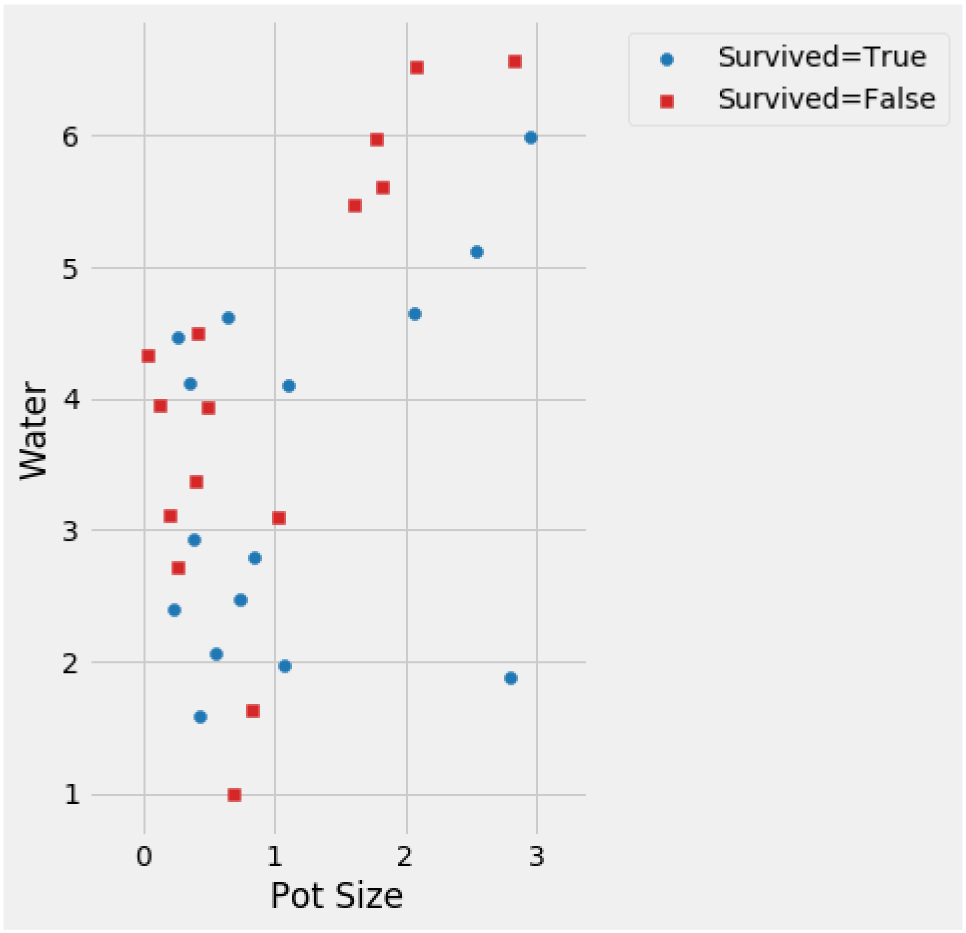
## Question 3: Gardening

Silla wants to get better at keeping his plants alive. So, he decides to try using a classifier to predict whether the plants into his garden will survive. He spends 8 weeks, from March 17 until May 12, collecting data for each plant on:

- **`Water`** : a float, the average amount of water he gave it per week (in ounces),
- **`Pot Size`** : a float, the size of the pot the plant was in (in gallons)
- **`Survived`** : a boolean, whether or not the plant was alive on May 12

He trains a $k$-nearest neighbor classifier on his data, and uses $k = 3$.

He also draws the following scatterplot, marking plants that survived with blue circles and plants that died with red squares. Assume all his plants are shown here:



Silla wants to use the classifier to make a predictions. For each of the following questions, you should answer one of:

- `True` , if the prediction is that the plant would survive
- `False` , if the prediction is that the plant would die,
- A string with an explanation if it isn't appropriate to use his classifier to answer the question.

If you answer `True` or `False` , then you should *not* include any explanation: it will not be graded.

For example, if a question asked "How tall will his tallest plant be?", then your answer should be `"The classifier can't predict how tall plants are, only whether or not they survive"` .

**Question 3.1**

(3 points) Suppose Silla had planted an extra plant in a 0.25-gallon pot that received 3.5 ounces of water per week. Would the classifier predict that this plant will survive?

```
BEGIN QUESTION
name: q3_1
manual: true
```

**SOLUTION:** False

**Question 3.2**

(3 points) Silla was taking care of his roommate's plant (which isn't shown in the graph above). It was in a 2.5-gallon pot, and he gave it 5 ounces of water per week. Would the classifier predict that this plant will survive?

```
BEGIN QUESTION
name: q3_2
manual: true
```

**SOLUTION:** True

### Question 3.3

(3 points) Silla was taking care of his other roommate's plant (which also isn't shown in the graph above). It was in a 6-gallon pot, and he gave it 6 ounces of water per week. Would the classifier predict that this plant will survive?

```
BEGIN QUESTION
name: q3_3
manual: true
```

**SOLUTION:** We don't have any training data for such large pots, so he shouldn't use this classifier.

### Question 3.4

(3 points) Silla graduates and moves from Berkeley to Miami, Florida. After moving in to his new apartment, he buys some new plants. He puts one in a 1-gallon pot, and gives it 2 ounces of water per week for 8 weeks (from June 1 to July 27). Would the classifier predict that this plant will survive?

```
BEGIN QUESTION
name: q3_4
manual: true
```

**SOLUTION:** Our training data is good for Berkeley in the spring, but we don't know how the weather in Florida in the summer changes the amount of water plants need. We also don't know if he got different kinds of plants.

## Question 4: Electricity

The table `electricity` contains data for 200 randomly sampled energy utilities from the US in 2017.

```
In [32]: electricity = Table.read_table('electricity2017_sample.csv').drop('Power
         in')
         electricity.show(3)
```

| Name | State | Type | Residential customers | Revenue | Power generated | Power bought | Summer demand | Winter demand |
|---|---|---|---|---|---|---|---|---|
| Village of Freeport - (NY) | NY | Municipal | 12928 | 34074 | 31785 | 261412 | 58.1 | 47.9 |
| Iowa Lakes Electric Coop | IA | Cooperative | 9247 | 63818.7 | 0 | 642428 | 99.5 | 108.3 |
| XOOM Energy Maine, LLC | NC | Retail Power Marketer | 1407 | 1352 | 0 | 14347 | 0 | 0 |

... (197 rows omitted)

It has the following columns:

- `Name` : a string, the name of the utility
- `State` : a string, the two-letter abbreviation for the state the utility operates in
- `Type` : a string, the type of utility
- `Residential customers` : an int, the number of residential customers the utility serves
- `Revenue` : a float, the total revenue for the utility in 2017 measured in thousands of dollars
- `Power generated` : a float, the amount of power the utility generated itself (in megawatt-hours)
- `Power bought` : a float, the amount of power the utility bought or exchanged from other utilities (in megawatt-hours)
- `Summer demand` : a float, representing peak demand in the summer (in megawatts)
- `Winter demand` : a float, representing peak demand in the winter (in megawatts)

### Question 4.1

(3 points) Janea wants to use this random sample to understand energy utilities more broadly. In particular, she wants to estimate the following quantities from this particular sample:

1. The maximum number of residential customers served by any energy utility in the US
2. The average (mean) demand in the summer across all energy utilities in the US
3. The median revenue of all energy utilities in Hawaii (*Hint*: the two-letter abbreviation for Hawaii is HI).
4. The slope of a linear prediction of winter demand from summer demand, across all energy utilities in the US

If she decides to use the bootstrap to construct a 95% confidence interval from this sample, which of the quantities above are good choices for this technique? Assign `good_electricity_bootstrap_candidates` to an array of your numbered answer(s).

```
BEGIN QUESTION
name: q4_1
manual: true
```

In [33]:
```
good_electricity_bootstrap_candidates = make_array(2, 4) # SOLUTION
```

