# hw03

March 10, 2022

Name: Allan Gongora

Section: 0131

## 1 Homework 3: Table Manipulation and Visualization

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

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

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-
packages (1.1.0)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
packages (from jinja2->gofer-grader) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
# Don't change this cell; just run it.

import numpy as np
from datascience import *


# These lines do some fancy plotting magic.\n",
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

from gofer.ok import check
```

**Recommended Reading**: * Visualization

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

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

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

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

## 1.1 1. Differences Between Universities

**Question 1.1** Suppose you're choosing a university to attend, and you'd like to *quantify* how *dissimilar* any two universities are. You rate each university you're considering on several numerical traits. You decide on a very detailed list of 1000 traits, and you measure all of them! Some examples:

- The cost to attend (per year).
- The average Yelp review of nearby Thai restaurants.
- The USA Today ranking of the Medical school.
- The USA Today ranking of the Engineering school.

You decide that the dissimilarity between two universities is the *total* of the differences in their traits. That is, the dissimilarity is:

- the **sum** of
- the absolute values of
- the 1000 differences in their trait values.

In the next cell, we've loaded arrays containing the 1000 trait values for Stanford and Berkeley. Compute the dissimilarity (according to the above technique) between Stanford and Berkeley. Call your answer `dissimilarity`. Use a single line of code to compute the answer.

*Note:* The data we're using aren't real – we made them up for this exercise, except for the cost-of-attendance numbers, which were found online.

[3]:
```
stanford = Table.read_table("stanford.csv").column("Trait value")
berkeley = Table.read_table("berkeley.csv").column("Trait value")

dissimilarity = abs(stanford - berkeley).sum()
dissimilarity
```

[3]:
```
14060.558701067921
```

[4]:
```
check('tests/q1_1.py')
```

[4]:
```
<gofer.ok.OKTestsResult at 0x7f0b46b6d310>
```

**Question 1.2** Why do we sum up the absolute values of the differences in trait values, rather than just summing up the differences?

We just care about the magnitude not the direction. Also positive changes and negative changes may cancel out and a score of 0 would mean they are they are same which is probably not true.

### 1.1.1 Weighing the Traits

After computing dissimilarities between several schools, you notice a problem with your method: the scale of the traits matters a lot.

Since schools cost tens of thousands of dollars to attend, the cost-to-attend trait is always a much bigger *number* than most other traits. That makes it affect the dissimilarity a lot more than other traits. Two schools that differ in cost-to-attend by $900, but are otherwise identical, get a dissimilarity of 900. But two schools that differ in graduation rate by 0.9 (a huge difference!), but are otherwise identical, get a dissimilarity of only 0.9.

One way to fix this problem is to assign different "weights" to different traits. For example, we could fix the problem above by multiplying the difference in the cost-to-attend traits by .001, so that a difference of $900 in the attendance cost results in a dissimilarity of $900 $\times$ .001, or 0.9.

Here's a revised method that does that for every trait:

1. For each trait, subtract the two schools' trait values.
2. Then take the absolute value of that difference.
3. Now multiply that absolute value by a trait-specific number, like .001 or 2.
4. Now, sum the 1000 resulting numbers.

**Question 1.3** Suppose you've already decided on a weight for each trait. These are loaded into an array called `weights` in the cell below. `weights.item(0)` is the weight for the first trait, `weights.item(1)` is the weight for the second trait, and so on. Use the revised method to compute a revised dissimilarity between Berkeley and Stanford.

*Hint:* Using array arithmetic, your answer should be almost as short as in question 1.

[5]:
```
weights = Table.read_table("weights.csv").column("Weight")

