Homework 3: Tables and Charts

Please complete this notebook by filling in the cells provided. When you're done:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells.
2. Select `Download as HTML (.html)` from the `File` menu
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.
4. Select `Download as notebook (.ipynb)` from the `File` menu
5. Give both files a name in this format: hw_01_First Name_Last Name
6. Submit both downloaded files to Moodle.

Run the cell below to prepare the notebook.

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

# These lines do some fancy plotting magic.
import matplotlib
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)
%matplotlib inline
```

1. Differences between Universities, Part II

Question 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 method) 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 we estimated for a student from a median-income family living in California using this tool.

In [2]:
```
stanford = Table.read_table("~/DS_113_S23/HW/HW_3/stanford.csv").column("Tra
berkeley = Table.read_table("~/DS_113_S23/HW/HW_3/berkeley.csv").column("Tra

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

Out[2]:
```
14060.558701067919
```

Question 2. Identify all the subexpressions in your answer to the previous question, excluding the whole expression itself. Write each on its own line. Before each one, write a one-line comment describing the value of the subexpression, including what type of value it is. We've written the first one for you. (It should appear somewhere in your answer to the previous question!)

In [3]:
```
# An array of 1000 numbers, each a different measured trait of Stanford Univ
stanford

# An array of 1000 numbers, each a different measured trait of Berkeley Univ
berkeley

# The 1000 differences in trait values between Stanford and Berkeley
(stanford-berkeley)

# The absolute values of the 1000 differences in their trait values
abs(stanford-berkeley)
```

```
Out[3]: array([ 5.61600000e+03,   2.41979515e+00,   1.97372327e+01,
                1.21378628e+01,   2.47006573e+00,   1.34441019e+00,
                3.00684613e+01,   9.02123498e-01,   9.73692348e+00,
                5.10505479e+00,   1.41208547e+01,   2.41895300e-01,
                2.91122014e+01,   5.35688027e-01,   2.59284879e+00,
                2.42488140e+00,   8.21891530e-01,   2.44133240e+01,
                3.38664338e-01,   1.82125839e+00,   2.11965349e-01,
                9.66773373e-01,   1.83297789e+01,   8.07627632e+00,
                1.00426323e+01,   7.19553024e+00,   3.34766992e+01,
                7.53576197e+00,   9.05351376e-01,   3.67089507e-01,
                9.82798368e+00,   1.71004011e-01,   2.32624206e+01,
                3.49601069e+01,   5.71255825e+00,   1.20670294e+01,
                2.72517490e-01,   1.83015949e+01,   1.64402761e+00,
                8.15170025e-01,   4.19736706e+00,   1.90515081e+01,
                5.47656323e-01,   5.51614634e+00,   6.43273434e+00,
                6.78062557e+00,   1.78974219e+01,   9.35305831e-01,
                2.43012236e+00,   8.79391765e-01,   1.35344324e+00,
                2.13539710e+00,   5.28986313e-01,   2.03587166e+01,
                1.67649111e+00,   1.19599011e+00,   1.65133474e+00,
                3.37436951e+01,   1.47205441e+00,   3.13028057e+00,
                1.46150880e+01,   2.82056768e+01,   9.36377261e-01,
                4.48515047e-01,   1.85642671e-01,   1.55314616e+00,
                1.71343669e+00,   1.20114947e+00,   1.84861412e+00,
                1.42990222e+01,   4.38559070e-01,   2.71453764e-01,
                5.91926358e-03,   1.00833044e+01,   3.16995455e+01,
                3.00168672e+00,   2.91979752e+00,   8.51826464e-01,
                3.26628317e+01,   5.39847103e-01,   6.40306921e-01,
                1.51258958e+01,   9.62633123e-01,   3.04476426e-01,
                4.27673053e+01,   6.23675401e-01,   1.48396219e+01,
                6.18843225e-01,   6.77333165e+00,   7.80949805e-01,
                3.54713663e+01,   2.00300182e+00,   1.23137254e+00,
                2.44401000e+01,   1.72261249e+01,   1.16319728e+00,
                1.18873551e+01,   2.90742722e+00,   1.00436535e+00,
                3.88852641e-01,   1.77277776e+00,   1.02215063e+00,
                1.00483521e+00,   9.14941663e+00,   3.16805952e+01,
                2.28590729e+00,   8.89581381e+00,   1.86181290e-01,
                6.22838034e-02,   1.61981010e+01,   3.87785188e-01,
                1.24173012e+00,   3.84917947e-01,   6.89898910e+00,
                2.31730666e+01,   1.05828001e+00,   3.66677646e+00,
                1.03100035e+00,   1.13305349e+00,   3.67384173e+01,
                6.54942534e+00,   6.62397040e-01,   7.20020167e-01,
                2.17817981e+01,   1.50870633e+01,   1.86099239e+00,
                2.93349156e+00,   4.74118242e-01,   2.30473455e+01,