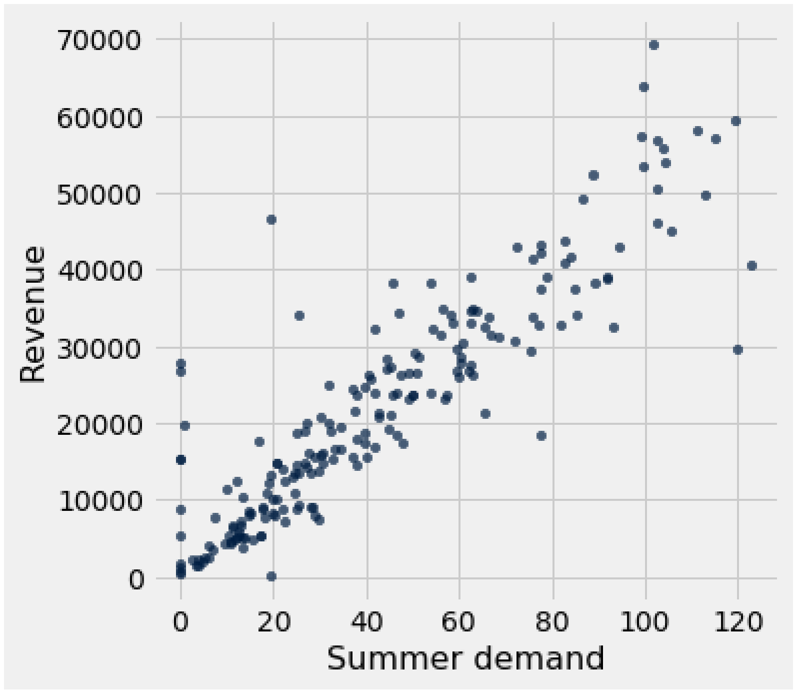
**Question 4.2**

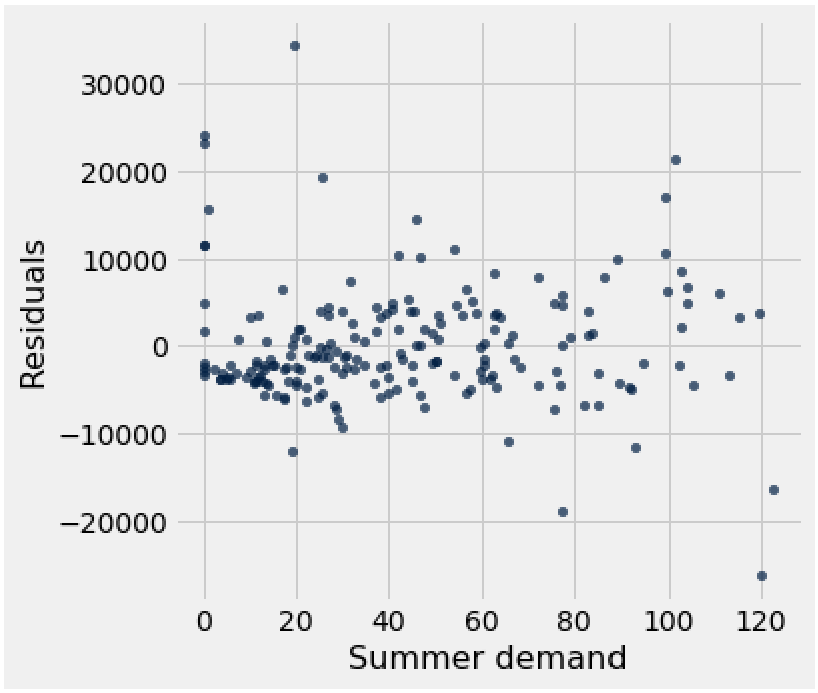
(2 points) Javier is on the board of a US energy utility, and wants to predict what his revenue will be based on summer demand. He isn't sure whether linear regression is a good fit: create a plot to help convince him that it is.

*Hint*: you shouldn't need more than 3-4 lines to solve this, and our solution took fewer than that.

```
BEGIN QUESTION
name: q4_2
manual: true
```

In [34]:
```
# Code to create a plot goes here.
# BEGIN SOLUTION
electricity.scatter('Summer demand', 'Revenue')
Table().with_columns("Summer demand", electricity.column("Summer demand"
), "Residuals",
                     residuals(electricity, "Summer demand", "Revenue"))
.scatter("Summer demand")
# END SOLUTION
```





### Question 4.3

(5 points) Javier's utility had 90 megawatts of demand this summer. He uses the bootstrap to construct a 95% confidence interval for the prediction of its revenue based on linear regression, and finds that the interval is $40.6 to \$45 million dollars (remember, the `Revenue` column is in thousands of dollars).

Which of the following must be true? Assign `revenue_demand_prediction_choices` to an array of your numbered answer(s).

1. 95% of utilities in the sample had revenue between $40.6 and \$45.0 million dollars.
2. In 95% of Javier's bootstrap samples, the height of the regression line at $x = 90$ was between 40600 and 45000 (that is, $40.6 and \$45 million dollars).
3. In the population of all US energy utilities, 95% of the utilities with 90 megawatts of summer demand have between $40.6 and \$45 million dollars of revenue.
4. There is a 95% chance that Javier's utility's revenue this year will be between $40.6 and \$45 million dollars.
5. When drawing a random sample of 200 US energy utilities from the population of all US energy utilities, there is a 95% chance that in the sample, the linear regression prediction for utilities with 90 megawatts of summer demand will be between $40.6 and \$45 million dollars.

```
BEGIN QUESTION
name: q4_3
manual: true
```

```
In [35]: revenue_demand_prediction_choices = make_array(2, 5) # SOLUTION
```

### Question 4.4

(4 points) Javier wants to estimate the difference between the average size (number of customers) of municipal utilities (owned by local governments) and cooperative utilities (owned by customers). In other words, he wants to compute the municipal average number of customers minus the cooperative average number of customers.

Help him by completing the function below.

For this question, the public tests check for correctness.

```
BEGIN QUESTION
name: q4_4
manual: false
```

```
In [36]: def compute_municipal_cooperative_difference(tbl):
             """
             Given a table with columns 'Type' and 'Residential customers',
             returns the difference between  the average size of municipal utilit
         ies
             and cooperative utilities.
             """
             # BEGIN SOLUTION
             grouped_tbl = tbl.group("Type", np.average)
             municipal_avg = grouped_tbl.where("Type", "Municipal").column("Resid
         ential customers average").item(0)
             cooperative_avg = grouped_tbl.where("Type", "Cooperative").column("R
         esidential customers average").item(0)
             return municipal_avg - cooperative_avg
             # END SOLUTION
```

```
In [37]: # TEST
         np.round(compute_municipal_cooperative_difference(electricity), 2) == -4
         284.59
```

```
Out[37]: True
```

```
In [38]: # TEST
         # Make sure you're not hardcoding; use the `tbl` argument in your functi
         on
         np.round(compute_municipal_cooperative_difference(electricity.take(np.ar
         ange(10))), 2) == -1288.10
```

```
Out[38]: True
```

**Question 4.5**

(4 points) Javier wonders: if his sample had been different, could he have gotten a different answer? Complete the function below, which computes 1,000 bootstrap samples of the statistic from the previous part and saves them in the array `bootstrap_utility_diff` which is returned.

```
BEGIN QUESTION
name: q4_5
manual: false
```

```
In [39]: def bootstrap_u_d():
             bootstrap_utility_diff = make_array() # SOLUTION
             for i in np.arange(1000): # SOLUTION
                 resample = electricity.sample() # SOLUTION
                 bootstrap_utility_diff = np.append(bootstrap_utility_diff, compu
         te_municipal_cooperative_difference(resample)) # SOLUTION
             return bootstrap_utility_diff

         # Don't edit this code below
         bootstrap_utility_differences = bootstrap_u_d()
```

```
In [40]: # TEST
         all([type(bootstrap_utility_differences) != type(...), type(bootstrap_ut
         ility_differences) != None])

Out[40]: True
```

```
In [41]: # HIDDEN TEST
         len(bootstrap_utility_differences) == 1000

Out[41]: True
```