revised_dissimilarity = abs((stanford * weights) - (berkeley * weights)).sum()
revised_dissimilarity
```

[5]:
```
505.98313211458833
```

[6]:
```
check('tests/q1_3.py')
```

[6]:
```
<gofer.ok.OKTestsResult at 0x7f0b30d70710>
```

## 1.2 2. Unemployment

The Federal Reserve Bank of St. Louis publishes data about jobs in the US. Below, we've loaded data on unemployment in the United States. There are many ways of defining unemployment, and our dataset includes two notions of the unemployment rate:

1. Among people who are able to work and are looking for a full-time job, the percentage who can't find a job. This is called the Non-Employment Index, or NEI.
2. Among people who are able to work and are looking for a full-time job, the percentage who can't find any job *or* are only working at a part-time job. The latter group is called "Part-Time for Economic Reasons", so the acronym for this index is NEI-PTER. (Economists are great at marketing.)

The source of the data is here.

**Question 2.1** The data are in a CSV file called `unemployment.csv`. Load that file into a table called `unemployment`.

[7]:
```
unemployment = Table.read_table("unemployment.csv")
unemployment
```

[7]:
```
Date       | NEI     | NEI-PTER
1994-01-01 | 10.0974 | 11.172
1994-04-01 | 9.6239  | 10.7883
1994-07-01 | 9.3276  | 10.4831
1994-10-01 | 9.1071  | 10.2361
1995-01-01 | 8.9693  | 10.1832
1995-04-01 | 9.0314  | 10.1071
1995-07-01 | 8.9802  | 10.1084
1995-10-01 | 8.9932  | 10.1046
1996-01-01 | 9.0002  | 10.0531
1996-04-01 | 8.9038  | 9.9782
… (80 rows omitted)
```

[8]:
```
check('tests/q2_1.py')
```

[8]:
```
<gofer.ok.OKTestsResult at 0x7f0b78786810>
```

**Question 2.2** Sort the data in descending order by NEI, naming the sorted table `by_nei`. Create another table called `by_nei_pter` that's sorted in descending order by NEI-PTER instead.

[9]:
```
by_nei = unemployment.sort("NEI", descending=True)
by_nei_pter = unemployment.sort("NEI-PTER", descending=True)
```

[10]:
```
check('tests/q2_2.py')
```

[10]:
```
<gofer.ok.OKTestsResult at 0x7f0b30d24550>
```

**Question 2.3** Use `take` to make a table containing the data for the 10 quarters when NEI was greatest. Call that table `greatest_nei`.

[11]:
```
greatest_nei = by_nei.take(range(10))
greatest_nei
```

[11]:
```
Date       | NEI     | NEI-PTER
2009-10-01 | 10.9698 | 12.8557
2010-01-01 | 10.9054 | 12.7311
2009-07-01 | 10.8089 | 12.7404
2009-04-01 | 10.7082 | 12.5497
2010-04-01 | 10.6597 | 12.5664
2010-10-01 | 10.5856 | 12.4329
2010-07-01 | 10.5521 | 12.3897
2011-01-01 | 10.5024 | 12.3017
2011-07-01 | 10.4856 | 12.2507
2011-04-01 | 10.4409 | 12.247
```

[12]:
```
check('tests/q2_3.py')
```

[12]:
```
<gofer.ok.OKTestsResult at 0x7f0b30cf5ad0>
```

**Question 2.4** It's believed that many people became PTER (recall: "Part-Time for Economic Reasons") in the "Great Recession" of 2008-2009. NEI-PTER is the percentage of people who are unemployed (and counted in the NEI) plus the percentage of people who are PTER. Compute an array containing the percentage of people who were PTER in each quarter. (The first element of the array should correspond to the first row of `unemployment`, and so on.)

*Note:* Use the original `unemployment` table for this.

[18]:
```
pter = abs(unemployment.column("NEI-PTER") - unemployment.column("NEI"))
pter
```

[18]:
```
array([1.0746, 1.1644, 1.1555, 1.129 , 1.2139, 1.0757, 1.1282, 1.1114,
       1.0529, 1.0744, 1.1004, 1.0747, 1.0705, 1.0455, 1.008 , 0.9734,
       0.9753, 0.8931, 0.9451, 0.8367, 0.8208, 0.8105, 0.8248, 0.7578,
       0.7251, 0.7445, 0.7543, 0.7423, 0.7399, 0.7687, 0.8418, 0.9923,
       0.9181, 0.9629, 0.9703, 0.9575, 1.0333, 1.0781, 1.0675, 1.0354,
       1.0601, 1.01  , 1.0042, 1.0368, 0.9704, 0.923 , 0.9759, 0.93  ,
       0.889 , 0.821 , 0.9409, 0.955 , 0.898 , 0.8948, 0.9523, 0.9579,
       1.0149, 1.0762, 1.2873, 1.4335, 1.7446, 1.8415, 1.9315, 1.8859,
       1.8257, 1.9067, 1.8376, 1.8473, 1.7993, 1.8061, 1.7651, 1.7927,
       1.7286, 1.6387, 1.6808, 1.6805, 1.6629, 1.6253, 1.6477, 1.6298,
       1.4796, 1.5131, 1.4866, 1.4345, 1.3675, 1.3097, 1.2319, 1.1735,
       1.1844, 1.1746])
```

[19]:
```
check('tests/q2_4.py')
```

[19]:
```
<gofer.ok.OKTestsResult at 0x7f0b30bfc290>
```