                7.20273945e-01,   1.13280881e+00,   3.92579873e-01,
                1.83526522e-01,   4.29141188e-01,   1.95776754e+01,
                5.92766059e-02,   1.29219179e+00,   4.68246032e+00,
                8.99053002e+00,   1.97594749e+00,   3.52837659e+00,
                9.29359367e-01,   1.74262417e+00,   2.76092853e-03,
                2.21275169e+01,   1.98873533e+00,   6.43621475e-01,
                3.09988783e+01,   2.45018906e+00,   1.90218915e+01,
                2.98928224e+01,   1.67196227e+00,   1.14663602e+00,
                2.63613708e+01,   8.36615200e-02,   7.06285790e-01,
                1.14341103e+00,   2.59648447e+01,   2.29142547e+01,
                2.62374800e+01,   1.52027166e+00,   4.53310845e+01,
                5.82043088e-01,   1.92896833e+00,   3.18090492e+01,
                6.65009366e-01,   8.90456028e-02,   3.14734816e+00,
                1.50198716e+00,   1.63533038e+01,   7.37881019e-01,
                9.89869820e-01,   6.26452228e-01,   1.82675902e+00,
                7.14344426e+00,   8.84346084e+00,   2.18482396e+00,
                9.08642161e+00,   9.77799962e+00,   2.22979422e+00,
                3.29643733e+01,   5.16859345e-01,   1.71667860e+00,
                3.40167580e-01,   8.88081286e+00,   2.63347271e+00,
                9.15183172e+00,   4.55401115e+00,   1.45136503e+00,
                1.04398691e+00,   3.27403202e-01,   1.93629180e+01,
                2.46850178e+00,   6.51137388e+00,   1.87602130e+00,
                2.30249407e-01,   2.12860325e+01,   2.64534366e+01,
                1.30807409e+00,   9.18697349e-01,   1.62473697e+00,
                3.14213701e+01,   3.37126420e+01,   3.22762419e+01,
                2.59554875e+01,   5.27981057e-01,   7.71063086e-01,
                3.04035923e-01,   2.88389285e+00,   9.41646246e-01,
                5.64822749e+00,   1.49910443e+00,   1.82789427e+01,
                1.13737986e-01,   7.08807357e-01,   2.08717999e+00,
                2.29715614e-01,   9.06621411e-01,   2.36126907e+01,
                8.97296719e-01,   2.91354847e+01,   2.10988280e+01,
                3.14842627e+01,   1.98774153e+01,   2.85132309e+01,
                1.29737587e+01,   2.97048311e-01,   2.33224244e+00,
                8.40075597e+00,   1.68076145e+00,   2.02453664e+01,
                1.28409124e+00,   2.15510075e-02,   2.23409626e+00,
                2.03298567e-01,   1.38670655e+00,   2.43545442e-01,
                1.21267083e-01,   1.96208370e+01,   3.21912731e+00,
                6.81799155e-02,   6.62860118e+00,   1.47849991e+00,
                7.49036488e-01,   2.17927350e+00,   1.97904930e-01,
                3.62704051e+01,   8.73013024e-01,   8.51212215e-01,
                1.32032132e+00,   6.10246010e+00,   2.23890626e+00,
                1.25697837e+00,   3.57003267e+01,   1.03314888e+01,
                1.60802139e+01,   1.66946122e-03,   3.15050618e+00,
                1.08002618e+01,   3.86703175e+01,   1.60068577e+01,
                4.01512456e-01,   2.31319134e+00,   9.69827014e-01,
                1.15293650e+01,   1.61327636e+01,   1.59420647e-02,
                1.55714689e+00,   3.37196654e+00,   1.12044336e+01,
                1.12892597e+00,   2.58642984e+01,   7.35349159e+00,
                1.13783682e-01,   2.34667130e+00,   1.57673847e+01,
                1.89574864e-01,   1.91453868e+01,   6.17336922e+00,
                8.72294433e-01,   5.15065673e+00,   3.34494993e+01,
                8.89514744e-02,   1.54845033e+00,   1.35610392e+00,
                1.65936064e+01,   7.60979120e-01,   2.57141459e+00,
                2.47414980e+01,   1.57347479e+01,   3.72746629e-01,
                2.20757519e+01,   3.24212281e+01,   6.09674548e-01,
                2.86843593e+01,   1.30231555e+00,   1.36215187e+01,
                9.25953120e+00,   1.81301508e+00,   1.27674965e+00,
                5.36248018e-02,   6.96740447e+00,   4.08033542e+00,
                1.33363093e+01,   5.18791099e+00,   4.72079350e-01,
                7.17626184e-01,   5.44011833e-01,   1.18989236e+00,
                1.20057094e+00,   6.10576297e-01,   2.40169647e+01,
                4.95029428e-01,   1.49466581e+00,   3.77083192e-01,
                1.82716276e+00,   6.96503886e+00,   7.10117524e-01,
                9.92340595e-02,   6.37812524e-01,   3.53617505e-01,
                1.11307471e+01,   7.32266327e-01,   4.96472659e-01,
                1.83524757e+00,   1.70641384e+00,   1.26851665e+01,
                2.76890386e+01,   9.12808212e-01,   3.42119097e+01,
                2.65901850e+00,   1.26742898e+01,   2.65898561e+00,
                2.50771469e-01,   3.06727863e-01,   1.16870214e+01,
                2.21179375e-01,   1.29238446e+01,   2.58045464e+01,
                1.15519881e-01,   5.90417219e-01,   7.95072373e-01,
                1.56772395e-01,   1.39531545e+00,   2.02927914e+01,
                7.97857912e-01,   2.04566816e+00,   1.29108291e+00,