**Question 4.6**

(3 points) Javier's favorite number is 92, so he wants to make a 92% confidence interval to estimate the value of the statistic above in the population. Use the `bootstrap_utility_differences` to assign `bootstrap_utility_left` and `bootstrap_utility_right` :

```
BEGIN QUESTION
name: q4_6
manual: false
```

```
In [42]: bootstrap_utility_left = percentile(4, bootstrap_utility_differences) #
          SOLUTION
         bootstrap_utility_right = percentile(96, bootstrap_utility_differences)
         # SOLUTION
         (bootstrap_utility_left, bootstrap_utility_right)

Out[42]: (-5715.00465416936, -3013.1817953231066)
```

```
In [43]: # TEST
         all([type(bootstrap_utility_left) != type(...), type(bootstrap_utility_l
         eft) != None])

Out[43]: True
```

```
In [44]: # TEST
         all([type(bootstrap_utility_right) != type(...), type(bootstrap_utility_
         right) != None])

Out[44]: True
```

In [45]:
```
# HIDDEN TEST
any([all([bootstrap_utility_left == percentile(4, bootstrap_utility_diff
erences),
     bootstrap_utility_right == percentile(96, bootstrap_utility_differe
nces)]),
    all([bootstrap_utility_left == np.percentile(bootstrap_utility_diffe
rences, 4),
     bootstrap_utility_right == np.percentile(bootstrap_utility_differen
ces, 96)])])
```

Out[45]:
```
True
```

## Question 5: The Office

The Data 8 instructors start arguing over one of their favorite TV shows, and decide to resolve their debates using data. They find data on how many words each character speaks in each episode of the show The Office in the table `office` .

In [46]:
```
office = Table.read_table('the_office.csv')
office.show(3)
```

| season | episode | overall_episode | speaker | num_words |
|---|---|---|---|---|
| 1 | 1 | 1 | TOTAL | 2861 |
| 1 | 1 | 1 | Michael | 1598 |
| 1 | 1 | 1 | Jim | 335 |

... (3077 rows omitted)

The table contains the following columns:

- **`season`** : an int, indicating which season of the show
- **`episode`** : an int, indicating which episode within that season the data is for
- **`overall_episode`** : an int, indicating the episode number within the entire show (for example, since the first season has 6 episodes, the first episode of the second season is the seventh episode overall)
- **`speaker`** : a string, the name of the character (or the string `TOTAL` , indicating the row contains the total number of words for the episode)
- **`num_words`** : an int, the number of words that character spoke in that episode

For example, the second row indicates that the character Michael said 1598 words in the first episode of the first season.

(4 points) In order to make line and scatter graphs of how much the characters speak over the course of the show, we'll need to write a function that takes an array of character names and returns a table with one row per overall episode number. It should have one column for the episode numbers, and a column for each character in the array, where the values are the total number of words spoken by that character in that episode.

For example, calling `get_character_episode_table(make_array('Jim', 'Dwight', 'Pam'))` should give you a table whose first few rows look like:

| overall_episode | Dwight | Jim | Pam |
|---:|---:|---:|---:|
| 1 | 296 | 335 | 295 |
| 2 | 185 | 366 | 113 |
| 3 | 782 | 342 | 218 |
| 4 | 609 | 588 | 236 |

**Question 5.1**

This means that Dwight spoke 296 words in the first episode, and Jim and Pam spoke 335 and 295 respectively.

(*Hint*: the predicate `are.contained_in` might be helpful.)

```
BEGIN QUESTION
name: q5_1
manual: false
```

In [47]:
```
def get_character_episode_table(character_names):
    """
    Takes an array of character names, and returns a table like the one
 shown above.
    """
    # BEGIN SOLUTION
    selected_characters = office.where('speaker', are.contained_in(chara
cter_names))
    return (selected_characters.select('overall_episode', 'speaker', 'nu
m_words')
            .pivot('speaker', 'overall_episode', 'num_words', sum))
    # END SOLUTION
```

In [48]:
```
# TEST
get_character_episode_table(make_array("Angela")) != None
```

Out[48]:
```
True
```

In [49]:
```
# HIDDEN TEST
get_character_episode_table(make_array("Angela", "Dwight")).take(np.aran
ge(5))
```

Out[49]:

| overall_episode | Angela | Dwight |
|---|---|---|
| 1 | 10 | 296 |
| 2 | 4 | 185 |
| 3 | 26 | 782 |
| 4 | 35 | 609 |
| 5 | 31 | 147 |

*Just for fun*: Use the function you just wrote to visualize the relationships between how much characters speak, and how that changes over the course of the show.

In [50]:
```
# This cell is for answering the "Just for fun" question above. Nothing
 in it will be graded.
```

Ramesh and Swupnil start arguing over Season 8's popularity, and find the following data to help support their claims:

In [51]:
```
office_viewers = Table.read_table('office_wikipedia.csv')
office_viewers.show(3)
```

| Overall number | Name | Writer | Millions of viewers |
|---|---|---|---|
| 1 | Pilot | Ricky Gervais & Stephen Merchant and Greg Daniels | 11.2 |
| 2 | Diversity Day | B. J. Novak | 6 |
| 3 | Health Care | Paul Lieberstein | 5.8 |

... (183 rows omitted)

The table has the following columns:

- **`Overall number`** : an int, indicating the episode number within the entire show (for example, since the first season has 6 episodes, the first episode of the second season is the seventh episode overall)
- **`Name`** : a string, the episode name
- **`Writer`** : a string, the person or people who wrote the script for that episode
- **`Millions of viewers`** : a float, the number of viewers, in millions, who watched that episode when it aired on live TV (on NBC).

*Just for fun*: Which episode had the most viewers and why?

```
In [52]: # This cell is for answering the "Just for fun" question above. Nothing
          in it will be graded.
```

## Question 5.2

(3 points) Using the two tables above ( `office_viewers` and `office` ), make a new table that has one row for each episode of the show, and three columns: **`season`** , **`overall_episode`** , and **`Millions of viewers`** . The columns may be in any order, but they must contain the correct values as described by their names.

```
BEGIN QUESTION
name: q5_2
manual: false
```

```
In [53]: # The overall_and_season table was helpful in our solution,
         # but you don't have to use it.
         overall_and_season = office.group(['overall_episode', 'season']).select(
         0, 1)
         office_season_viewers = overall_and_season.join("overall_episode", offic
         e_viewers, "Overall number").select(1, 0, 4) # SOLUTION
         office_season_viewers.show(3)
```

| season | overall_episode | Millions of viewers |
|---|---|---|
| 1 | 1 | 11.2 |
| 1 | 2 | 6 |
| 1 | 3 | 5.8 |

... (183 rows omitted)

```
In [54]: # TEST
         all([type(office_season_viewers) != type(...), type(office_season_viewer
         s) != None])

Out[54]: True
```

```
In [55]: # HIDDEN TEST
         set(office_season_viewers.labels) == set(['Millions of viewers', 'overal
         l_episode', 'season'])

Out[55]: True
```

```
In [56]: # HIDDEN TEST
         np.round(sum(office_season_viewers.take(np.arange(7)).column("Millions o
         f viewers")), 1) == 47.2

Out[56]: True
```

### Question 5.3

(2 points) Ramesh thinks that Season 8 is significantly less popular (measured by number of viewers) than any other season of the show. Swupnil disagrees. They decide on the following null and alternative hypotheses for their test:

**Null hypothesis**:

The average viewer count for Season 8 is like the average viewer count for the same number of episodes picked at random from the entire show.

**Alternative hypothesis**:

No, the average viewer count for Season 8 is lower.