**Question 2.5** Add `pter` as a column to `unemployment` (named "PTER") and sort the resulting

table by that column in descending order. Call the table `by_pter`.

Try to do this with a single line of code, if you can.

[20]:
```
by_pter = unemployment.with_column("PTER", pter).sort("PTER", descending=True)
by_pter
```

[20]:
```
Date       | NEI     | NEI-PTER | PTER
2009-07-01 | 10.8089 | 12.7404  | 1.9315
2010-04-01 | 10.6597 | 12.5664  | 1.9067
2009-10-01 | 10.9698 | 12.8557  | 1.8859
2010-10-01 | 10.5856 | 12.4329  | 1.8473
2009-04-01 | 10.7082 | 12.5497  | 1.8415
2010-07-01 | 10.5521 | 12.3897  | 1.8376
2010-01-01 | 10.9054 | 12.7311  | 1.8257
2011-04-01 | 10.4409 | 12.247   | 1.8061
2011-01-01 | 10.5024 | 12.3017  | 1.7993
2011-10-01 | 10.3287 | 12.1214  | 1.7927
… (80 rows omitted)
```

[21]:
```
check('tests/q2_5.py')
```

[21]:
```
<gofer.ok.OKTestsResult at 0x7f0b30bd7350>
```

**Question 2.6** Create a line plot of the PTER over time. To do this, first add the `year` array and the `pter` array to the `unemployment` table. Label these columns "Year" and "PTER", respectively. Then, generate a line plot using one of the table methods you've learned in class. Assign this new table to `pter_over_time`.

**Note:** If you're failing the test, but your plot looks correct, make sure that in your `unemployment` table, the "Year" column comes before the "PTER" column.

[85]:
```
year = 1994 + np.arange(by_pter.num_rows)/4
unemployment = unemployment.with_columns("Year", year, "PTER", pter)
pter_over_time = unemployment
pter_over_time.plot("Year", "PTER")
```


[28]:
```
check('tests/q2_6.py')
```

[28]:
```
<gofer.ok.OKTestsResult at 0x7f0b2ed81090>
```

**Question 2.7** Were PTER rates high during or directly after the Great Recession (that is to say, were PTER rates particularly high in the years 2008 through 2011)? Assign highPTER to `True` if you think PTER rates were high in this period, and `False` if you think they weren't.

[42]:
```
highPTER = True
```

[43]:
```
check('tests/q2_7.py')
```

[43]:
```
<gofer.ok.OKTestsResult at 0x7f0b2a0026d0>
```

## 1.3 3. Birth Rates

The following table gives census-based population estimates for each state on both July 1, 2015 and July 1, 2016. The last four columns describe the components of the estimated change in population during this time interval. **For all questions below, assume that the word "states" refers to all 52 rows including Puerto Rico & the District of Columbia.**

The data was taken from here.

If you want to read more about the different column descriptions, go here! As of February 2017, no descriptions were posted for 2010 - 2016.

[44]:
```
# Don't change this cell; just run it.
pop = Table.read_table('nst-est2016-alldata.csv').where('SUMLEV', 40).
 ↪select([1, 4, 12, 13, 27, 34, 62, 69])
pop = pop.relabeled('POPESTIMATE2015', '2015').relabeled('POPESTIMATE2016',␣
 ↪'2016')
pop = pop.relabeled('BIRTHS2016', 'BIRTHS').relabeled('DEATHS2016', 'DEATHS')
pop = pop.relabeled('NETMIG2016', 'MIGRATION').relabeled('RESIDUAL2016',␣
 ↪'OTHER')
pop.set_format([2, 3, 4, 5, 6, 7], NumberFormatter(decimals=0)).show(5)
```

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

**Question 3.1** Assign `us_birth_rate` to the total US annual birth rate during this time interval. The annual birth rate for a year-long period is the total number of births in that period as a proportion of the population size at the start of the time period.

**Hint:** What year corresponds to the start of the time period?

[47]:
```
us_birth_rate = (pop.column("BIRTHS") / pop.column("2015")).sum()
us_birth_rate
```

[47]:
```
0.6410672726840716
```

[48]:
```
check('tests/q3_1.py')
```

[48]:
```
<gofer.ok.OKTestsResult at 0x7f0b2aa255d0>
```

**Question 3.2** Assign `fastest_growth` to an array of the names of the five states with the fastest population growth rates in *descending order of growth rate*. We have first created a new version of the `pop` table, called `growth_rates`, which includes a column with the growth rate of each state. Making intermediate tables can improve the readability of the code and make it easier to follow when revisiting at a later time.

[53]:
```
growth_rates = pop.with_column('Growth Rate', (pop.column(3) / pop.column(2)) -␣
 ↪1)

