hw03

October 22, 2018

1 Homework 3: Tables and Charts, Due Sunday, October 21, at 11:59pm

Reading: Textbook chapters 6 and 7.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged.

You should start early so that you have time to get help if you're stuck.

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests. Each time you start your server, you will need to execute this cell again to load the tests.

In [1]:
```
# Don't change this cell; just run it.

import numpy as np
from datascience import *

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

from client.api.notebook import Notebook
ok = Notebook('hw03.ok')
_ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Homework 3: Tables and Charts
OK, version v1.13.11
=====================================================================



ERROR  | auth.py:91 | {'error': 'invalid_grant'}



Open the following URL:

https://okpy.org/client/login/
```

```
After logging in, copy the code from the web page and paste it into the box.
Then press the "Enter" key on your keyboard.

Paste your code here: n57atIhVQhC4cZ2NeebBrxFDK3rudo
Successfully logged in as wec149@ucsd.edu
```

Important: The `ok` tests don't always tell you that your answer is correct. More often, they help catch careless mistakes. It's up to you to ensure that your answer is correct. If you're not sure, ask someone (not for the answer, but for some guidance about your approach).

Once you're finished, you must do two things:

1.0.1 a. Turn into OK

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.

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

1.0.2 b. Turn PDF into Gradescope

Select File > Download As > PDF via LaTeX in the File menu. Turn in this PDF file into the respective assignement at https://gradescope.com/. If you submit more than once before the deadline, we will only grade your final submission

1.1 1. Avocadopocalypse

In early 2017, reduced harvests and labor strikes caused avocado production in Mexico, Peru, and California to dramatically decrease. In this problem we will investigate how and where prices fluctuated in response to the "avocadopocalypse".

The source of the avocado price data is here. We will use a subset of the data. The data has the following columns:

Column	Description
Date	The date on which a price was recorded in YEAR-MONTH format
Type	The type of avocado. Either "conventional" or "organic".
Region	Where the price was recorded.
Price	The average avocado price for that time period/location.
Volume	The number of avocados sold (in thousands).

Question 1. The data are in a CSV called `avocado.csv`. Read this file into a table named `avocado`.

```
In [76]: avocado = Table.read_table('avocado.csv')
         avocado.sort("Date", descending=False)
```

```
Out[76]: Date    | Region              | Type         | Price  | Volume
         2015-01 | Boise               | conventional | 1.075  | 66.5824
```

```
2015-01 | Seattle             | organic      | 1.525  | 38.6135
2015-01 | Chicago             | organic      | 1.73   | 13.5226
2015-01 | Plains              | conventional | 1.05   | 1568.1
2015-01 | BaltimoreWashington | conventional | 1.17   | 702.233
2015-01 | NorthernNewEngland  | organic      | 1.8725 | 7.04072
2015-01 | Southeast           | conventional | 1.125  | 2752.8
2015-01 | SouthCentral        | conventional | 0.7925 | 5254.72
2015-01 | California          | conventional | 0.9825 | 5575.77
2015-01 | West                | conventional | 0.935  | 5516.05
... (4124 rows omitted)
```

```
In [7]: _ = ok.grade('q1_1')
```

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

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

Question 2. Assign `cheapest` to the name of the region which had the cheapest avocados (of any type, and any date), and similarly assign `most_expensive` to the name of the region with the most expensive avocados (of any type, and any date).

```
In [27]: cheapest = avocado.sort('Price', descending=False).column('Region').take(0)
         most_expensive = avocado.sort('Price', descending=True).column('Region').take(0)
         # cheapest
         most_expensive
```

```
Out[27]: 'SanFrancisco'
```

```
In [28]: _ = ok.grade('q1_2')
```

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

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

Question 3. Avocados are sold in two types: organic and conventional. Make a table named `organic` containing the data for organic avocados, and a table named `conventional` for conventional avocados.

```
In [50]: organic = avocado.where('Type', 'organic')
         conventional = avocado.where('Type', 'conventional')
         organic
         conventional
         region = avocado.sort('Date', descending=False).where('Region','SanFrancisco').column('
         region

Out[50]: array(['2015-01', '2015-02', '2015-03', '2015-04', '2015-05', '2015-06',
                '2015-07', '2015-08', '2015-09', '2015-10', '2015-11', '2015-12',
                '2016-01', '2016-02', '2016-03', '2016-04', '2016-05', '2016-06',
                '2016-07', '2016-08', '2016-09', '2016-10', '2016-11', '2016-12',
                '2017-01', '2017-02', '2017-03', '2017-04', '2017-05', '2017-06',
                '2017-07', '2017-08', '2017-09', '2017-10', '2017-11', '2017-12',
                '2018-01', '2018-02', '2018-03'], dtype='<U7')

In [31]: _ = ok.grade('q1_3')

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

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

Question 4: Create a table named `sf_total` containing two columns: `Date` and `Total Volume`. The `Total Volume` column should be equal to the total volume of avocados (organic + conventional) sold in San Francisco for each date.