Which hypothesis supports Swupnil's argument? Assign the variable `season_8_swupnil` to either `'null'` or `'alternative'`:

```
BEGIN QUESTION
name: q5_3
manual: false
```

```
In [57]: season_8_swupnil = 'null' # SOLUTION
```

```
In [58]: # TEST
         all([type(season_8_swupnil) != type(...), type(season_8_swupnil) != None
         ])

Out[58]: True
```

```
In [59]: # TEST
         # Make sure your answer is either 'null' or 'alternative'
         season_8_swupnil in ('null', 'alternative')

Out[59]: True
```

```
In [60]: # HIDDEN TEST
         season_8_swupnil == 'null'

Out[60]: True
```

### Question 5.4

(3 points) Based on the null and alternative hypothesis above, describe a test statistic they could use to conduct a hypothesis test:

```
BEGIN QUESTION
name: q5_4
manual: true
```

**SOLUTION:** Average viewer count of a random sample of all episodes with the same number of episodes as season 8.

### Question 5.5

(3 points) They decide on a $p$-value cutoff of $0.01$, and carry out the test using a correct test statistic.

They obtain a $p$-value of 0. Based only on the information provided to you, which must be true? Assign `season_8_test_conclusions` to an array with your numbered answer(s).

1. If the null hypothesis were true, the probability of observing a result that supports the alternative hypothesis is $0.01$.
2. If the alternative hypothesis were true, the probability of observing their test statistic is 1.
3. The data support the alternative hypothesis.
4. The data support the null hypothesis.

```
BEGIN QUESTION
name: q5_5
manual: true
```

In [61]:
```
season_8_test_conclusions = make_array(1, 3) # SOLUTION
```

## Question 6: Actors

Recall the `actors` table from lecture:

In [62]:
```
actors = Table.read_table('actors.csv').where('Number of Movies', are.ab
ove(10))
actors.show(3)
```

| Actor | Total Gross | Number of Movies | Average per Movie | #1 Movie | Gross |
|---|---|---|---|---|---|
| Harrison Ford | 4871.7 | 41 | 118.8 | Star Wars: The Force Awakens | 936.7 |
| Samuel L. Jackson | 4772.8 | 69 | 69.2 | The Avengers | 623.4 |
| Morgan Freeman | 4468.3 | 61 | 73.3 | The Dark Knight | 534.9 |

... (46 rows omitted)

You can find a description of the columns in Chapter 7 (https://www.inferentialthinking.com/chapters/07/Visualization.html) of the textbook. Note that just like in lecture and the textbook, we've removed Anthony Daniels since he's an outlier.

For this question, we'll focus on using the box office gross from the actor's top movie (that is, the `Gross` column) to predict the total box office receipt from all the actor's movies (that is, the `Total Gross` column).

Recall that when learning about linear regression, we used root mean square error (RMSE). Here's a function that computes it:

In [63]:
```
# You don't have to do anything in this cell: this is the same function
 that we
# defined in lecture, reproduced just so you can see it.
def rmse(predictions, actual_values):
    """
    Takes an array of predictions and an array of actual observed value
s, and returns
    the root mean squared error.
    """
    error = actual_values - predictions
    squared_error = error ** 2
    mean_squared_error = np.mean(squared_error)
    return np.sqrt(mean_squared_error)
```

### Question 6.1

(3 points) Suppose instead of using RMSE, we decide to use a *weighted error*. We want our prediction to be more accurate for actors who've been in fewer movies, and we don't care as much about actors who've been in lots of movies. So, we're going to compute the *weighted RMSE* for a prediction line, using this procedure:

1. Compute the squared error, just like before.
2. For each actor, *divide* the squared error by the weights (i.e., the number of movies the actor has been in). The result of this division will be called the weighted squared error for each actor.
3. Compute the average weighted squared error.
4. Take the square root.

Complete the function below to implement the procedure described here.

```
BEGIN QUESTION
name: q6_1
manual: false
```

In [64]:
```
def weighted_rmse(predictions, actual_values, weights):
    error = actual_values - predictions
    # BEGIN SOLUTION
    squared_error = error ** 2
    weighted_squared_error = squared_error / weights
    return np.sqrt(np.mean(weighted_squared_error))
    # END SOLUTION
```

In [65]:
```
# TEST
# If this errored, you need to make sure you're using the arguments (i.
e. weights) passed into the function,
# and not hardcoding actors.column('Number of Movies')
np.round(weighted_rmse(make_array(1, 3, 5), make_array(2, 3, 4), make_ar
ray(1, 2, 3)), 2) == 0.67
```

Out[65]:
```
True
```

In [66]:
```
# HIDDEN TEST
np.round(weighted_rmse(np.arange(10), np.arange(1, 11), np.arange(2, 12
)), 2) == 0.45
```

Out[66]:
```
True
```

### Question 6.2

(4 points) Complete the `weighted_prediction_error` function below, which takes in the slope and intercept for any line, and computes the weighted RMSE for predicting these actors' total box office gross using that line.

As a reminder, the line predicts total box office gross (the `Total Gross` column) from the box office gross from the actor's top movie (that is, the `Gross` column), and uses the number of movies the actor has been in (the `Number of Movies` column) for the weights in computing the weighted RMSE.

*Hint*: your solution should use the `weighted_rmse` function you defined above.

```
BEGIN QUESTION
name: q6_2
manual: false
```

In [67]:
```
def weighted_prediction_error(any_slope, any_intercept):
    # BEGIN SOLUTION
    predictions = any_slope * actors.column('Gross') + any_intercept
    err = weighted_rmse(predictions, actors.column('Total Gross'), actor
s.column('Number of Movies'))
    return err
    # END SOLUTION
```

In [68]:
```
# TEST
type(weighted_prediction_error(4, 5)) != type(...)
```

Out[68]:
```
True
```

In [69]:
```
# HIDDEN TEST
np.round(weighted_prediction_error(2, 4), 2) == 368.04
```

Out[69]:
```
True
```

### Question 6.3

(4 points) Find the slope and intercept of the line whose predictions have the smallest weighted RMSE.

```
BEGIN QUESTION
name: q6_3
manual: false
```

```
In [70]: # BEGIN SOLUTION
         params = minimize(weighted_prediction_error)
         # END SOLUTION
         best_weighted_slope = params.item(0) # SOLUTION
         best_weighted_intercept = params.item(1) # SOLUTION
```

```
In [71]: # TEST
         all([type(best_weighted_slope) != type(...), type(best_weighted_slope) !
         = None])

Out[71]: True
```

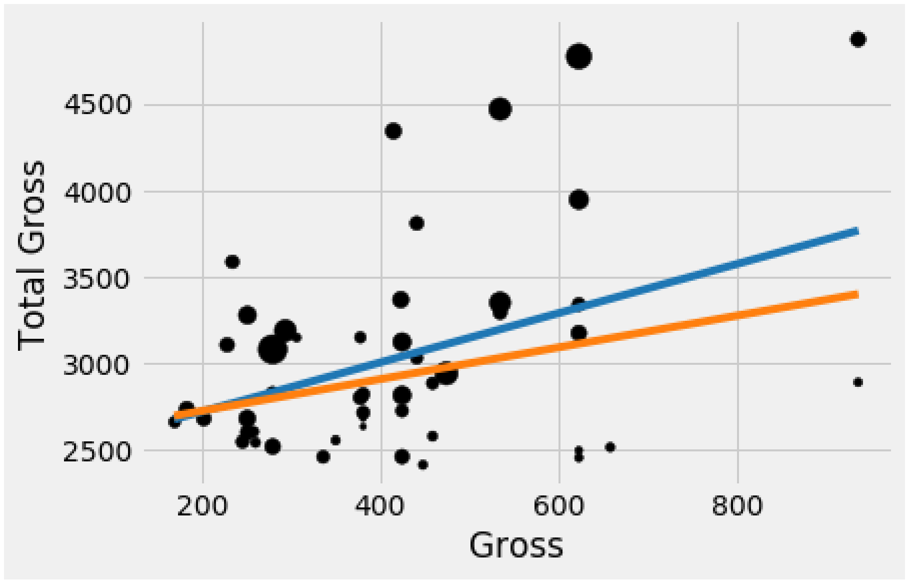
```
In [72]: # TEST
         all([type(best_weighted_intercept) != type(...), type(best_weighted_inte
         rcept) != None])

Out[72]: True
```

```
In [73]: # HIDDEN TEST
         all([np.round(best_weighted_slope, 3) == 0.917, np.round(best_weighted_i
         ntercept, 3) == 2541.847])

Out[73]: True
```

The code below plots the original linear regression line (blue) and the line with the smallest weighted RMSE (orange) using your answer from above. The size of each dot represents the number of movies that actor has been in.

