project1

March 10, 2016

1 Project 1 - California Water Usage

Welcome to the first project in Data 8! We will be exploring possible connections between water usage, geography, and income in California. The water data for this project was procured from the California State Water Resources Control Board and curated by the Pacific Institute. The map data includes US topography, California counties, and ZIP codes.

The dataset on income comes from the IRS (documentation). We have identified some interesting columns in the dataset, but a full description of all the columns (and a definition of the population in the dataset and some interesting anonymization procedures they used) is available in this description.

As usual, **run the cell below** to prepare the automatic tests. **Passing the automatic tests does not guarantee full credit on any question.** The tests are provided to help catch some common errors, but it is your responsibility to answer the questions correctly.

In [2]:
```
# Run this cell, 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')

# These lines load the tests.
from client.api.assignment import load_assignment
project1 = load_assignment('project1.ok')
```

```
=====================================================================
Assignment: Project 1
OK, version v1.5.1
=====================================================================
```

First, load the data. Loading may take some time.

In [3]:
```
# Run this cell, but please don't change it.

districts = Map.read_geojson('water_districts.geojson')
zips = Map.read_geojson('ca_zips.geojson.gz')
usage_raw = Table.read_table('water_usage.csv', dtype={'pwsid': str})
income_raw = Table.read_table('ca_income_by_zip.csv', dtype={'ZIP': str}).drop(['STATEFIPS', 'S
wd_vs_zip = Table.read_table('wd_vs_zip.csv', dtype={'PWSID': str, 'ZIP': str}).set_format([2,
```

2 Part 0: Maps

The `districts` and `zips` data sets are `Map` objects. Documentation on mapping in the `datascience` package can be found at data8.org/datascience/maps.html. To view a map of California's water districts, run the cell below. Click on a district to see its description.

```
In [4]: districts.format(width=400, height=200)
```

```
Out[4]: <datascience.maps.Map at 0x7f6ab55ab630>
```

A `Map` is a collection of regions and other features such as points and markers, each of which has a **string** `id` and various properties. You can view the features of the `districts` map as a table using `Table.from_records`.

```
In [5]: district_table = Table.from_records(districts.features)
        district_table.show(3)
```

```
<IPython.core.display.HTML object>
```

To display a `Map` containing only two features from the `district_table`, call `Map` on a list containing those two features from the `feature` column.

Question 0.0 Draw a map of the Alameda County Water District (row 0) and the East Bay Municipal Utilities District (row 2).

```
In [6]: alameda = district_table.column("feature").item(0)
        eastbay = district_table.column("feature").item(2)
        alameda_and_east_bay = np.append(alameda,eastbay)
        # district_table.select([1]).item(0)
        Map(alameda_and_east_bay, height=300, width=300)
```

```
Out[6]: <datascience.maps.Map at 0x7f69ed278eb8>
```

```
In [7]: _ = project1.grade('q00')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Hint: If scrolling becomes slow on your computer, you can clear maps for the cells above by running `Cell > All Output > Clear` from the `Cell` menu.

3 Part 1: California Income

Let's look at the `income_raw` table.

```
In [8]: income_raw
```

```
Out[8]: ZIP   | N1    | MARS1 | MARS2 | MARS4 | PREP  | N2    | NUMDEP | A00100 | N02650 | A02650 | N00
        90001 | 13100 | 6900  | 1890  | 4270  | 10740 | 29670 | 15200  | 181693 | 13100  | 184344 | 102
        90001 | 5900  | 1700  | 1970  | 2210  | 4960  | 17550 | 9690   | 203628 | 5900   | 204512 | 561
        90001 | 1480  | 330   | 760   | 390   | 1240  | 4710  | 2470   | 89065  | 1480   | 89344  | 144
```