fastest_growth = growth_rates.sort("Growth Rate", descending=True).
 ↪column("NAME")[:5]
```

```
fastest_growth
```

[53]:
```
array(['Utah', 'Nevada', 'Idaho', 'Florida', 'Washington'], dtype='<U20')
```

[54]:
```
check('tests/q3_2.py')
```

[54]:
```
<gofer.ok.OKTestsResult at 0x7f0b2af5f090>
```

**Question 3.3** Assign `movers` to the number of states for which the **absolute value** of the **annual rate of migration** was higher than 1%. The annual rate of migration for a year-long period is the net number of migrations (in and out) as a proportion of the population size at the start of the period. The `MIGRATION` column contains estimated annual net migration counts by state.

[66]:
```
migration_rates = abs(pop.column("MIGRATION") / pop.column("2015"))
movers = len(migration_rates[migration_rates > .01])
movers
```

[66]:
```
9
```

[67]:
```
check('tests/q3_3.py')
```

[67]:
```
<gofer.ok.OKTestsResult at 0x7f0b2a372fd0>
```

**Question 3.4** Assign `west_births` to the total number of births that occurred in region 4 (the Western US).

*Hint*: Make sure you double check the type of the values in the region column.

[75]:
```
pop
```

[75]:
```
REGION | NAME                 | 2015       | 2016       | BIRTHS  | DEATHS  |
MIGRATION | OTHER
3      | Alabama              | 4,853,875  | 4,863,300  | 58,556  | 52,405  |
3,874     | -600
4      | Alaska               | 737,709    | 741,894    | 11,255  | 4,511   |
-2,557    | -2
4      | Arizona              | 6,817,565  | 6,931,071  | 87,204  | 56,564  |
76,405    | 6,461
3      | Arkansas             | 2,977,853  | 2,988,248  | 37,936  | 30,581  |
3,530     | -490
4      | California           | 38,993,940 | 39,250,017 | 502,848 | 273,850 |
33,530    | -6,451
4      | Colorado             | 5,448,819  | 5,540,545  | 67,453  | 37,121  |
60,773    | 621
1      | Connecticut          | 3,584,730  | 3,576,452  | 35,848  | 30,638  |
-12,822   | -666
3      | Delaware             | 944,076    | 952,065    | 10,922  | 8,945   |
5,583     | 429
```