If you see errors trying to run this cell, it might be because you redefined the `slope` and `intercept` functions from the beginning of this notebook. Try changing the variable names you use, re-running the cell at the top, and running this cell again.

In [74]:
```
# Do not change any of the code in this cell.

# You don't have to answer any questions here or understand how this cel
l works:
# it just shows you the result of your work.
x = actors.column('Gross')
y = actors.column('Total Gross')
weights = actors.column('Number of Movies')

x_plot = make_array(169, 937)
regression_line = x_plot * slope(actors, 'Gross', 'Total Gross') + inter
cept(actors, 'Gross', 'Total Gross')
weighted_line = x_plot * best_weighted_slope + best_weighted_intercept
plots.figure()
plots.scatter(x, y, s=(weights ** 2) / 30, c='black')
plots.plot(x_plot, regression_line, color='tab:blue')
plots.plot(x_plot, weighted_line, color='tab:orange')
plots.xlabel('Gross')
plots.ylabel('Total Gross');
```



## Question 7: US Counties and Food Deserts

A food desert (https://en.wikipedia.org/wiki/Food_desert) is an area that has limited access to nutritious food. In this question, we'll look at food access data for US counties from 2010 to 2015.

In [75]:
```
food_access = Table.read_table('food_access_2010_2015.csv')
food_access.show(3)
```

| county | state | population | housing_units | urban_pct | low_access_1 | low_access_10 | low_access |
|---|---|---|---|---|---|---|---|
| Abbeville | SC | 25417 | 9990 | 16.6667 | 21510 | 5176 | |
| Acadia | LA | 61773 | 22841 | 50 | 32874 | 625 | |
| Accomack | VA | 33164 | 13798 | 0 | 26847 | 727 | |

... (3134 rows omitted)

The `food_access` table contains one row for each county in the 50 states of the US.

It has the following columns:

- **`county`** : a string, the name of the county
- **`population`** : an int, the number of people living in that county
- **`state`** : a string, the two-letter abbreviation for the state the county is in
- **`housing_units`** : an int, the number of housing units available in the county
- **`urban_pct`** : a float between 0 and 100, the percentage of housing units that are urban in the county
- **`low_access_20`** : an int, the number of people in that county living more than twenty miles away from their nearest grocery store
- **`low_access_10`** : an int, the number of people in that county living more than ten miles away from their nearest grocery store (includes people living more than twenty miles away)
- **`low_access_1`** : an int, the number of people in that county living more than one mile away from their nearest grocery store (includes people living more than 10 and 20 miles away)
- **`carless_pct`** : a float between 0 and 100, the percentage of people in that county who don't have a car in their household

In this question, you'll also work with the `states` table:

In [76]:
```
states = Table.read_table('states.csv')
states.show(3)
```

| State | State Code | Region | Division |
|---|---|---|---|
| Alaska | AK | West | Pacific |
| Alabama | AL | South | East South Central |
| Arkansas | AR | South | West South Central |

... (48 rows omitted)

For each state, the table has:

- **`State`** : a string, the name of the state
- **`State Code`** : a string, the two-letter abbreviation for the state
- **`Region`** : a string, which region the state belongs to
- **`Division`** : a string, which division the state belongs to

If you're curious about what the regions and divisions are (and what the difference between them is), see this map from the US Census Bureau (https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf).

## Part 1: Table manipulation

For each of the following questions, write Python code that computes (or draws) the specified quantity, table, or graph.

### Question 7.1.1

(2 points) The number of people living more than 10 miles away from a grocery store in Alameda County, where UC Berkeley is.

```
BEGIN QUESTION
name: q7_1_1
manual: false
```

```
In [77]: num_people_more_than_10_miles_away_alameda_county = food_access.where("c
ounty", "Alameda").column("low_access_10").item(0) # SOLUTION
num_people_more_than_10_miles_away_alameda_county
```

```
Out[77]: 111
```

```
In [78]: # TEST
all([type(num_people_more_than_10_miles_away_alameda_county) != type(...
),
     type(num_people_more_than_10_miles_away_alameda_county) != None])
```

```
Out[78]: True
```

```
In [79]: # HIDDEN TEST
num_people_more_than_10_miles_away_alameda_county == 111
```

```
Out[79]: True
```

*Just for fun*: Same as the above question, but for any other county in California that you've lived in or spent time in. Do the results surprise you?

```
In [80]: # This cell is for answering the "Just for fun" question above. Nothing
          in it will be graded.
```

**Question 7.1.2**

(3 points) The county in California that has the most people living far away (>10 miles) from a grocery store. The abbreviation for California is CA.

```
BEGIN QUESTION
name: q7_1_2
manual: false
```

```
In [81]: # BEGIN SOLUTION
         ca = food_access.where("state", "CA").sort("low_access_10", descending=T
         rue)
         # END SOLUTION
         california_county_most_food_deserted = ca.column("county").item(0) # SOL
         UTION
         california_county_most_food_deserted

Out[81]: 'Fresno'
```

```
In [82]: # TEST
         all([type(california_county_most_food_deserted) != type(...), type(calif
         ornia_county_most_food_deserted) != None])

Out[82]: True
```

```
In [83]: # HIDDEN TEST
         california_county_most_food_deserted == 'Fresno'

Out[83]: True
```

**Question 7.1.3**

(6 points) A table `biggest_state_for_each_division` with the largest state (by population) in each **division** (not region). It should have **one row for each division**, and two columns: one with the name of the division, and one with the name (not the two letter abbreviation) of the most populated state in that division. The names of the columns don't matter, but they must be in that order (division column first, state column second).

For example, the most populated state in the Pacific division is California, so one of the rows in your table should have as its first item the string `Pacific` and as its second item the string `California` .

(*Hint*: you may find the `first` function helpful: we've defined it for you here.)

```
BEGIN QUESTION
name: q7_1_3
manual: false
```

In [84]:
```
def first(arr):
    return arr.item(0)

state_populations = food_access.select('state', 'population').group('sta
te', sum) # SOLUTION
# BEGIN SOLUTION
joined_tbl = state_populations.join('state', states, 'State Code')
# END SOLUTION
biggest_state_for_each_division = joined_tbl.sort(1, descending=True).gr
oup('Division', first).select(0, 3) # SOLUTION
biggest_state_for_each_division
```

Out[84]:

| Division | State first |
|---|---|
| East North Central | Illinois |
| East South Central | Tennessee |
| Middle Atlantic | New York |
| Mountain | Arizona |
| New England | Massachusetts |
| Pacific | California |
| South Atlantic | Florida |
| West North Central | Missouri |
| West South Central | Texas |

In [85]:
```
# TEST
all([type(state_populations) != type(...), type(state_populations) != No
ne])
```

Out[85]:
```
True
```

In [86]:
```
# TEST
all([type(biggest_state_for_each_division) != type(...), type(biggest_st
ate_for_each_division) != None])
```

Out[86]:
```
True
```

In [87]:
```
# HIDDEN TEST
examp = biggest_state_for_each_division.sort(0).take(2)
all([examp.column(0).item(0) == 'Middle Atlantic', examp.column(1).item(
0) == 'New York'])
```

Out[87]:
```
True
```

## Part 2: relationships between quantities

For the rest of this question, instead of looking at the number of people without access to a grocery store, we'll look at the percentage.