```
90001 | 330   | 50    | 210   | 70    | 290   | 1100  | 560   | 28395  | 330   | 28555  | 320
90001 | 160   | 30    | 100   | 40    | 130   | 510   | 250   | 24676  | 160   | 25017  | 150
90001 | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0      | 0     | 0      | 0
90002 | 12150 | 6330  | 1460  | 4330  | 9580  | 27240 | 14070 | 167261 | 12150 | 170095 | 944
90002 | 5030  | 1510  | 1490  | 1980  | 4120  | 14410 | 7890  | 173280 | 5030  | 174335 | 476
90002 | 1320  | 300   | 600   | 400   | 1060  | 4090  | 2180  | 78559  | 1320  | 78871  | 127
90002 | 340   | 90    | 190   | 90    | 270   | 1060  | 530   | 28502  | 340   | 28558  | 320
... (8888 rows omitted)
```

Some observations:

1. The table contains several numerical columns and a column for the ZIP code.
2. For each ZIP code, there are 6 rows. Each row for a ZIP code has data from tax returns in one income bracket – a group of people who make between some income and some other income.

3. According to the IRS documentation, all the numerical columns are totals – either total numbers of returns that fall into various categories, or total amounts of money (in thousands of dollars) from returns in those categories. For example, the column `'N02650'` is the number of returns that included a total income amount, and `'A02650'` is the total amount of total income (in thousands of dollars) from those returns.

Question 1.0. Since we don't care about income brackets, but we do care about totals per ZIP code, let's group together our income data by ZIP code. Assign the name `income_by_zipcode` to a table with just one row per ZIP code. When you group according to ZIP code, the remaining columns should be summed. In other words, for any other column such as `'N02650'`, the value of `'N02650'` in a row corresponding to ZIP code 90210 (for example) should be the sum of the values of `'N02650'` in the 6 rows of `income_raw` corresponding to ZIP code 90210.

```
In [9]: income_by_zipcode= income_raw.group("ZIP", sum)
        income_by_zipcode
```

```
Out[9]: ZIP   | N1 sum | MARS1 sum | MARS2 sum | MARS4 sum | PREP sum | N2 sum | NUMDEP sum | A00100 su
        90001 | 20970  | 9010      | 4930      | 6980      | 17360    | 53540  | 28170      | 527457
        90002 | 18960  | 8230      | 3830      | 6800      | 15120    | 47200  | 24850      | 462823
        90003 | 26180  | 11310     | 5130      | 9640      | 20570    | 64470  | 33760      | 612733
        90004 | 27360  | 15330     | 7000      | 4670      | 20260    | 51180  | 17800      | 1.61777e+
        90005 | 15430  | 8550      | 3870      | 2830      | 11210    | 29910  | 11130      | 707020
        90006 | 22630  | 11470     | 5400      | 5630      | 17840    | 47590  | 20210      | 563530
        90007 | 11710  | 6350      | 2270      | 3020      | 8310     | 23380  | 9950       | 311779
        90008 | 14710  | 8060      | 2310      | 4110      | 9990     | 27000  | 10310      | 662036
        90010 | 2210   | 1270      | 690       | 210       | 1760     | 3790   | 960        | 314333
        90011 | 36670  | 15540     | 8600      | 12390     | 30240    | 95640  | 51260      | 857731
        ... (1473 rows omitted)
```

```
In [10]: _ = project1.grade('q10')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 1.1. Relabel the columns in `income_by_zipcode` to match the labels in `income_raw`; you probably modified all the names slightly in the previous question.

Hint: Inspect `income_raw.labels` and `income_by_zipcode.labels` to find the differences you need to change.

Hint 2: Since there are many columns, it will be easier to relabel each of them by using a `for` statement. See Section 3.2 of the textbook for details.

Hint 3: You can use the `replace` method of a string to remove excess content. See lab02 for examples.