                2.01697387e+00,   3.44251674e+01,   3.55519615e+01,
                3.27508623e-01,   5.88194891e-01,   8.85991944e-01,
                9.93558396e-01,   2.27440175e-01,   2.62791341e+00,
                1.55151463e+00,   1.63527966e+00,   2.35080889e+00,
                3.37988114e+01,   2.07164136e+00,   9.79340760e-01,
                1.74804713e+00,   1.30444950e+00,   1.96054917e+01,
                4.79046935e+00,   2.57417845e+01,   8.94619952e-01,
                4.43033358e+00,   1.08150244e+01,   1.18350296e+01,
                1.60772931e-01,   5.15124017e-01,   9.76321826e+00,
                1.64480696e+00,   1.69632969e-01,   4.20743981e-01,
                2.21104363e+00,   1.07589595e+00,   3.78231078e-01,
                2.02405969e+00,   2.18085541e+01,   9.50920742e+00,
                6.60744251e-01,   3.93787853e+01,   1.60209418e+01,
                6.60871727e+00,   3.39494019e+01,   4.49375360e+01,
                4.64950695e+00,   9.71055295e+00,   3.20971671e-01,
                1.10818392e+00,   1.99315480e-01,   4.19004862e+01,
                1.03890999e-01,   1.74643601e+01,   1.63144187e+01,
                2.35443842e-01,   5.00137167e-01,   3.64341944e-01,
                1.90967511e+01,   1.12855771e+01,   2.52397428e+00,
                2.79241291e+01,   3.83892161e+00,   2.61402469e-01,
                1.37342357e-01,   5.12469662e-01,   9.19305329e-01,
                1.53110031e+00,   1.43615812e+01,   1.78252980e+00,
                1.13218709e+00,   4.00342530e+01,   2.34997779e+01,
                5.88879526e-01,   9.86514415e+00,   1.87517864e+01,
                3.25043367e-01,   1.18919811e+01,   8.47365630e+00,
                1.12470160e+01,   2.77562151e+01,   1.86391379e+00,
                3.27425845e-01,   2.11571003e+01,   2.53991010e+01,
                3.97760453e-01,   1.06704603e+01,   3.01672920e+01,
                1.83119637e+00,   3.62648433e+00,   5.19747092e+00,
                2.19537251e+01,   3.95594746e-01,   2.87184168e+00,
                2.12122678e+00,   2.62810328e+01,   9.80798027e+00,
                3.55592166e-01,   2.96271969e+01,   2.24456382e+01,
                1.93808600e+01,   3.28079171e-01,   3.74512925e+01,
                1.11663939e+01,   3.13875733e-01,   1.19331289e+01,
                4.78837722e-01,   1.29589063e+00,   1.45861077e+00,
                2.91210193e+01,   1.02054584e+00,   4.09772441e-01,
                4.07958740e+00,   1.33130325e+01,   3.14977987e+01,
                1.11465218e+00,   1.52213801e+00,   1.15518345e+00,
                2.61367153e+01,   1.94251930e+01,   2.00103782e+00,
                9.89913021e-01,   1.99525864e-01,   6.09734075e-01,
                5.86125463e-01,   1.22823088e+01,   5.57427008e-01,
                8.48812810e-01,   8.64662291e-01,   2.81621083e+00,
                1.99151306e+01,   1.86999752e+00,   7.52603322e-01,
                7.08211159e+00,   1.23041182e+01,   2.55644240e+00,
                1.77195003e+01,   2.08672770e+00,   1.60333811e+01,
                2.66919042e-01,   2.55476643e-01,   1.13973757e+00,
                1.48134434e-01,   2.09332442e+01,   8.50616950e+00,
                6.16855198e-01,   1.19102141e+00,   6.12022060e-01,
                3.35478126e-01,   7.20507059e-01,   1.42417171e-01,
                9.80984342e+00,   2.94480016e-01,   1.87160106e+00,
                3.58848781e-01,   5.02125927e-01,   1.69212345e-01,
                1.20945802e-01,   2.59027526e+01,   6.88863749e+00,
                2.50838658e+01,   7.08849091e-01,   7.93207515e+00,
                1.42913658e+01,   2.76057942e-01,   4.52965480e-01,
                1.33538269e+00,   2.01475037e+00,   1.73432569e+01,
                2.18534418e+00,   6.24203262e-01,   1.70345570e+00,
                3.84499191e+01,   4.30021655e-01,   1.96767688e+00,
                1.96515305e+00,   2.12413709e+01,   1.98983759e+00,
                4.18538855e+01,   9.95450383e+00,   1.79213904e+00,
                7.84163945e-01,   8.76632864e-01,   2.17972316e+01,
                1.82865771e+00,   1.30580263e+01,   4.68181379e+00,
                2.79449720e+00,   9.16168869e-01,   1.54605774e+01,
                1.08307539e+01,   1.88264801e+01,   1.67800829e+00,
                3.08038062e-01,   1.63204671e+00,   6.89054950e-01,
                4.08411836e+01,   2.09289343e-01,   2.42426507e+01,
                8.00731790e+00,   3.64442878e+01,   2.70548994e+01,
                1.50442055e+00,   1.08804493e+01,   5.81672409e-01,
                1.54058978e+01,   1.24495197e-01,   3.38399309e+01,
                5.47689759e-01,   2.37537805e+00,   3.12478538e+00,
                1.24564130e+01,   2.29876024e+00,   3.52772607e-01,
                9.32473350e-01,   4.46813826e-02,   1.62305828e+00,
                1.84032236e+00,   1.37354133e+00,   1.70998549e+01,