We'll use the following terms:

- *low access at 1 mile*: the percentage of people in a county whose nearest grocery store is at least 1 mile away
- *low access at 10 miles*: the percentage of people in a county whose nearest grocery store is at least 10 miles away

In [88]:
```
food_access_with_pcts = food_access.with_columns(
    'low_access_1_pct', food_access.column('low_access_1') / food_access
.column('population') * 100,
    'low_access_10_pct', food_access.column('low_access_10') / food_acce
ss.column('population') * 100,
    'low_access_20_pct', food_access.column('low_access_20') / food_acce
ss.column('population') * 100,
).drop('low_access_1', 'low_access_10', 'low_access_20')
food_access_with_pcts.show(3)
```

| county | state | population | housing_units | urban_pct | carless_pct | low_access_1_pct | low_acces |
|---|---|---|---|---|---|---|---|
| Abbeville | SC | 25417 | 9990 | 16.6667 | 8.5 | 84.6284 | |
| Acadia | LA | 61773 | 22841 | 50 | 8.9 | 53.2174 | |
| Accomack | VA | 33164 | 13798 | 0 | 9.7 | 80.9522 | |

... (3134 rows omitted)

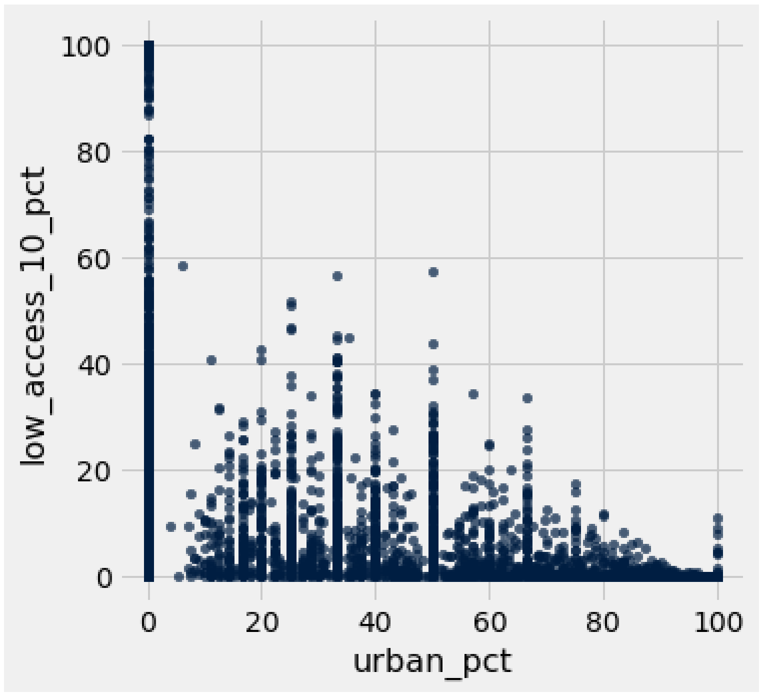
**Question 7.2.1**

(2 points) Create one plot that you'd use to answer the following question. You'll use your plot to answer the related multiple choice question below, but you do **not** have to answer the question directly, only create a plot.

Is there any association between the percentage of urban housing and low access at **10 miles** in US counties?

```
BEGIN QUESTION
name: q7_2_1
manual: true
```

In [89]:
```
# Code to generate your plot goes here
# BEGIN SOLUTION
food_access_with_pcts.scatter('urban_pct', 'low_access_10_pct')
# END SOLUTION
```



**Question 7.2.2**

(3 points) Based only on the data in the table (and the graph you created using that data), which must be true? Assign `low_access_10_urban_choices` to an array with your numbered answer(s).

1. Knowing the percentage of urban housing in a county cannot help us predict low access at 10 miles.
2. There is a strong linear association between percentage of urban housing and low access at 10 miles.
3. The distribution of low access at 10 miles is different between non-urban counties (<5% urban housing) and other counties.
4. For counties that are very urban (>80% urban housing), more than half of their population is within 10 miles of a grocery store.

```
BEGIN QUESTION
name: q7_2_2
manual: true
```

In [90]:
```
low_access_10_urban_choices = make_array(3, 4) # SOLUTION
```

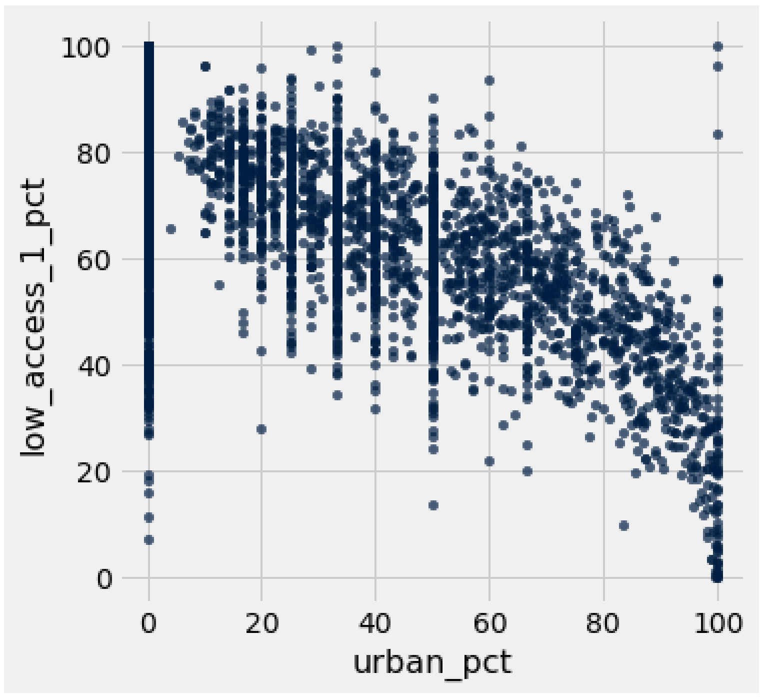
**Question 7.2.3**

(2 points) Create one plot that you'd use to answer the following question. You do not have to answer the question directly, only create a plot.

Is there any association between the percentage of urban housing and low access at **1 mile** in US counties?

```
BEGIN QUESTION
name: q7_2_3
manual: true
```

```
In [91]: # Code to generate your plot goes here
         # BEGIN SOLUTION
         food_access_with_pcts.scatter('urban_pct', 'low_access_1_pct')
         # END SOLUTION
```



### Question 7.2.4

(2 points) What is the correlation between percentage of urban housing and low access at 1 mile?

(*Hint*: You may find it helpful to use some of the functions defined in the "Useful functions" section near the top of this notebook).

```
BEGIN QUESTION
name: q7_2_4
manual: false
```

```
In [92]: correlation_urban_low1 = correlation(food_access_with_pcts, 'urban_pct',
         'low_access_1_pct') # SOLUTION
         correlation_urban_low1

Out[92]: -0.6871958281943565
```

```
In [93]: # TEST
         all([type(correlation_urban_low1) != type(...), type(correlation_urban_l
         ow1) != None])

Out[93]: True
```

```
In [94]: # HIDDEN TEST
         np.round(correlation_urban_low1, 3) == -0.687

Out[94]: True
```

## Question 8: Likes on Instagram

Natalia, an engineer at Instagram, conducts a randomized controlled experiment to evaluate whether social media anxiety is **reduced** when users can't see the number of likes shown on a post. When the next Instagram software update ships, she randomly assigns users in the city of Berkeley to two groups:

- **Group A**: 10,000 users who can no longer see the number of likes, and
- **Group B**: 8,000 users who can still see the number of likes.

One month after shipping the new update, she measures each user's happiness and general sentiment towards Instagram with a survey. She wants to test the claim that the treatment, **hiding the number of likes, increases users' happiness scores**.