```
In [54]: # hint: you can always use intermediate variables and more than one line of code
         sf_total = Table().with_columns(
             "Date", avocado.sort('Date', descending=False).where('Region','SanFrancisco').colum
             "Total Volume", avocado.sort('Date', descending=False).where('Region','SanFrancisco
             + avocado.sort('Date', descending=False).where('Region','SanFrancisco').where('Type
         )
         sf_total

Out[54]: Date    | Total Volume
         2015-01 | 853.113
         2015-02 | 975.321
         2015-03 | 874.978
         2015-04 | 771.943
         2015-05 | 873.352
         2015-06 | 716.492
         2015-07 | 694.041
         2015-08 | 640.496
```

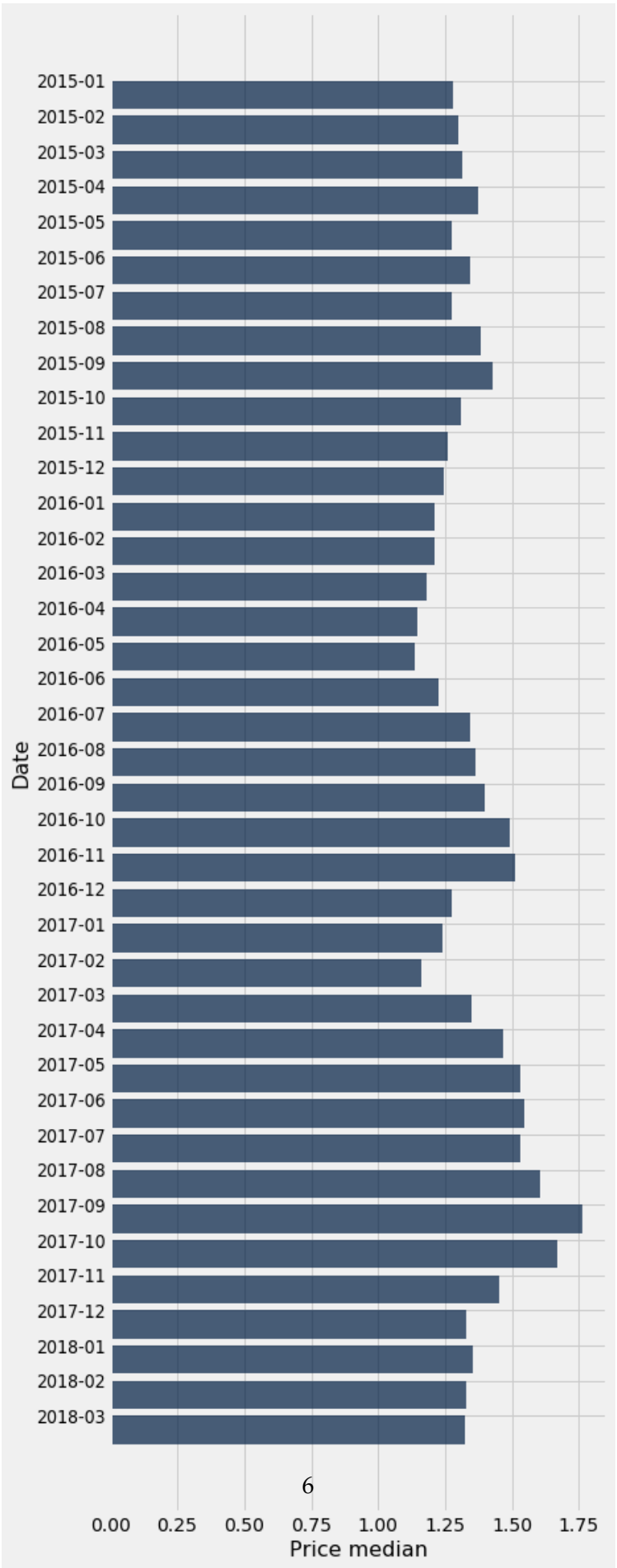
```
          2015-09 | 634.54
          2015-10 | 634.018
          ... (29 rows omitted)