                2.18446928e-01,   2.13424116e+01,   2.05730055e+01,
                3.03103370e+01,   2.58406786e+01,   2.35471907e-01,
                2.54345821e+00,   5.84631254e-01,   2.20600755e+01,
                2.12676510e+00,   7.93968009e-01,   9.73557052e+00,
                4.31927765e+00,   5.76336349e+00,   7.46593823e-01,
                2.99116209e-01,   8.90945877e-01,   1.64073889e+00,
                8.19320788e+00,   6.00155989e-02,   3.31050737e+01,
                1.92693589e+01,   7.73439184e-02,   1.04661636e+00,
                5.23465201e+00,   9.66405362e-01,   2.34057943e-01,
                1.93664932e+01,   5.46869449e+00,   5.84895245e+00,
                1.95350412e+01,   1.16880368e+00,   1.14361102e+00,
                2.85444216e+01,   3.92041943e+01,   6.88080396e-01,
                1.59848169e+00,   1.09471761e+00,   6.99026233e-01,
                9.32176005e-01,   4.04742648e-01,   3.80847341e+01,
                1.49198298e+00,   1.58050144e+01,   2.14860302e+01,
                7.31092704e+00,   7.42401602e+00,   1.74788468e+00,
                1.67793535e+00,   2.86767389e+00,   2.92811569e+01,
                3.57795141e+01,   1.52853044e+01,   7.08761883e+00,
                7.38839601e-01,   2.25355344e+01,   8.08128952e-01,
                6.13442082e+00,   6.60209508e-01,   2.18090248e+00,
                4.26641062e-01,   4.93345347e+01,   1.07964445e+01,
                2.48693457e-02,   4.20224489e-02,   3.95155212e+01,
                9.40879169e+00,   3.13996041e-02,   1.48193822e+00,
                1.56008531e+01,   2.60850582e+00,   4.90046976e-01,
                7.06855721e-01,   1.41619814e+01,   5.48053739e-01,
                2.48332944e+01,   2.09114451e+01,   3.11705081e+01,
                1.01615302e+01,   2.17765743e+00,   4.03006703e+00,
                2.40961181e+01,   1.69030162e+00,   2.54584763e+01,
                1.89717587e+00,   1.82607266e+00,   4.51831104e+00,
                8.45223551e-01,   2.70726438e-01,   7.59992432e-02,
                2.43850862e+00,   1.14313956e+00,   1.21143926e+01,
                2.96016468e+01,   8.30657671e-01,   6.60029594e-01,
                7.55716553e-01,   3.13187716e+00,   3.58109490e+01,
                1.31221726e+00,   4.73042389e+00,   1.97166475e+01,
                3.82611771e+01,   1.30382683e+01,   1.07155668e+00,
                6.27769169e-01,   2.36672266e+01,   1.75933230e+01,
                1.17312836e+00,   1.51896851e+01,   2.49950917e+01,
                1.38338512e+00,   7.23206053e+00,   1.83700204e+00,
                1.51665656e+01,   1.21268295e+00,   1.99624406e+00,
                1.69631029e+00,   4.36453156e+01,   1.14446729e+00,
                1.00324210e-01,   5.00832244e-01,   2.11361292e+01,
                8.26468100e+00,   2.38346521e+00,   2.11711221e+00,
                6.04140293e-01,   3.88989389e-01,   2.95145033e+01,
                2.99705038e+00,   1.23933659e+01,   5.83931723e+00,
                2.93697523e+00,   4.24904116e+00,   1.24151066e+01,
                1.84984859e+00,   9.81482013e-01,   3.88642268e+01,
                4.07850150e+01,   3.91872565e+01,   3.72525178e-02,
                2.08198318e+00,   1.09437193e+00,   5.27212105e-01,
                3.02464786e+01,   1.98980307e+00,   1.49085548e+00,
                8.38421274e+00,   1.36519654e+00,   9.77500282e+00,
                1.42974440e+00,   5.52612863e-01,   6.35057248e-01,
                7.32787527e+00,   1.38355495e+01,   4.39212825e+00,
                1.01280732e-01,   6.27056642e-01,   1.41719189e+00,
                1.08557622e+01,   1.90880448e+00,   5.60659280e+00,
                7.88334727e+00,   9.40391569e-01,   3.05633886e-01,
                2.44879634e+00,   1.09183317e+00,   2.04657108e+00,
                2.32587813e+01,   7.32556600e-01,   1.28543880e+01,
                5.39710015e+00,   1.18725898e+01,   2.11806020e+00,
                1.12519609e+00,   1.65595678e+01,   2.32136908e+01,
                1.34892610e+01,   1.34333659e+00,   1.34562393e+01,
                7.42351739e-02,   3.66161333e+01,   1.61501685e+00,
                7.12307617e-01,   1.59340734e+01,   2.86787590e+00,
                1.27786226e+00,   4.46107852e+01,   6.76724524e-02,
                5.72342115e+00,   5.92262205e+00,   2.49304498e+01,
                3.99422345e+00,   1.75034367e+01,   9.19798809e-01,
                5.34314842e-01,   2.52792686e-01,   3.96557687e-01,
                1.68386298e+00,   7.05462580e+00,   1.24115793e+00,
                2.51353666e+01,   1.06432484e+00,   1.90746417e+00,
                2.48671072e+00,   3.01451875e-01,   1.60088458e+01,
                1.87844316e+01,   1.55026952e+00,   7.92387662e-01,
                3.16998655e+00,   3.60242472e+00,   2.35987346e-01,
                1.50529756e+01,   1.67764975e+01,   1.64514293e+01,
                2.78827718e-01,   3.14404223e-01,   1.09529610e+01,