Hint 4: To create a new table from an existing table with one label replaced, use `relabeled`. To **change** a label in an existing table permanently, use `relabel`. Both methods take two arguments: the old label and the new label. You can solve this problem with either one, but `relabel` is simpler.

```
In [11]: #another method would be something with length: len(i)-3
         #a = "Income_sum"
         #a[:-4]
         #for i in np.arange(len(income_by_zipcode)):
             #income_by_zipcode[i].replace("sum", " ")
         #income_by_zipcode

         column_labels = income_by_zipcode.labels
         for i in column_labels:
             r=i.replace(' sum','')
             income_by_zipcode.relabel(i, r)
         income_by_zipcode
```

```
Out[11]: ZIP   | N1    | MARS1 | MARS2 | MARS4 | PREP  | N2    | NUMDEP | A00100       | N02650 | A02650
         90001 | 20970 | 9010  | 4930  | 6980  | 17360 | 53540 | 28170  | 527457       | 20970  | 531772
         90002 | 18960 | 8230  | 3830  | 6800  | 15120 | 47200 | 24850  | 462823       | 18960  | 467128
         90003 | 26180 | 11310 | 5130  | 9640  | 20570 | 64470 | 33760  | 612733       | 26180  | 618848
         90004 | 27360 | 15330 | 7000  | 4670  | 20260 | 51180 | 17800  | 1.61777e+06 | 27360  | 1.64943
         90005 | 15430 | 8550  | 3870  | 2830  | 11210 | 29910 | 11130  | 707020       | 15430  | 717290
         90006 | 22630 | 11470 | 5400  | 5630  | 17840 | 47590 | 20210  | 563530       | 22630  | 571157
         90007 | 11710 | 6350  | 2270  | 3020  | 8310  | 23380 | 9950   | 311779       | 11710  | 315581
         90008 | 14710 | 8060  | 2310  | 4110  | 9990  | 27000 | 10310  | 662036       | 14710  | 668523
         90010 | 2210  | 1270  | 690   | 210   | 1760  | 3790  | 960    | 314333       | 2210   | 320471
         90011 | 36670 | 15540 | 8600  | 12390 | 30240 | 95640 | 51260  | 857731       | 36670  | 864961
         ... (1473 rows omitted)
```

```
In [12]: _ = project1.grade('q11')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 1.2. Create a table called `income` with one row per ZIP code and the following columns.

1. A `ZIP` column with the same contents as `'ZIP'` from `income_by_zipcode`.
2. A `returns` column containing the total number of tax returns that include a total income amount (column `'N02650'` from `income_by_zipcode`).
3. A `total` column containing the total income in all tax returns in thousands of dollars (column `'A02650'` from `income_by_zipcode`).

4. A `farmers` column containing the number of farmer returns (column `'SCHF'` from `income_by_zipcode`).

```
In [13]: income = Table().with_columns(['ZIP', income_by_zipcode.column('ZIP'),
                                        'returns', income_by_zipcode.column('N02650'),
                                        'total', income_by_zipcode.column('A02650'),
                                        'farmers', income_by_zipcode.column('SCHF')])
         income.set_format('total', NumberFormatter(0)).show(5)
```

```
<IPython.core.display.HTML object>
```

```
In [14]: _ = project1.grade('q12')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 1.3. What is the average total income reported on all California tax returns that include a total income amount? **Express the answer in dollars as an `int` rounded to the nearest dollar.**

```
In [15]: y = income.column('total')
         x = np.sum(y)/(income.num_rows)
         np.round(x) * 1000
```

```
Out[15]: 828284000.0
```

Question 1.4. All ZIP codes with less than 100 returns (or some other special conditions) are grouped together into one ZIP code with a special code. Remove the row for that ZIP code from the `income` table. Hint: This ZIP code value has far more returns than any of the other ZIP codes.

Hint: To **remove** a row in the `income` table using `where`, assign `income` to the smaller table using the following expression structure:

```
income = income.where(...)
```

Hint 2: Each ZIP code is represented as a string, not an `int`.

```
In [16]: income = income.where(income.column('ZIP') != '99999')
         income
```

```
Out[16]: ZIP   | returns | total      | farmers
         90001 | 20970   | 531,772    | 0
         90002 | 18960   | 467,128    | 0
         90003 | 26180   | 618,848    | 0
         90004 | 27360   | 1,649,431  | 0
         90005 | 15430   | 717,290    | 0
         90006 | 22630   | 571,157    | 0
         90007 | 11710   | 315,581    | 0
         90008 | 14710   | 668,523    | 0
         90010 | 2210    | 320,471    | 0
         90011 | 36670   | 864,961    | 0
         ... (1472 rows omitted)
```

```
In [17]: _ = project1.grade('q14')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 1.5. Among the tax returns in California for ZIP codes represented in the `incomes` table, is there an association between income and living in a ZIP code with a higher-than-average proportion of farmers?

Answer the question by comparing the average incomes for two groups of tax returns: those in ZIP codes with a greater-than-average proportion of farmers and those in ZIP codes with a less-than-average (or average) proportion. Make sure both of these values are displayed (preferably in a table). Then, describe your findings.

```
In [18]: # Build and display a table with two rows:
         #   1) incomes of returns in ZIP codes with a greater-than-average proportion of farmers
         #   2) incomes of returns in other ZIP codes
         farm_proportion = income.column('farmers')/income.column('returns')
         income_return = income.with_column('greater than average',
                                            farm_proportion > np.mean(farm_proportion))
         income_return.with_column('average', 1000 * income.column('total')/income.column('returns'))\
         .select(['greater than average', 'average']).group('greater than average',np.mean)