In [55]: _ = ok.grade('q1_4')

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

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

Question 5. By default, the `.group()` method asks for a column name whose values will be used to group the rows into categories. It creates a `count` column which contains the size of each group. But you can also provide `.group()` with a second argument: a function which will be applied to each group. For example, to compute the median price of conventional avocados in the entire US, we group by `Date` and apply `np.median()`:

```
In [73]: median_price = conventional.select('Date', 'Price').group('Date')
         median_price

Out[73]: Date    | count
         2015-01 | 53
         2015-02 | 53
         2015-03 | 53
         2015-04 | 53
         2015-05 | 53
         2015-06 | 53
         2015-07 | 53
         2015-08 | 53
         2015-09 | 53
         2015-10 | 53
         ... (29 rows omitted)
```

Plot the median price of conventional avocados in the entire US as a function of date.

```
In [74]: # put your line of code here:
         median_price_whole_year = avocado.select('Date', 'Price').group('Date', np.median)
         median_price_whole_year.barh('Date', 'Price median')

         # keep this line here to rotate the x-axis labels
         # plt.gca().tick_params(axis='x', rotation=90)
```


2015-01
2015-02
2015-03
2015-04
2015-05
2015-06
2015-07
2015-08
2015-09
2015-10
2015-11
2015-12
2016-01
2016-02
2016-03
2016-04
2016-05
2016-06
2016-07
2016-08
2016-09
2016-10
2016-11
2016-12
2017-01
2017-02
2017-03
2017-04
2017-05
2017-06
2017-07
2017-08
2017-09
2017-10
2017-11
2017-12
2018-01
2018-02
2018-03
Date
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Price median

Question 6. Does the data support the "avocadopocalypse"; that is, were prices higher in 2017? What could you plot in order to investigate the *reason* that prices were higher?

By just looking at the price medians in 2017, I think that the price of avocados were higher overall during 2017. The overall price medians in 2017 were higher than the other previous years, and kept on increasing througout the year. In order to investigate the reason that prices were higher, we could plot the volume produced by each region during each year and their prices. If some regions were to produce less avocados and increase the price of those avocados, that would probably mean that there might've been a natural catastrophe during that time(heat waves, droughts, etc...), leading to a decrease in volume of the avocado cultivation, and pushing the prices up, increasing the price median in 2017.

1.2 2. Music Duration and Loudness

The Million Song Dataset is a "freely-available collection of audio features and metadata for a million contemporary popular music tracks". In this problem, we will explore a small subset of the dataset.

```
In [95]: # run this cell to load the data
         songs = Table.read_table('songs.csv')
         songs.sort('Year', descending=False)
```

```
Out[95]: Artist                | Title                      | Genre               | Year | Dura
         Blind Lemon Jefferson | Got The Blues              | blues               | 1926 | 175.
         Blind Lemon Jefferson | Long Lonesome Blues        | blues               | 1926 | 178.
         Ma Rainey             | Ma Rainey's Black Bottom   | classic female blues | 1927 | 188.
         Blind Lemon Jefferson | Wartime Blues              | blues               | 1927 | 185.
         Blind Willie McTell   | Writing Paper Blues        | blues               | 1927 | 191.
         Charley Patton        | Mississippi Boweavil Blues | blues               | 1929 | 191.
         Sleepy John Estes     | Street Car Blues           | blues               | 1930 | 196.
         Charley Patton        | Moon Going Down            | blues               | 1930 | 196.
         Charlie Patton        | 34 Blues                   | blues               | 1934 | 173.
         Sleepy John Estes     | Down South Blues           | blues               | 1935 | 188.
         ... (4667 rows omitted)
```

Each row in the table is a song. The `Duration` column measures the length of the song, in seconds. The `Loudness` column measures the average loudness of the song, in decibels.