                2.67105454e+01,   4.67258932e+01,   3.54764082e+01,
                1.98882098e+01,   1.63042321e-01,   1.88718117e+01,
                1.25864447e+01,   1.20093605e-01,   1.76519248e+01,
                2.82273214e+00,   2.89288992e-01,   2.37138339e+00,
                2.09524639e+00,   1.28320747e+00,   1.71913221e+01,
                2.18137684e+00,   5.30589581e+00,   1.70779437e+00,
                3.32471744e+01,   4.41211786e+01,   1.14506109e+01,
                1.92932377e+00,   2.76150160e-02,   1.56757914e+01,
                3.22018442e+01,   8.53138684e+00,   5.10355683e+00,
                2.85638550e+01,   1.56726595e-01,   2.75729755e+00,
                1.06375388e+00,   1.04780701e+00,   6.62828980e+00,
                3.15564415e+00,   4.15015443e-01,   9.23969503e-01,
                1.56956511e+00,   2.03670911e+00,   1.48472604e+01,
                4.27121811e-01,   1.93803684e-01,   1.20938652e+01,
                1.81012768e+01,   3.06668514e+01,   1.52559193e+01,
                7.79645498e+00,   2.14777195e+00,   2.34679895e+01,
                4.39594772e-01,   1.39288125e-01,   2.39724498e+00,
                1.03116715e+00,   4.16720922e-01,   2.79356953e+01,
                1.32944408e+00,   1.70512133e+00,   1.76348926e+01,
                2.40239857e-01,   8.35068861e+00,   2.46155064e+01,
                1.35405206e+00,   1.50641529e+00,   1.80097882e-01,
                6.08267923e+00,   7.79103831e-02,   1.90277046e+00,
                1.34211469e+00,   1.28520861e+01,   1.58645073e+00,
                2.54913498e+01,   1.76137264e+00,   7.49477720e+00,
                4.53190168e-01,   3.59246824e+00,   1.68965313e+01,
                9.65257440e-01,   5.54019094e-01,   3.29938996e+01,
                5.00599707e-01,   7.33324249e-01,   1.76829314e+00,
                1.07978996e+01,   4.12408932e+00,   1.55683398e+00,
                5.97774451e-01,   3.45778155e-01,   2.91719587e+01,
                2.57381885e+01,   5.11391677e-02,   1.81093425e+00,
                3.60193896e+00,   1.56370806e-01,   1.87578690e-01,
                1.01007418e+01,   1.23176296e+00,   6.23607357e-01,
                9.69809441e+00,   2.01583497e+00,   9.53865467e+00,
                5.70110959e-01,   1.66128695e-01,   2.90285673e-01,
                2.16613160e+01,   8.71868523e-02,   2.16171643e+00,
                5.55161844e-01,   1.45131985e+01,   7.94541135e-01,
                1.10959298e+01,   2.52718251e+01,   5.25944270e-01,
                7.95084037e-02,   2.95907442e+01,   3.58665109e-01,
                4.79265595e-01,   1.66974716e+00,   5.26149289e+00,
                4.44683410e+00,   4.09395512e-01,   3.30562842e+01,
                2.74206245e+01,   1.16549535e+01,   9.22417134e+00,
                1.22021862e+00,   4.04369197e-02,   9.13013800e-02,
                3.56001538e+01,   9.17811579e-02,   1.16079894e+01,
                7.26784223e-01,   1.81126019e+00,   8.20449516e-01,
                9.59524009e-01,   6.32900987e+00,   3.71487914e-01,
                2.82920371e+00,   4.57624720e+01,   2.77769565e+01,
                7.22165383e-01,   1.39172584e+00,   4.25164900e+00,
                7.39457252e+00,   1.29054091e+00,   8.65232874e-01,
                7.20842624e-01,   1.86838454e-01,   2.16198985e+00,
                4.84800044e+00,   3.12580317e-02,   1.04907541e+00,
                1.19303788e+00,   1.72495977e+00,   2.60129548e+00,
                2.90536220e-01,   1.10971453e+01,   6.32906060e-01,
                3.49015594e-01,   1.66795670e+00,   1.73327403e-01,
                1.29814971e-01,   5.88282519e-02,   2.75236907e+01,
                1.57485072e+00,   2.30754197e+00,   9.69976164e-01,
                8.47211134e-02,   3.16985706e+01,   1.49867081e+00,
                1.49992667e+01,   4.09282831e-01,   8.98600432e-01,
                1.91979711e+00,   3.37287091e+00,   2.46440145e+01,
                3.78382267e+01,   6.75098110e-01,   4.31109946e-01,
                3.76378619e-01,   4.90704459e-01,   3.06599806e+01,
                2.34523364e+01,   6.38520949e-01,   3.77880377e-01,
                7.38155694e-01,   1.05313209e-01,   6.11263012e-02,
                1.08792695e+01,   1.94733636e+00,   1.21405659e+00,
                1.93003365e+01,   7.23899453e-01,   2.90093302e+01,
                1.38113622e+01,   1.41804444e-01,   2.15489268e+01,
                1.43710743e+01,   2.03637024e+01,   4.04527109e-01,
                3.53835261e+00,   9.55420552e+00,   3.65626547e+01,
                3.05051876e+00,   1.57384118e-01,   2.22125346e+01,
                6.60156120e-01,   1.21098664e+00,   1.82168031e+01,
                2.85897609e+00,   3.99834443e+01,   1.87125788e+01,
                8.80009549e-01,   1.14171068e+00,   3.57459545e+01,
                3.63836995e+01,   2.00140466e+01,   1.38050648e+01,
                2.32630887e-01])
```