```
3      | District of Columbia | 670,377    | 681,170    | 9,779   | 5,455   |
6,392     | 77
3      | Florida              | 20,244,914 | 20,612,439 | 222,793 | 201,485 |
325,986   | 20,231
… (42 rows omitted)
```

[76]:
```
west_births = pop.where("REGION", "4").column("BIRTHS").sum()
west_births
```

[76]:
```
979657
```

[77]:
```
check('tests/q3_4.py')
```

[77]:
```
<gofer.ok.OKTestsResult at 0x7f0b2a497b50>
```

**Question 3.5** Assign `less_than_west_births` to the number of states that had a total population in 2016 that was smaller than the *total number of births in region 4 (the Western US)* during this time interval.

[78]:
```
less_than_west_births = pop.where("2016", are.below(west_births)).num_rows
less_than_west_births
```

[78]:
```
7
```

[79]:
```
check('tests/q3_5.py')
```

[79]:
```
<gofer.ok.OKTestsResult at 0x7f0b2a9f3f50>
```

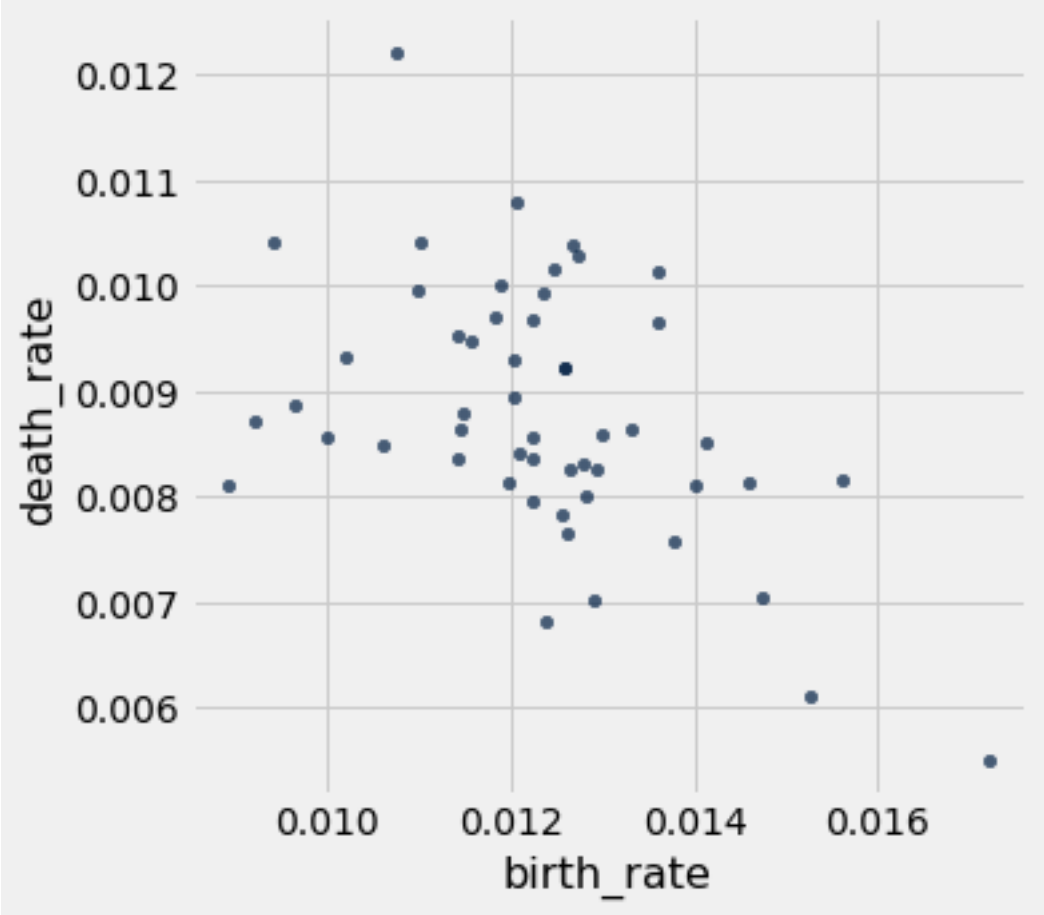
**Question 3.6** In the code cell below, create a visualization that will help us determine if there is an association between birth rate and death rate during this time interval. It may be helpful to create an intermediate table here.

[81]:
```
pop
```

[81]:
```
REGION | NAME                 | 2015       | 2016       | BIRTHS  | DEATHS  |
MIGRATION | OTHER
3      | Alabama              | 4,853,875  | 4,863,300  | 58,556  | 52,405  |
3,874     | -600
4      | Alaska               | 737,709    | 741,894    | 11,255  | 4,511   |
-2,557    | -2
4      | Arizona              | 6,817,565  | 6,931,071  | 87,204  | 56,564  |
76,405    | 6,461
3      | Arkansas             | 2,977,853  | 2,988,248  | 37,936  | 30,581  |
3,530     | -490
4      | California           | 38,993,940 | 39,250,017 | 502,848 | 273,850 |
33,530    | -6,451
4      | Colorado             | 5,448,819  | 5,540,545  | 67,453  | 37,121  |
60,773    | 621
```