Out[18]: greater than average | average mean
         False                | 79038.3
         True                 | 60225.7
```

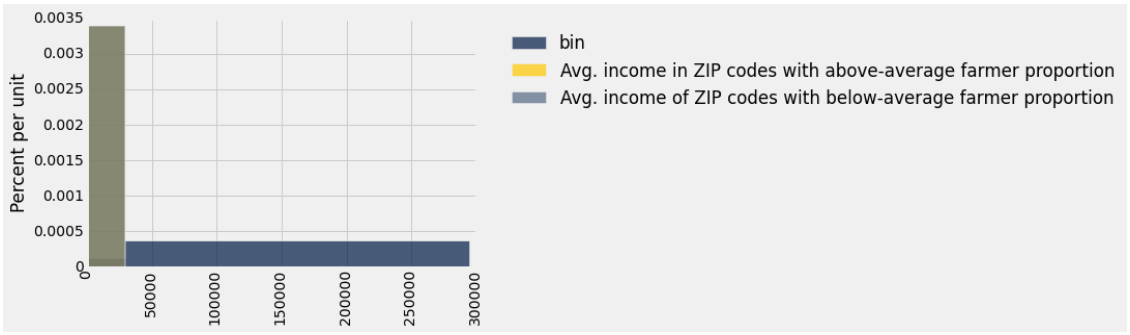
There is an association between income and living in a ZIP code with a higher than average proportion of farmers. Areas with higher than average proportion of farmers had lower tax returns.

Question 1.6. Investigate the same question by comparing two histograms: the average incomes of ZIP codes that have above-average vs below-average proportions of farmers. Quantify and describe the difference in the standard deviations of average incomes for the two kinds of ZIP codes.

```
In [19]: # You do not need to change this cell; just look at the chart it generates.

         # Note: there was provided code here to draw the histogram above, but it
         # did not work for everyone, so it has been commented out below.
         # If you don't see a chart above, it's here: https://i.imgur.com/jicA2to.png

         bins = np.arange(20000, 300000, 5000)
         avg_income = 1000 * income.column('total')/income.column('returns')
         binned_incomes=Table()\
              .with_column('income', avg_income)\
              .where(farm_proportion > np.mean(farm_proportion))\
              .bin(bins=bins).relabeled(1, 'Avg. income in ZIP codes with above-average farmer proportio
              .join('bin', Table().with_column('income', avg_income)\
                  .where(farm_proportion <= np.mean(farm_proportion))\
                  .bin(bins=bins).relabeled(1, 'Avg. income of ZIP codes with below-average farmer propo
         binned_incomes.hist()
```




```
In [37]: # Compute and compare the spread of both distributions
         #note to self: need to calculate SD of both distributions
         std_data_1= Table().with_column('above average income',avg_income).where(farm_proportion
                                                                    > np.mean(farm_proportion))
         std_above_avg= np.std(std_data_1.column(0))
         std_data_2=Table().with_column('below average income', avg_income).where(farm_proportion
                                                                    <= np.mean(farm_proportion))
         std_below_avg=np.std(std_data_2.column(0))
         std_above_avg, std_below_avg
```

```
Out[37]: (26121.687236432997, 78380.888351132875)
```

The standard deviation for average incomes in ZIP codes with above-average farmer proportion is smaller than the standard deviation for average incomes in ZIP codes with below-average farmer proportion. This means that that spread for the below-average farmer proportion ZIP code average incomes is greater than the average income in ZIP codes with above-average farmer proportions. These numbers in the below-average region are more spread out. The difference in spreads between the two distributions implies that there is an association.

ZIP codes cover all the land in California and do not overlap. Here's a map of all of them.

Question 1.7. Among the ZIP codes represented in the `incomes` table, is there an association between high average income and some aspect of the ZIP code's location? If so, describe one aspect of the location that is clearly associated with high income.

Answer the question by drawing a map of all ZIP codes that have an average income above 100,000 dollars. Then, describe an association that you observe.

In order to create a map of certain ZIP codes, you need to - Construct a table containing only the ZIP codes of interest, called `high_average_zips`, - Join `high_average_zips` with the `zip_features` table to find the region for each ZIP code of interest, - Call `Map(...)` on the column of features (provided).

```
In [21]: # Write code to draw a map of only the high-income ZIP codes
         zip_features = Table.from_records(zips.features)
         high_average_zips = income.where(1000 * income.column('total')/income.column('returns')
                                          > 100000).select([0])
         high_zips_with_region = zip_features.join("ZIP",high_average_zips)
         Map(list(high_zips_with_region.column('feature')), width=400, height=300)
```

```
Out[21]: <datascience.maps.Map at 0x7f69d7f97a58>
```

The map of only-income ZIP codes shows there is an association between location and high-income ZIP codes. The majority of high-income ZIP codes are concentrated in the San Francisco Bay Area.