Question 3. Why do we sum up the absolute values of the differences in trait values, rather than just summing up the differences?

With no absolute values, the differences in trait values could be negative numbers and the sum may not be meaningful.

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 USD, 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 USD in cost-to-attend results in a dissimilarity of 900 x 0.001, 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 0.001 or 2.*
4. Now sum the 1000 resulting numbers.

Question 4. 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.

In [4]:
```
weights = Table.read_table("~/DS_113_S23/HW/HW_3/weights.csv").column("Weigh

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

Out[4]:
```
505.98313211458793
```

2. Period Plots

Below is a plot that compares the number of characters to the number of periods in each chapter of Little Women. Each point represents one chapter.

Question 1. About how many periods are in the chapter with the most characters per period?

In [5]:
```
periods_in_most_characters_per_period = 60
## the chapter with about 60 period has the most characters per period (1200
## other plots have less than 200 characters per period
```

Question 2. About how many periods are in the chapter with the most characters?

In [6]:
```
periods_in_most_characters = 390
## a period between 350 and 400 has the highest point, about 41000 characte
```

3. 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 1. The data are in a CSV file called `unemployment.csv`. Load that file into a table called `unemployment`.

```
In [7]: unemployment = Table.read_table("~/DS_113_S23/HW/HW_3/unemployment.csv")
        unemployment
```

Out[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)

Question 2. Sort the data in decreasing order by NEI, naming the sorted table `by_nei`. Create another table called `by_nei_pter` that's sorted in decreasing order by NEI-PTER instead.

```
In [8]: ## use sort(,descending=True) to sort the data in decreasing order
        by_nei = unemployment.sort("NEI", descending=True)
        by_nei_pter = unemployment.sort("NEI-PTER", descending=True)
```

Question 3. Use `take` to make a table containing the data for the 10 quarters when NEI was greatest. Call that table `greatest_nei`.