Question 1. Assign `by_genre` to a table which counts the number of songs per genre.

```
In [79]: by_genre = songs.group('Genre')
         by_genre
```

```
Out[79]: Genre               | count
         acid jazz           | 3
         acoustic            | 1
         alternative         | 9
         alternative country | 2
         alternative dance   | 3
         alternative metal   | 22
```

```
          alternative rock    | 9
          ambient             | 4
          arabesque           | 1
          art rock            | 1
          ... (240 rows omitted)
```

```
In [80]: _ = ok.grade('q2_1')
```

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

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

Question 2. Compute the number of songs in the dataset which were released during or before 1970, and save the result in `on_or_before_1970`.

```
In [86]: on_or_before_1970 = songs.where('Year', are.below_or_equal_to(1970)).column('Title').si
         on_or_before_1970
```

```
Out[86]: 204
```

```
In [87]: _ = ok.grade('q2_2')
```

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

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

Question 3. Make an array called `longest` which contains the titles of the 10 longest songs in order of decreasing duration.

```
In [111]: longest = songs.sort('Duration', descending=True).take(np.arange(1,11,1)).column('Titl
          longest
          "Mean Loudness", songs.select('Year', 'Loudness').group('Year', np.mean)
```

```
Out[111]: ('Mean Loudness', Year | Loudness mean
           1926 | -21.6725
```

```
            1927 | -16.135
            1929 | -14.873
            1930 | -16.751
            1934 | -30.53
            1935 | -17.0835
            1936 | -10.697
            1940 | -18.69
            1947 | -19.4945
            1950 | -16.472
            ... (58 rows omitted))

In [108]: _ = ok.grade('q2_3')

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

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

Question 4. Create a table named `mean_loudness` which has two columns: the `Year`, and a column `Mean Loudness` which contains the mean loudness of every song released in that year.

Hint: You can use the 2-argument form of `.group()` discussed above.

```
In [118]: mean_loudness = songs.select('Year', 'Loudness').group('Year', np.mean).relabeled('Lou
          mean_loudness

Out[118]: Year | Mean Loudness
          1926 | -21.6725
          1927 | -16.135
          1929 | -14.873
          1930 | -16.751
          1934 | -30.53
          1935 | -17.0835
          1936 | -10.697
          1940 | -18.69
          1947 | -19.4945
          1950 | -16.472
          ... (58 rows omitted)

In [119]: _ = ok.grade('q2_4')

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

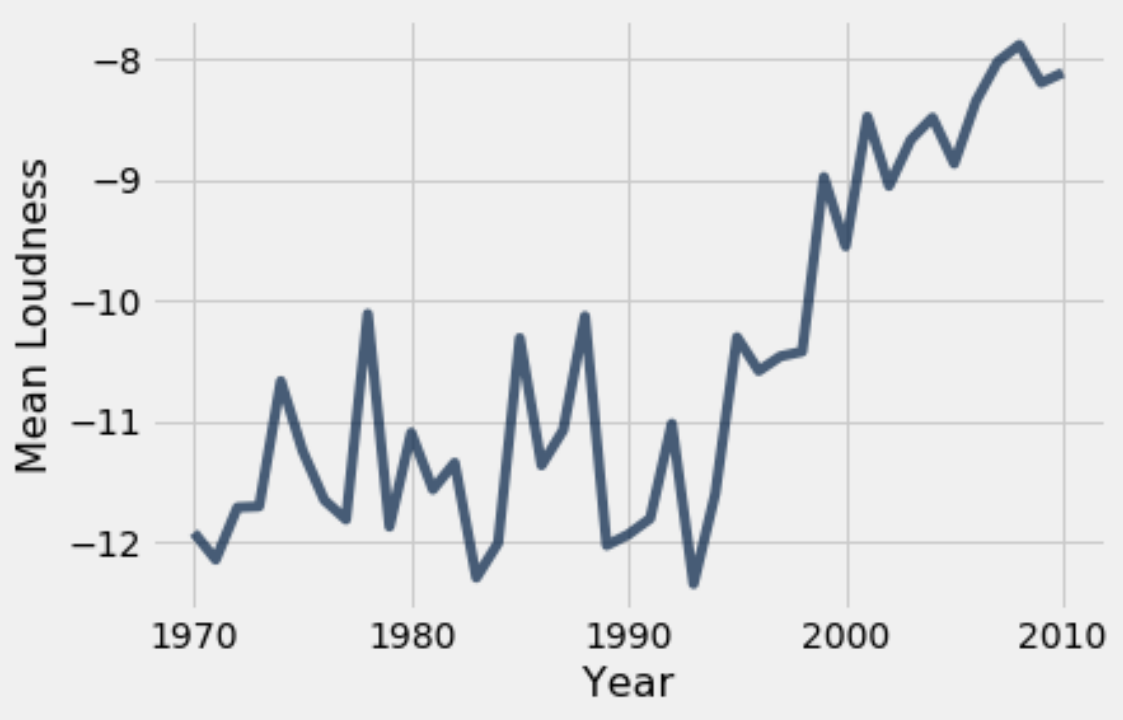
```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 5. Create an array named `above_9` which contains all years whose mean loudness is above -9 dB.