```
1      | Connecticut          | 3,584,730  | 3,576,452  | 35,848  | 30,638  |
-12,822   | -666
3      | Delaware             | 944,076    | 952,065    | 10,922  | 8,945   |
5,583     | 429
3      | District of Columbia | 670,377    | 681,170    | 9,779   | 5,455   |
6,392     | 77
3      | Florida              | 20,244,914 | 20,612,439 | 222,793 | 201,485 |
325,986   | 20,231
… (42 rows omitted)
```

[82]:
```
# Generate your chart in this cell
br_dr = pop.with_columns("birth_rate", pop.column("BIRTHS") / pop.
 ↪column("2015"), "death_rate", pop.column("DEATHS") / \
                          pop.column("2015"))
br_dr.scatter("birth_rate", "death_rate")
```




**Question 3.7** `True` or `False`: There is an association between birth rate and death rate during this time interval.

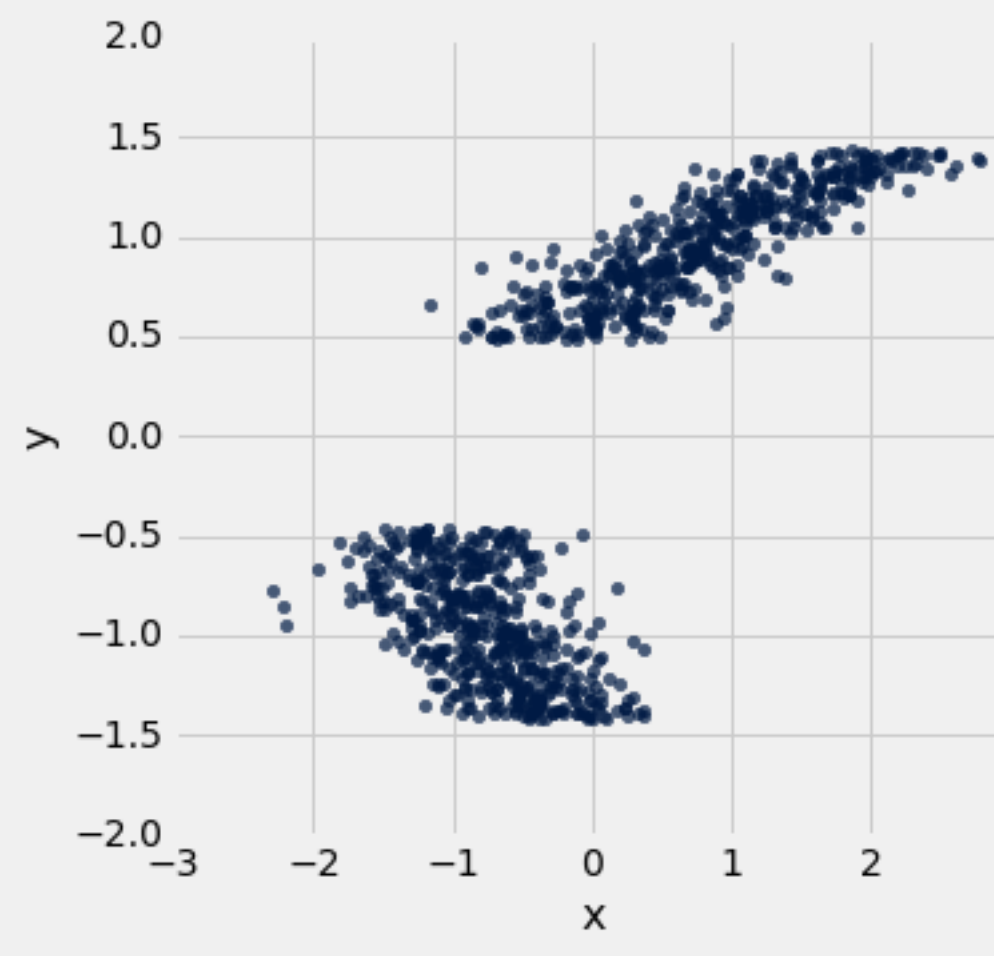
Assign `assoc` to `True` or `False` in the cell below.

[83]:
```
assoc = False
```

[84]:
```
check('tests/q3_7.py')
```

[84]:
```
<gofer.ok.OKTestsResult at 0x7f0b2a365890>
```

### 1.4 4. Marginal Histograms




Consider the following scatter plot:

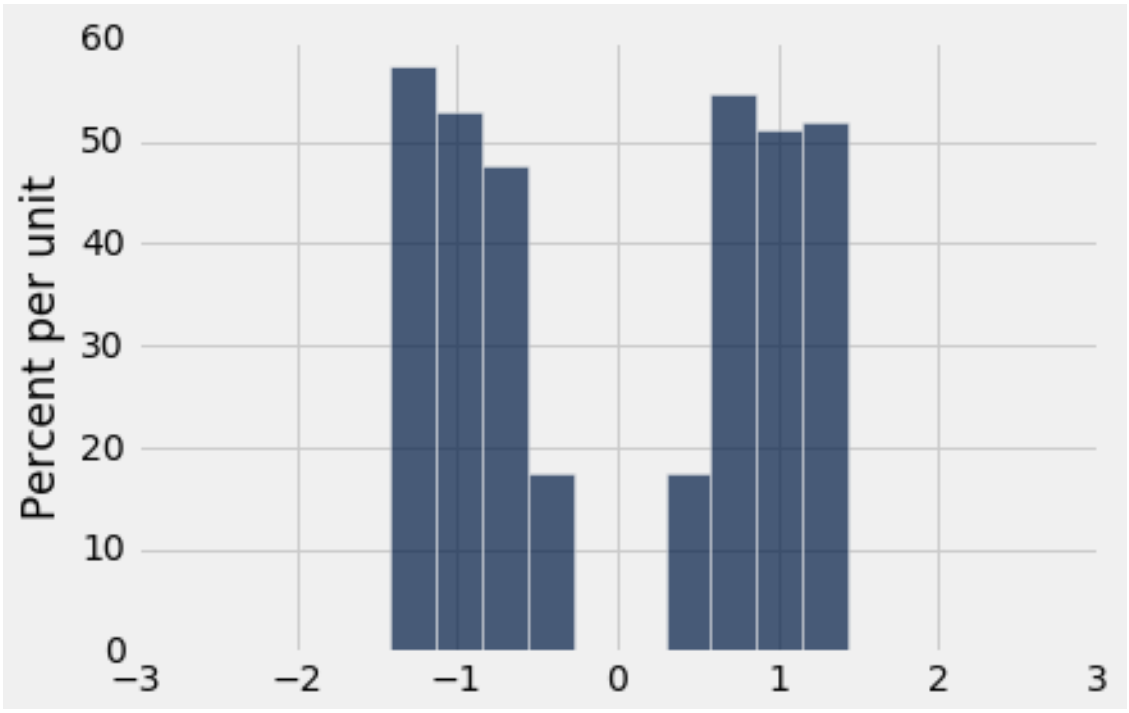