```
In [9]: ## use np.arange to assign the first 10 rows
        greatest_nei = by_nei.take(np.arange(0, 10))
        greatest_nei
```

Out[9]:

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

Question 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 proportion 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.

In [11]:
```
pter = unemployment["NEI-PTER"]-unemployment["NEI"]
pter
```

Out[11]:
```
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])
```

Question 5. Add `pter` as a column to `unemployment` (named "PTER") and sort the resulting table by that column in decreasing order. Call the table `by_pter`.

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

In [13]:
```
## use with_column to add another column
## sort() to sort the table in decreasing order
by_pter = unemployment.with_column("PTER",pter).sort("PTER",descending=True)
by_pter
```

Out[13]:

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)

Question 6. Does it seem true that the PTER rate was very high during the Great Recession, compared to other periods in the dataset? **Also**, is the sorted table the best way to find this out, or can you think of other ways to look at the data to answer this question?

The PTER rate is not significantly high during the Great Depression compared to other periods. However, it seems that we can conclude that the Great Depression might have affected the rate of the PTER, because all of the highest rates of the PTER are during the Great Depression or right after the time. I think the line plot or the dot plot would be more helpful than the sorted table to find our that the PTER rate increased because of the Great Depression. To know how the data changed, the plots would be better to see the trend.

4. 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. 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 [17]:
```
# Just run this cell.
complaints = Table.read_table("~/DS_113_S23/HW/HW_3/complaints.csv")
complaints
```

Out[17]:

company	company_public_response	company_response	complaint_id	complaint_what_ha
TransUnion Intermediate Holdings, Inc.	Company has responded to the consumer and the CFPB and c ...	Closed with explanation	1920073	
TransUnion Intermediate Holdings, Inc.	Company has responded to the consumer and the CFPB and c ...	Closed with explanation	1914777	
Bank of America	Company has responded to the consumer and the CFPB and c ...	Closed with explanation	1907306	I became aware c charges on a Am
Finance of America Reverse LLC	Company believes it acted appropriately as authorized by ...	Closed with explanation	1919055	I applied for a mortgage and evert