```
In [126]: above_9 = mean_loudness.where('Mean Loudness', are.above(-9)).column('Mean Loudness')
          above_9

Out[126]: array([-8.97589888, -8.47577103, -8.66701575, -8.48297407, -8.86554934,
                 -8.34753437, -8.01922105, -7.87920949, -8.194216  , -8.11114063])

In [127]: _ = ok.grade('q2_5')

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

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

Question 6. Draw a line plot of the mean loudness by year, from 1970 on (including 1970).

```
In [130]: above_1970 = mean_loudness.where('Year', are.above_or_equal_to(1970))
          above_1970.plot('Year', 'Mean Loudness')
```



1.3 3. Consumer Financial Protection Bureau Complaints

The Consumer Financial Protection Bureau has collected and published consumer complaints against financial companies since 2011. The data are available here (or at this direct link. For this exercise, to make your code run faster, we've selected only the data from May 2016.

Run the next cell to load the data. Each row represents one consumer's complaint.

```
In [131]: # just run this cell
          complaints = Table.read_table("complaints.csv")
          complaints
```

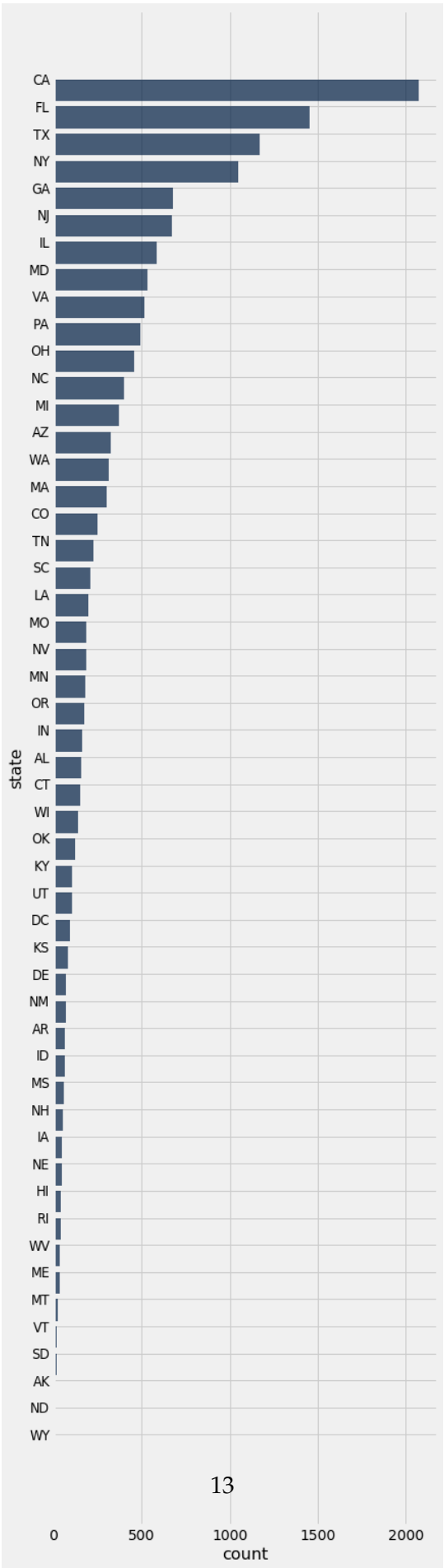
```
Out[131]: company                                | company_public_response
          TransUnion Intermediate Holdings, Inc. | Company has responded to the consumer and the
          TransUnion Intermediate Holdings, Inc. | Company has responded to the consumer and the
          Bank of America                        | Company has responded to the consumer and the
          Finance of America Reverse LLC         | Company believes it acted appropriately as au
          Acceptance Solutions Group, INC        | Company believes it acted appropriately as au
          Equifax                                | (None)
          TransUnion Intermediate Holdings, Inc. | Company has responded to the consumer and the
          Encore Capital Group                   | (None)
          Nationstar Mortgage                    | (None)
          Convergent Resources, Inc.             | (None)
          ... (14842 rows omitted)
```

Question 1. What percentage of complaints were made via the web? Save your answer to the variable `web_percent`. It should be a decimal between 0 and 1.

```
In [136]: web_number = complaints.where('submitted_via', are.equal_to('Web')).column('submitted_
          total_number = complaints.column('submitted_via').size
          web_percent = web_number/total_number
          web_percent

Out[136]: 0.7087934284944789

In [137]: _ = ok.grade('q3_1')

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

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

Question 2. Make a bar chart showing the number of complaints per state. Make sure that the bars are sorted from longest to shortest.