She prepares a table `instagram_users` , containing 18,000 rows, one for each user in her experiment. She can't share the full dataset with you since it's proprietary, but she chooses 3 random rows to show you, just so you can see what it looks like:

```
In [95]: instagram_users = Table.read_table('instagram_sample.csv')
         instagram_users
```

Out[95]:

| sex | age | group | happiness | sentiment |
|---|---|---|---|---|
| Female | 15 | A | 72 | Positive |
| Female | 31 | B | 53 | Negative |
| Male | 27 | B | 60 | Neutral |

- **`sex`** : a string, the user's sex ( `Male` or `Female` )
- **`age`** : an int, the user's age
- **`group`** : a string, the user's group for the experiment ( `A` or `B` )
- **`happiness`** : an int, the user's happiness score from the survey (between 0-100)
- **`sentiment`** : a string, the user's sentiment from the survey ( `Positive` , `Neutral` , or `Negative` )

Natalia's null hypothesis is that there is no difference in average happiness between people who see the number of likes compared to those who don't, and that any difference observed in the sample is due to chance.

For her test statistic, Natalia decides to use the difference between the average happiness scores of Group A and Group B (that is, the average of Group A minus the average of Group B).

Help her come up with an alternative hypothesis:

**Question 8.1**

(3 points) State an alternative hypothesis that she should use for her test.

```
BEGIN QUESTION
name: q8_1
manual: true
```

**SOLUTION**: Average happiness is higher for people who don't see the number of likes.

**Question 8.2**

(6 points) Natalia asks her coworker for help simulating one value of the test statistic under the null hypothesis, but the code he gives her has some mistakes. Fix the code below so that it correctly computes the test statistic under the null hypothesis.

```
BEGIN QUESTION
name: q8_2
manual: true
```

In [96]:
```
# This code contains mistakes that you need to fix.

def compute_instagram_test_statistic():
    # Shuffle the data
    shuffled_groups = instagram_users.take('group').sample(with_replacem
ent = False)

    users = instagram_users.append('shuffled_groups', shuffled_groups.co
lumn(1))
    # Two averages
    mean_A = np.average(users.where('shuffled_groups', 'A').select('grou
p'))
    mean_B = np.average(users.where('shuffled_groups', 'B').select('grou
p'))
    # Test statistic
    return mean_A - mean_B
```

### Question 8.3

(4 points) In the question above, why do we shuffle the data? Choose all that apply. Assign `instagram_shuffling_reasons` to an array with your numbered answer(s).

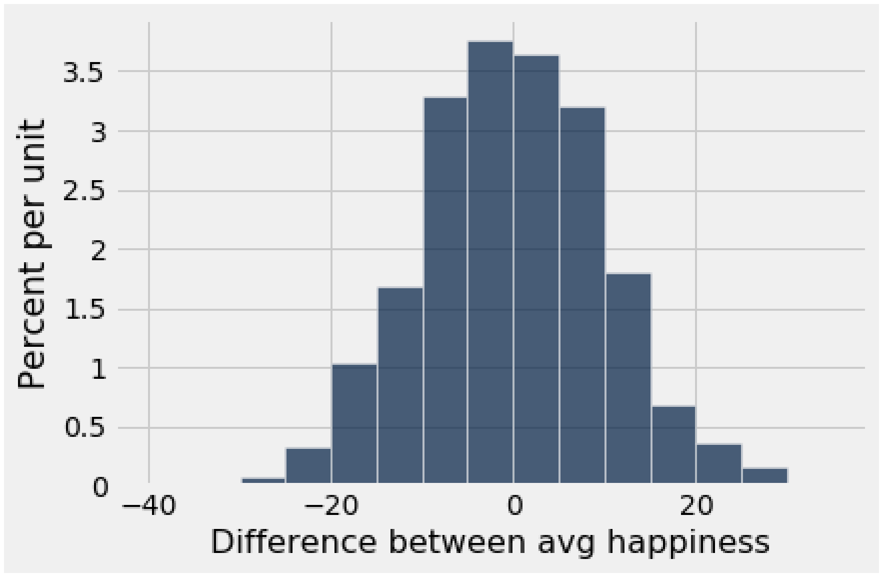
1. Under the null hypothesis, the label of being in group A or group B doesn't matter.
2. We want to randomize treatment and control to establish causation.
3. We want to simulate two groups of people whose expected happiness is identical under the null hypothesis.
4. We want to ensure that the users in the experiment are selected randomly.

```
BEGIN QUESTION
name: q8_3
manual: true
```

```
In [97]: instagram_shuffling_reasons = make_array(1, 3) # SOLUTION
         instagram_shuffling_reasons
```

```
Out[97]: array([1, 3])
```

Natalia fixes the code, and simulates 10,000 values of the test statistic under the null hypothesis. She shows them in the histogram below. You should assume that the histogram shows all of the simulated values.



### Question 8.4

(4 points) Based on this histogram, which of the following must be true? Assign `instagram_null_simulation_choices` to an array with your numbered answer(s).

1. Seeing the number of likes on a post has a positive effect on user happiness.
2. Seeing the number of likes on a post has no effect on user happiness.
3. If the $p$-value cutoff for the test is 0.01 and the observed test statistic is 35, then we should conclude the data are more consistent with the alternative hypothesis.
4. If the $p$-value cutoff for the test is 0.05 and the observed test statistic is 10, then we should conclude the data are more consistent with the null hypothesis.
5. Seeing the number of likes on a post has a negative effect on user happiness.

```
BEGIN QUESTION
name: q8_4
manual: true
```

```
In [98]: instagram_null_simulation_choices = make_array(3, 4) # SOLUTION
```

### Question 8.5

(5 points) Suppose Natalia tells you the observed test statistic was 42. Which conclusion(s) are the data consistent with? Assign `instagram_test_conclusion_choices` to an array with your numbered answer(s).

1. The difference in happiness scores between group A and group B is due to chance alone
2. The treatment has a negative association with happiness scores
3. The treatment has a positive effect on happiness scores
4. The treatment increases users' happiness scores by 42 points.
5. The data support the null hypothesis.
6. The data support the alternative hypothesis.
7. There isn't enough information to make a conclusion of any kind.

```
BEGIN QUESTION
name: q8_5
manual: true
```

```
In [99]: instagram_test_conclusion_choices = make_array(3, 6) # SOLUTION
```

For the remainder of this question, instead of testing for an increase in happiness scores, Natalia would like to test whether the treatment (i.e. hiding the number of likes) changes users' `sentiment` : that is, she's only interested in seeing whether the sentiment is significantly different.

Recall from the table above that `sentiment` is measured as either `Positive` , `Neutral` , or `Negative` .

She comes up with the following alternative hypothesis:

*The distribution of user sentiment between treatment (hiding the number of likes) and control (showing the number of likes) is different.*

Provide a null hypothesis and a test statistic she could use for her test.

### Question 8.6

(3 points) Null hypothesis:

```
BEGIN QUESTION
name: q8_6
manual: true
```

**SOLUTION:** There is no difference in the distribution of users' sentiment between treatment and control, and any observed difference in the sample is due to chance.

### Question 8.7

(3 points) What would be a valid test statistic to tell the two hypotheses above apart?

```
BEGIN QUESTION
name: q8_7
manual: true
```

**SOLUTION:** TVD

**Question 8.8**

(3 points) Natalia and her coworker are deciding whether to roll this feature out worldwide. Her coworker wants to collect more data, and develops a sentiment analysis algorithm that analyzes the content of users' posted images and captions to automatically determine their sentiment (positive, negative, or neutral).

Describe, in two sentences or less, any privacy concerns that you would have around the collection of this data.

*Hint*: this question is more open-ended than most of the rest of this exam: there isn't only one correct answer.

```
BEGIN QUESTION
name: q8_8
manual: true
```

**SOLUTION:** Example answer: Using people's personal images to train models without their direct consent is a violation of privacy.