The axes of the plot represent values of two variables: $x$ and $y$.

Suppose we have a table called `t` that has two columns in it:

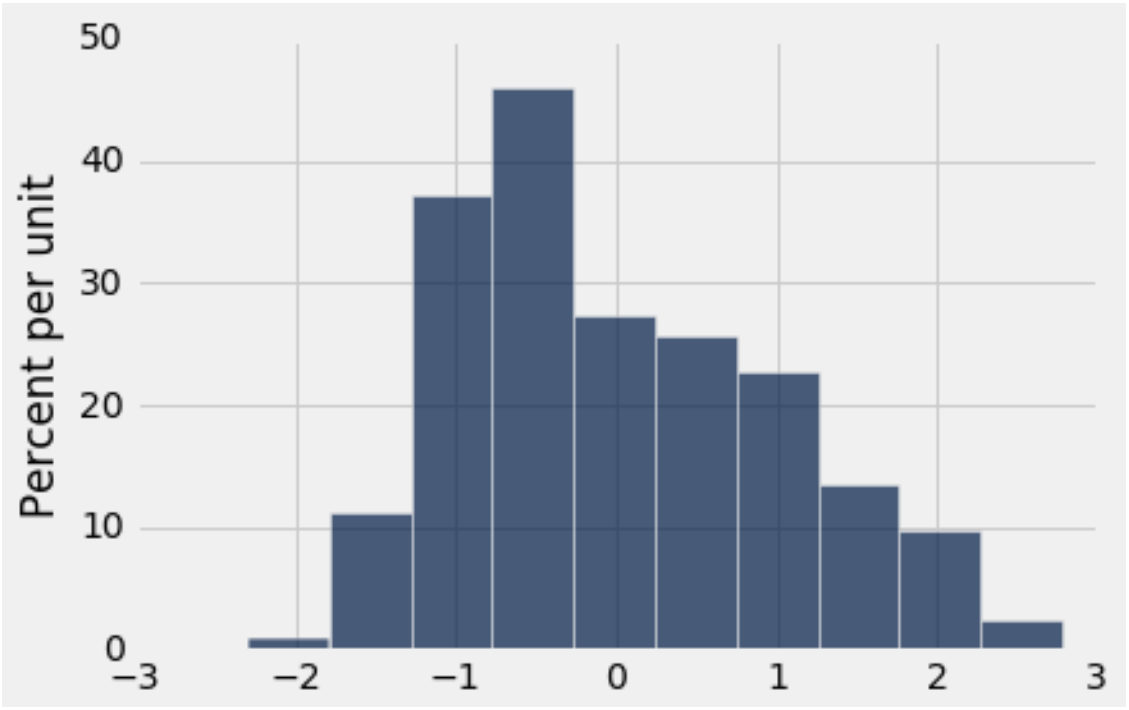
- `x`: a column containing the x-values of the points in the scatter plot
- `y`: a column containing the y-values of the points in the scatter plot

**Question 4.1** Match each of the following histograms to the code that produced them. Explain your reasoning.

**Histogram A:**




**Histogram B:**




**Line 1:** `t.hist('x')`

**Histogram for Line 1:**

**Explanation:** B because in relation to x the data is roughly normal

**Line 2:** `t.hist('y')`

**Histogram for Line 2:**

**Explanation:** A because in relation to y the data is bimodal

### 1.5 7. Submission

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

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

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

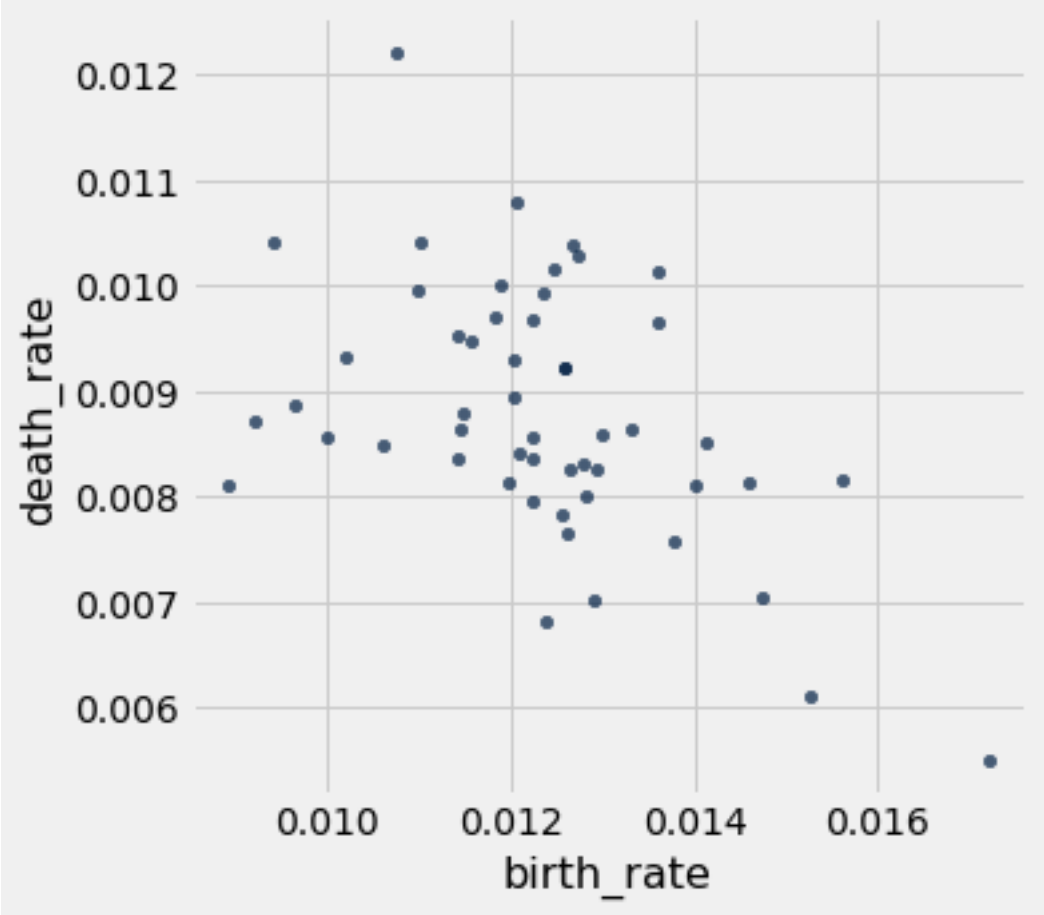
Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

[86]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('hw03.ipynb', sorted(glob.glob('tests/q*.py'))))
```