```
In [155]: complaint_counts = complaints.group('state').sort('count', descending=True).barh('stat
          complaint_counts
```


CA
FL
TX
NY
GA
NJ
IL
MD
VA
PA
OH
NC
MI
AZ
WA
MA
CO
TN
SC
LA
MO
NV
MN
OR
IN
AL
CT
WI
OK
KY
UT
DC
KS
DE
NM
AR
ID
MS
NH
IA
NE
HI
RI
WV
ME
MT
VT
SD
AK
ND
WY
state
0
500
1000
1500
2000
count

Question 3. Of course California has the most complaints... it has the most people! Now let's normalize by the state population. The CSV file `populations.csv` contains the population of each US state. The next cell reads it in:

```
In [151]: # don't edit these lines, just run them!
          populations = Table.read_table('populations.csv')
          populations = populations.sort('State')
          populations

Out[151]: Population | State
          736732     | AK
          4849377    | AL
          2966369    | AR
          6731484    | AZ
          38802500   | CA
          5355866    | CO
          3596677    | CT
          658893     | DC
          935614     | DE
          19893297   | FL
          ... (41 rows omitted)
```

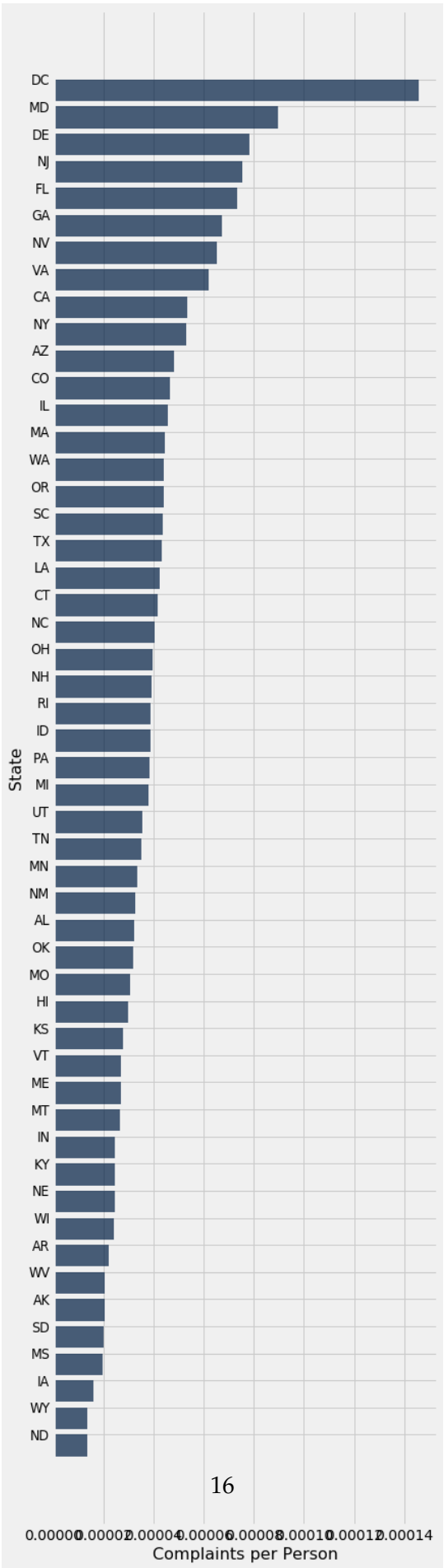
Make a table named `complaints_per_person` which has two columns: `State`, which is the name of the state, and `Complaints per Person`, which has the number of complaints submitted by people in that state divided by the number of people in the state. That is, the complaints per capita.