## Question 9: Bread

### Part 1: Frozen slices

Ilin bakes too much bread, so she decides to slice it and put it in her freezer. She has three loaves of bread: one rye, one sourdough, and one multigrain. She slices each loaf: **the rye bread has 8 slices, the sourdough bread has 5 slices, and the multigrain bread has 12 slices**. In each loaf, two of the slices are "heels" (the slice at each end). She puts all the slices into a giant bag and mixes them up. Each day, she pulls out one slice at random, toasts it, and eats it.

**Question 9.1.1**

(2 points) What is the probability that her first slice is sourdough?

```
BEGIN QUESTION
name: q9_1_1
manual: false
```

In [100]:
```
slice_sourdough = 5 / 25 # SOLUTION
slice_sourdough
```

Out[100]:
```
0.2
```

```
In [101]: # TEST
          all([type(slice_sourdough) != type(...), type(slice_sourdough) != None])

Out[101]: True

In [102]: # HIDDEN TEST
          slice_sourdough == .2

Out[102]: True
```

### Question 9.1.2

(3 points) What is the probability that in her first two slices, she doesn't get any sourdough?

```
BEGIN QUESTION
name: q9_1_2
manual: false
```

```
In [103]: no_sourdough = (20/25) * (19/24) # SOLUTION
          no_sourdough

Out[103]: 0.6333333333333333

In [104]: # TEST
          all([type(no_sourdough) != type(...), type(no_sourdough) != None])

Out[104]: True

In [105]: # HIDDEN TEST
          np.round(no_sourdough, 2) == 0.63

Out[105]: True
```

### Question 9.1.3

(3 points) What is the probability that the first three slices are all heels?

```
BEGIN QUESTION
name: q9_1_3
manual: false
```

```
In [106]: three_heels = (6/25) * (5/24) * (4/23) # SOLUTION
          three_heels

Out[106]: 0.008695652173913044
```

```
In [107]: # TEST
          all([type(three_heels) != type(...), type(three_heels) != None])

Out[107]: True
```

```
In [108]: # HIDDEN TEST
          np.round(three_heels, 3) == 0.009

Out[108]: True
```

**Question 9.1.4**

(3 points) The first slice she pulls out is a heel. What is the probability that it is sourdough?

```
BEGIN QUESTION
name: q9_1_4
manual: false
```

```
In [109]: sourdough_heel = 1/3 # SOLUTION
          sourdough_heel

Out[109]: 0.3333333333333333
```

```
In [110]: # TEST
          all([type(sourdough_heel) != type(...), type(sourdough_heel) != None])

Out[110]: True
```

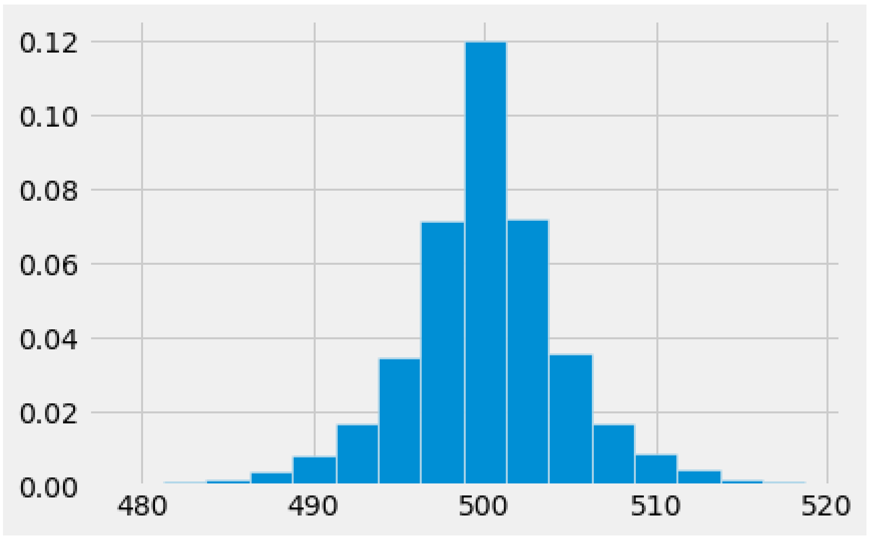
```
In [111]: # HIDDEN TEST
          np.round(sourdough_heel, 2) == 0.33

Out[111]: True
```

## Part 2: Quality control

Jin is in charge of quality control at a very large bakery that makes tens of thousands of loaves of bread each day. Each loaf is advertised as weighing 500 grams, but the exact weight varies a little bit from loaf to loaf.

Jin has had complaints from customers that their bread weighed less than 500 grams. He speaks with the bakers, who assure him that the average loaf weighs 500 grams, and that the standard deviation of the loaf weights is 5 grams. They show him the following histogram of distribution of weights that the loaves are supposed to follow.

According to the bakers, the distribution of weights in the population of loaves is as follows:



**Question 9.2.1**

(4 points) To verify this, he plans to hire 10,000 auditors to come into the bakery, and each one will randomly sample 100 loaves and weigh them. Each auditor will then tell Jin the average weight of the 100 loaves that they weighed.

Assuming the population distribution information above is correct, about how many of the 10,000 auditors will find an average weight below 499 grams? Your response should be a number between 0 and 10,000.

*Hint*: you should compute your answer using arithmetic from the information given, not using a simulation.

```
BEGIN QUESTION
name: q9_2_1
manual: false
```

In [112]:
```
# BEGIN SOLUTION
sds_from_mean = (499-500) / (5/10) # 2 SD's away from the sample mean --
95%
# END SOLUTION
auditors_avg_below_499 = (.05 * 10000) / 2 # SOLUTION
auditors_avg_below_499
```

Out[112]:
```
250.0
```

In [113]:
```
# TEST
all([type(auditors_avg_below_499) != type(...), type(auditors_avg_below_
499) != None])
```

Out[113]:
```
True
```

```
In [114]: # HIDDEN TEST
          any([int(np.round(auditors_avg_below_499)) == 250, np.round(auditors_avg
          _below_499) == 228])

Out[114]: True
```

**Question 9.2.2**

(4 points) When Jin discusses this idea with his peers, they tell him it's much too expensive. Instead, they suggest hiring one auditor, and asking the auditor to collect a slightly larger random sample of 225 loaves.

Assuming the population distribution information above is correct, approximately what is this probability that the auditor will find an average weight below 499 grams? Your response should be a number between 0 and 1.

You should compute your answer using arithmetic from the information given, not using a simulation.

```
BEGIN QUESTION
name: q9_2_2
manual: false
```

```
In [115]: # BEGIN SOLUTION
          sds_from_mean_prob = (499-500) / (5/15) # 3 SD's away from the sample me
          an -- 99.73%
          # END SOLUTION
          prob_auditors_avg_below_499 = .0027 / 2 # SOLUTION
          prob_auditors_avg_below_499

Out[115]: 0.00135
```

```
In [116]: # TEST
          all([type(prob_auditors_avg_below_499) != type(...), type(prob_auditors_
          avg_below_499) != None])

Out[116]: True
```

```
In [117]: # TEST
          # If you are failing this test, you should make sure your answer is a pr
          obability between 0 and 1.
          0 <= prob_auditors_avg_below_499 <= 1

Out[117]: True
```

```
In [118]: # HIDDEN TEST
          any([np.round(prob_auditors_avg_below_499, 4) == 0.0014, np.round(prob_a
          uditors_avg_below_499, 4) == 0.0015,
               np.round(prob_auditors_avg_below_499, 5) == 0.00135])

Out[118]: True
```

## 2. Submission

Once you're finished, select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org (https://okpy.org/) and flag the correct version. To do so, go to the website, click on this assignment, and find the version you would like to have graded. There should be an option to flag that submission for grading!

In [119]:
```
_ = ok.submit()
```

```
----------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-119-cc46ca874451> in <module>
----> 1 _ = ok.submit()

NameError: name 'ok' is not defined
```

In [120]:
```
# For your convenience, you can run this cell to run all the tests at on
ce!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')
and len(q) <= 10]
print("Finished running all tests.")
```

```
Running all tests...
Finished running all tests.
```