```
In [22]: _ = project1.grade('q17')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

4 Part 2: Water Usage

We will now investigate water usage in California. The `usage` table contains three columns:

- `PWSID`: The Public Water Supply Identifier of the district
- `Population`: Estimate of average population served in 2015
- `Water`: Average residential water use (gallons per person per day) in 2014-2015

```
In [23]: # Run this cell to create the usage table

         usage_raw.set_format(4, NumberFormatter)
         max_pop = usage_raw.select([0, 'population']).group(0, max).relabeled(1, 'Population')
         avg_water = usage_raw.select([0, 'res_gpcd']).group(0, np.mean).relabeled(1, 'Water')
         usage = max_pop.join('pwsid', avg_water).relabeled(0, 'PWSID')
         usage

Out[23]: PWSID   | Population | Water
         0110001 | 340000     | 70.7
         0110003 | 57450      | 90.2727
         0110005 | 1390000    | 76
         0110006 | 151037     | 57.1818
         0110008 | 73067      | 96.6364
         0110009 | 79547      | 68.6364
         0110011 | 31994      | 85.8182
         0310003 | 23347      | 82.8182
         0410002 | 101447     | 142
         0410005 | 11208      | 88.8182
         ... (401 rows omitted)
```

Question 2.1. Draw a map of the water districts, colored by the per capita water usage in each district.

Use the `districts.color(...)` method to generate the map. It takes as its first argument a two-column table with one row per district that has the district `PWSID` as its first column. The label of the second column is used in the legend of the map, and the values are used to color each region.

```
In [24]: per_capita_usage = Table().with_columns(["PWSID", usage.column('PWSID'),
                                                  'V', usage.column('Water')])
         districts.color(per_capita_usage, key_on='feature.properties.PWSID')

Out[24]: <datascience.maps.Map at 0x7f69e439f128>

In [25]: _ = project1.grade('q21')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------------
```

```
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2.2. Based on the map above, which part of California appears to use more water per person, the San Francisco area or the Los Angeles area?

Based on the map above, the Los Angeles area appears to use more water per person than the San Francisco area.

Next, we will try to match each ZIP code with a water district. ZIP code boundaries do not always line up with water districts, and one water district often covers multiple ZIP codes, so this process is imprecise. It is even the case that some water districts overlap each other. Nonetheless, we can continue our analysis by matching each ZIP code to the water district with the largest geographic overlap.

The table `wd_vs_zip` describes the proportion of land in each ZIP code that is contained in each water district and vis versa. (The proportions are approximate because they do not correctly account for discontiguous districts, but they're mostly accurate.)

```
In [26]: wd_vs_zip.show(5)
```

```
<IPython.core.display.HTML object>
```

Question 2.3. Complete the `district_for_zip` function that takes a ZIP code. It returns the PWSID with the largest value of `ZIP in District` for that `zip_code`, if that value is at least 50%. Otherwise, it returns the string `'No District'`.

```
In [35]: def district_for_zip(zip_code):
             zip_code = str(zip_code) #Ensure that the ZIP code is a string, not an integer
             districts = wd_vs_zip.where('ZIP', zip_code).sort('ZIP in District', descending=True)
             at_least_half = (districts.num_rows > 0 and
                              districts.column('ZIP in District').item(0) >= 0.5)

             if at_least_half:
                 return districts.column('PWSID').item(0)
             else:
                 return "No District"

         district_for_zip(94709)
```

```
Out[35]: '0110005'
```

```
In [28]: _ = project1.grade('q23')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[oooooooooook] 100.0% passed
```

This function can be used to associate each ZIP code in the `income` table with a `PWSID` and discard ZIP codes that do not lie (mostly) in a water district.

```
In [29]: zip_pwsids = income.apply(district_for_zip, 'ZIP')
         income_with_pwsid = income.with_column('PWSID', zip_pwsids).where(zip_pwsids != "No District")
         income_with_pwsid.set_format(2, NumberFormatter(0)).show(5)
```

```
<IPython.core.display.HTML object>
```

Question 2.4. Create a table called `district_data` with one row per PWSID and the following columns:

- `PWSID`: The ID of the district
- `Population`: Population estimate
- `Water`: Average residential water use (gallons per person per day) in 2014-2015
- `Income`: Average income in dollars of all tax returns in ZIP codes that are (mostly) contained in the district according to `income_with_pwsid`.

Hint: First create a `district_income` table that sums the incomes and returns for ZIP codes in each water district.