```
In [159]: # as always, you can use more than one line of code and intermediate variables, if you
          number_per = complaints.group('state').sort('state', descending=False).column('count')
          complaints_per_person = Table().with_columns(
              "State", populations.column('State'),
              "Complaints per Person", number_per
          )
          complaints_per_person

Out[159]: State | Complaints per Person
          AK    | 2.03602e-05
          AL    | 3.23753e-05
          AR    | 2.19123e-05
          AZ    | 4.8132e-05
          CA    | 5.33728e-05
          CO    | 4.64911e-05
          CT    | 4.17052e-05
          DC    | 0.000145699
          DE    | 7.80236e-05
          FL    | 7.31402e-05
          ... (41 rows omitted)
```

```
In [160]: _ = ok.grade('q3_3')

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

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

Question 4. Make a bar chart where the length of each bar is the complaints per person, and there is a bar for each state. Make sure that the bars are sorted from longest to shortest.

```
In [163]: complaints_per_person.sort('Complaints per Person', descending=True).barh('State')
```


State
DC
MD
DE
NJ
FL
GA
NV
VA
CA
NY
AZ
CO
IL
MA
WA
OR
SC
TX
LA
CT
NC
OH
NH
RI
ID
PA
MI
UT
TN
MN
NM
AL
OK
MO
HI
KS
VT
ME
MT
IN
KY
NE
WI
AR
WV
AK
SD
MS
IA
WY
ND
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
Complaints per Person

Question 5. Which company settles the most complaints by giving money to the complainant? Assign your answer to `most_monetary_relief`.