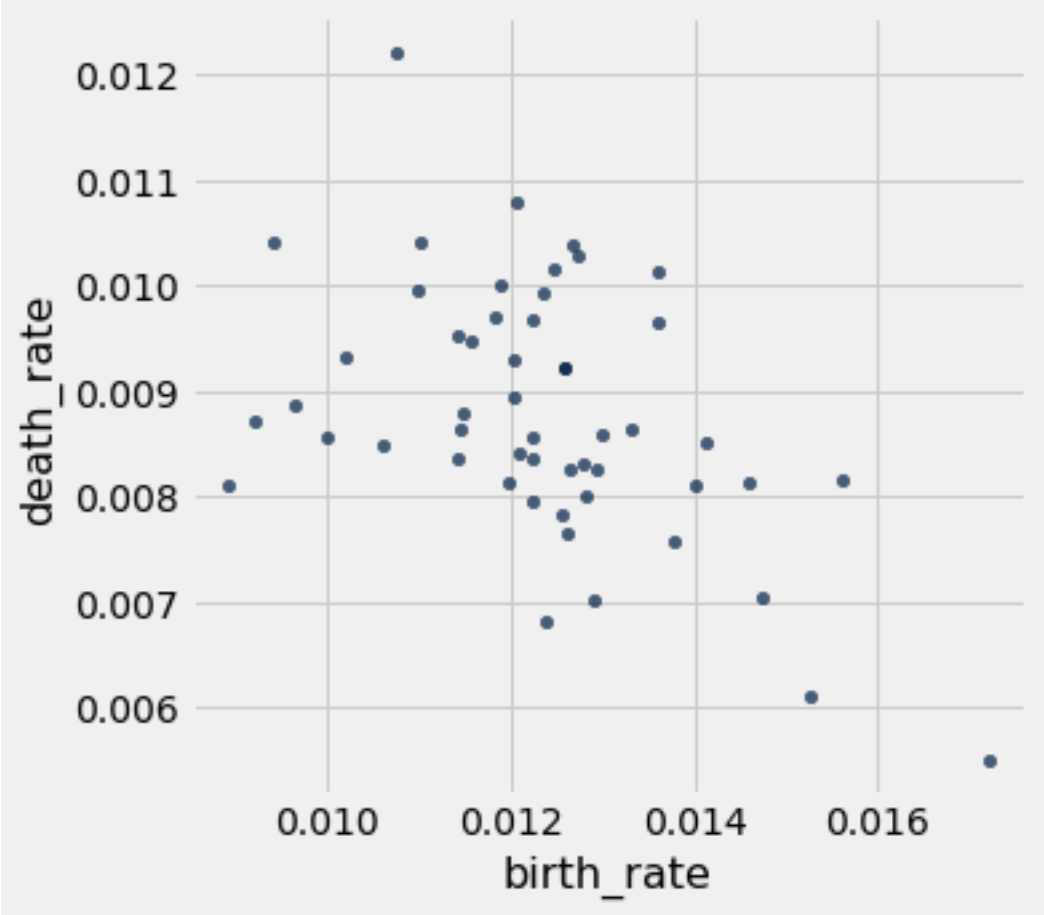
```
['tests/q1_1.py', 'tests/q1_3.py', 'tests/q2_1.py', 'tests/q2_2.py',
'tests/q2_3.py', 'tests/q2_4.py', 'tests/q2_5.py', 'tests/q2_6.py',
'tests/q2_7.py', 'tests/q3_1.py', 'tests/q3_2.py', 'tests/q3_3.py',
'tests/q3_4.py', 'tests/q3_5.py', 'tests/q3_7.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7f0b2a012b50>

Question 2:

<gofer.ok.OKTestsResult at 0x7f0b2a463a90>

Question 3:

<gofer.ok.OKTestsResult at 0x7f0b2a418090>

Question 4:

<gofer.ok.OKTestsResult at 0x7f0b2ae2c490>

Question 5:

<gofer.ok.OKTestsResult at 0x7f0b2ae2c890>

Question 6:

<gofer.ok.OKTestsResult at 0x7f0b2ae36650>
```

```
Question 7:
<gofer.ok.OKTestsResult at 0x7f0b2a864e10>
Question 8:
<gofer.ok.OKTestsResult at 0x7f0b2a964c50>
Question 9:
<gofer.ok.OKTestsResult at 0x7f0b2a463f50>
Question 10:
<gofer.ok.OKTestsResult at 0x7f0b2a87d890>
Question 11:
<gofer.ok.OKTestsResult at 0x7f0b2a8911d0>
Question 12:
<gofer.ok.OKTestsResult at 0x7f0b2a891190>
Question 13:
<gofer.ok.OKTestsResult at 0x7f0b2a87d950>
Question 14:
<gofer.ok.OKTestsResult at 0x7f0b2a8b1f10>
Question 15:
<gofer.ok.OKTestsResult at 0x7f0b2a8b1110>
1.0
```


1.8
1.6
1.4
1.2
1.0
0.8
PTER
1995
2000
2005
2010
2015
Year


0.012
0.011
0.010
0.009
0.008
0.007
0.006
death_rate
0.010
0.012
0.014
0.016
birth_rate


1.8
1.6
1.4
1.2
1.0
0.8
PTER
1995
2000
2005
2010
2015
Year




Name: Allan Gongora

Section: 0131