```
In [30]: district_income = income_with_pwsid.group('PWSID',sum)
         district_data_total1= usage.join('PWSID',district_income)
         district_data_total2 = district_data_total1.with_column('Income',
             np.round(1000 * district_data_total1.column(5)/district_data_total1.column(4)))
         district_data_total2.apply(np.round,"Water")
         district_data = district_data_total2.select([0,1,2,7])
         district_data.set_format(['Population', 'Water', 'Income'], NumberFormatter(0))

Out[30]: PWSID   | Population | Water | Income
         0110001 | 340,000    | 71    | 79,032
         0110005 | 1,390,000  | 76    | 82,497
         0110006 | 151,037    | 57    | 52,924
         0110008 | 73,067     | 97    | 163,257
         0110009 | 79,547     | 69    | 133,902
         0410002 | 101,447    | 142   | 50,401
         0410006 | 18,300     | 286   | 38,721
         0410011 | 9,615      | 92    | 44,707
         0710001 | 106,455    | 110   | 53,551
         0710003 | 197,536    | 102   | 73,914
         ... (200 rows omitted)

In [31]: _ = project1.grade('q24')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2.5. The `bay_districts` table gives the names of all water districts in the San Francisco Bay Area. Is there an association between water usage and income among Bay Area water districts? Use the tables you have created to compare water usage between the 10 Bay Area water districts with the highest average income and the rest of the Bay Area districts, then describe the association. Do not include any districts in your analysis for which you do not have income information.

The names below are just suggestions; you may perform the analysis in any way you wish.

Note: Some Bay Area water districts may not appear in your `district_data` table. That's ok. Perform your analysis only on the subset of districts where you have both water usage & income information.

```
In [32]: bay_districts = Table.read_table('bay_districts.csv')
         bay_water_vs_income = district_table.join('popupContent',bay_districts,'District')\
                                             .join('PWSID',district_data)
         top_10 = np.average(bay_water_vs_income.sort("Income",descending = True).column("Water")[:10])
         rest = np.average(bay_water_vs_income.sort("Income",descending = True).column("Water")[10:])
         print("Top 10:", top_10," The rest:", rest, "Difference: ", top_10 - rest)
```

```
Top 10: 92.7636363636  The rest: 68.1181818182 Difference:  24.6454545455
```

Complete this one-sentence conclusion: In the Bay Area, people in the top 10 highest-income water districts used an average of 24.65 more gallons of water per person per day than people in the rest of the districts.

Question 2.6. In one paragraph, summarize what you have discovered through the analyses in this project and suggest what analysis should be conducted next to better understand California water usage, income, and geography. What additional data would be helpful in performing this next analysis?

This project explored the relationship between ZIP code, income, and water usage. In the first part, we saw the association that zip codes of areas with higher-than-average farmer proportions had lower incomes. We also saw that there was an association between higher incomes and the location of those ZIP codes being in the San Francisco Bay Area. In the second part, we explored associations related to water usage and geographic location. We identified the association that the Los Angeles area has a higher water usage than the San Francisco area. We also found that people in the top 10 highest-income water districts used more water per day than teh rest of the districts in the Bay Area. In order to better understand California water usage, income, and geography, we would want to see where the zip-codes with greater-than-average farmer proportions are located and see if there is a connection between water usage, income, and geography there. This would also tell us if there is regional differences between the association between income and water usage (i.e. in the Bay Area higher income means higher water usage, but this may not be true in the Los Angeles area).

Congratulations - you've finished Project 1 of Data 8!

To submit:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells, including the test cells. Make sure that the visualizations you create are actually displayed.
2. Select `Download as PDF via LaTeX (.pdf)` from the `File` menu. (Sometimes that seems to fail. If it does, you can download as HTML, open the .html file in your browser, and print it to a PDF.)
3. Read that file! If any of your lines are too long and get cut off, we won't be able to see them, so break them up into multiple lines and download again. If maps do not appear in the output, that's ok.
4. Submit that downloaded file (called `project1.pdf`) to Gradescope.

If you cannot submit online, come to office hours for assistance. The office hours schedule appears on data8.org/weekly.

```
In [33]: # For your convenience, you can run this cell to run all the tests at once!
         import os
         _ = [project1.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 100.0% passed
```

If you want, draw some more maps below.

```
In [34]: # Your extensions here (completely optional)
```