Note: Complaints settled by monetary relief are those where `company_response` is "Closed with monetary relief"

```
In [174]: most_monetary_relief = complaints.where('company_response', are.equal_to('Closed with
          most_monetary_relief

Out[174]: 'Citibank'

In [175]: _ = ok.grade('q3_5')

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

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

1.4 4. Histograms

Suppose we have a table called `data` with two numerical columns, "x" and "y". Consider the following scatter plot, which was generated by calling `data.scatter('x', 'y')`:

Now consider these two histograms:

Histogram A:

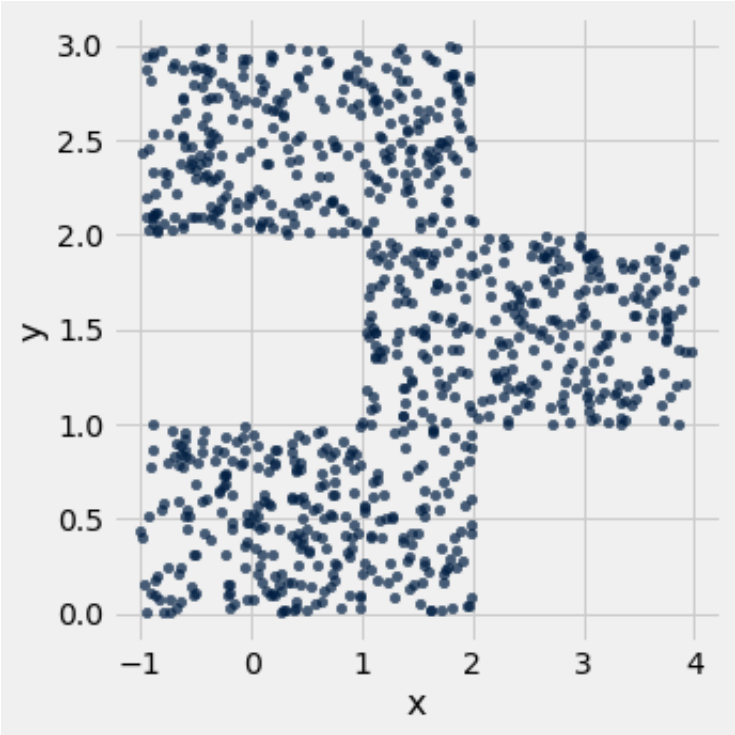
Histogram B:

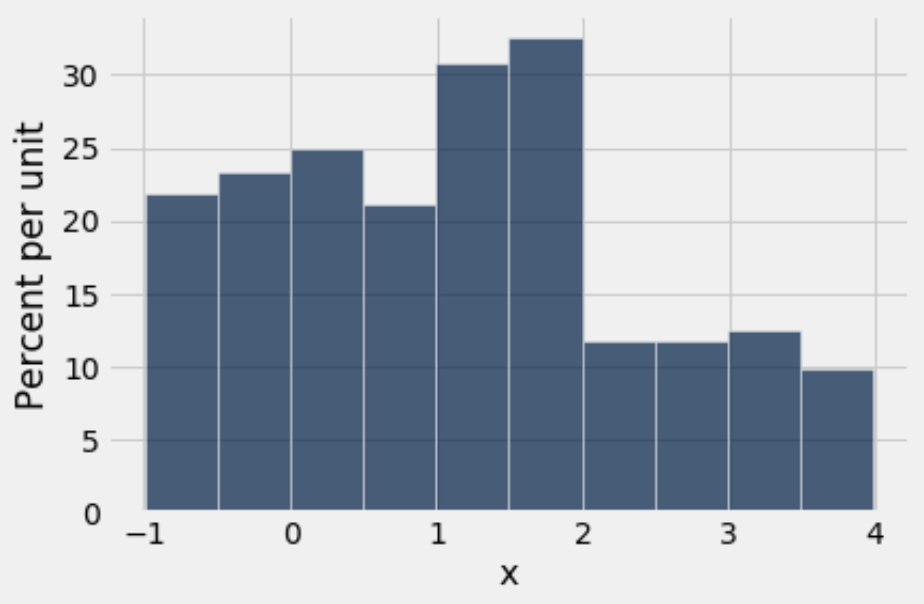
Question 1. Out of `data.hist('x')` and `data.hist('y')`, which line generated Histogram A? Which generated Histogram B? Explain.

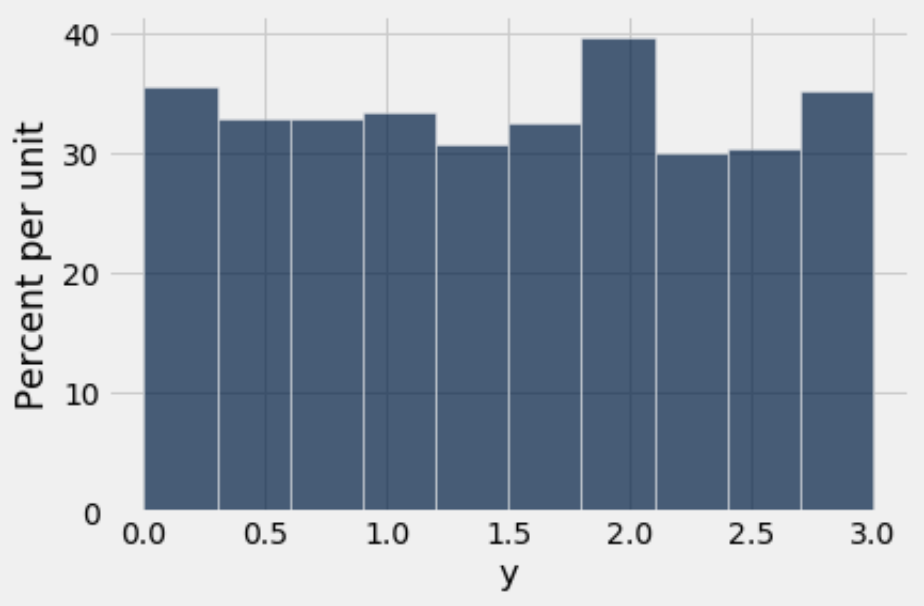
data.hist('x') generated Histogram A and data.hist('y') generated Histogram B. Since there are 3 areas generated by the dots, there are two areas overlapping in the x coordinate for the first 2 areas(the are on the top and the area on the bottom), therefore the histogram generated by data.hist('x') will have higher values in the first 2/3 half of the plot (since it will have more number of dots in the same area) and will have less value on the end, since only the values of one area(the area in the middle) are being counted. While on the y coordinate, each area is distributed evenly, since there are no overlaps between any of the dotted areas. Therefore data.hist('y') has a more uniform graph.

Question 2. Suppose we run this line of code:

```
new_data = Table().with_columns(
    'x',
    data.column('x') + 3
    'y',
    data.column('y')
)
```


3.0
2.5
2.0
y 1.5
1.0
0.5
0.0
−1
0
1
2
3
4
x


Percent per unit
30
25
20
15
10
5
0
−1
0
1
2
3
4
x



We then run `new_data.hist('x')`. What does the new histogram look like?

The new histrogram will have the same shape and the same y-coordinate values. While it will add each value in the x-coordinate a value of 3 (+3 to each value in x-coordinate). So the x-coordinate shown would be 2 3 4 5 6 7 (instead of -1 0 1 2 3 4).

1.5 Don't forget to submit to both OK *and* Gradescope!