Acceptance Solutions Group, INC	Company believes it acted appropriately as authorized by ...	Closed with explanation	1908628	Keeps calling num are not mine. And t
Equifax	(None)	Closed with explanation	1909176	
TransUnion Intermediate Holdings, Inc.	Company has responded to the consumer and the CFPB and c ...	Closed with explanation	1914477	When I enter my information to re
Encore Capital Group	(None)	Closed with non-monetary relief	1919937	
Nationstar Mortgage	(None)	Closed with explanation	1920517	I am livid with Natior refusing to work w
Convergent Resources, Inc.	(None)	Closed with explanation	1920464	

... (15021 rows omitted)

Question 1. Financial companies offer a variety of products. How many complaints were made against each kind of product? Make a table called `complaints_per_product` with one row per product category and 2 columns: "product" (the name of the product) and "number of complaints" (the number of complaints made against that kind of product).

In [20]:
```
## use .group() to group the table by the product
complaints_per_product = complaints.group("product")
complaints_per_product
```

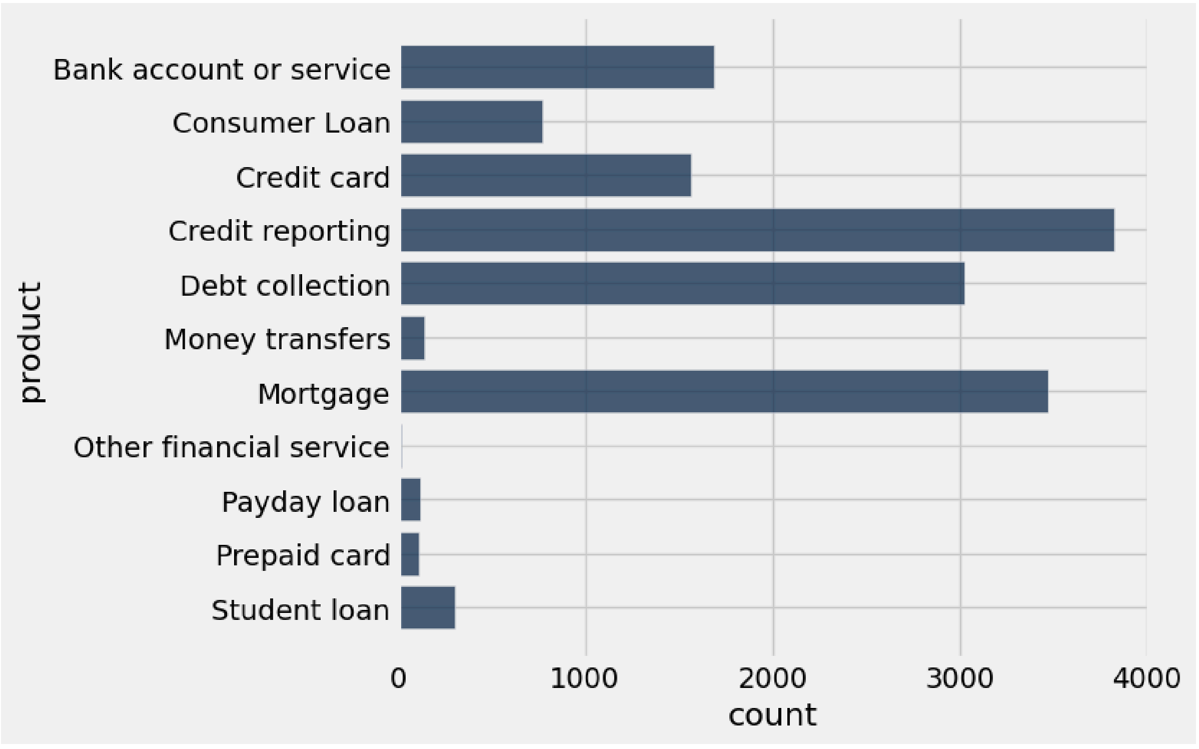
Out[20]:

product	count
Bank account or service	1687
Consumer Loan	775
Credit card	1566
Credit reporting	3820
Debt collection	3022
Money transfers	142
Mortgage	3468
Other financial service	16
Payday loan	119
Prepaid card	110

... (1 rows omitted)

Question 2. Make a bar chart showing how many complaints were made about each product category.

In [21]:
```
## use .barh() to make a bar chart
complaints_per_product.barh("product","count")
```



Question 3. Make a table of the number of complaints made against each *company*. Call it `complaints_per_company`. It should have one row per company and 2 columns: "company" (the name of the company) and "number of complaints" (the number of complaints made against that company).

In [22]:
```
complaints_per_company = complaints.group("company")
complaints_per_company
```

Out[22]:

company	count
1st Preference Mortgage	2
21st Mortgage Corporation	7
2288984 Ontario Inc.	3
360 Mortgage	1
3rd Generation, Inc.	1
4M Collections, LLC	1
A.R.M. Solutions, Inc.	2
AC Autopay, LLC	1
ACE Cash Express Inc.	21
ACS Education Services	8

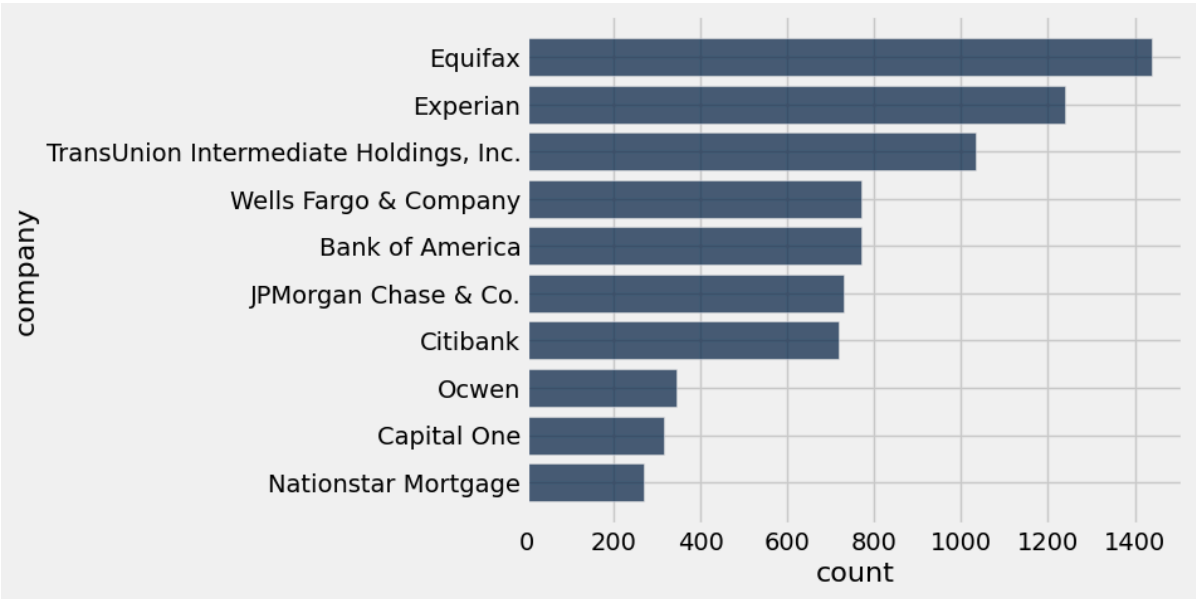
... (1131 rows omitted)

Question 4. It wouldn't be a good idea to make a bar chart of that data. (Don't try it!) Why not?

Because there are more than a thousand companies. If we make a bar chart, there will be 1141 bars in the chart.

Question 5. Make a bar chart of just the companies with the most complaints. Specifically, make a chart that displays the number of complaints against the companies with the 10 most complaints.

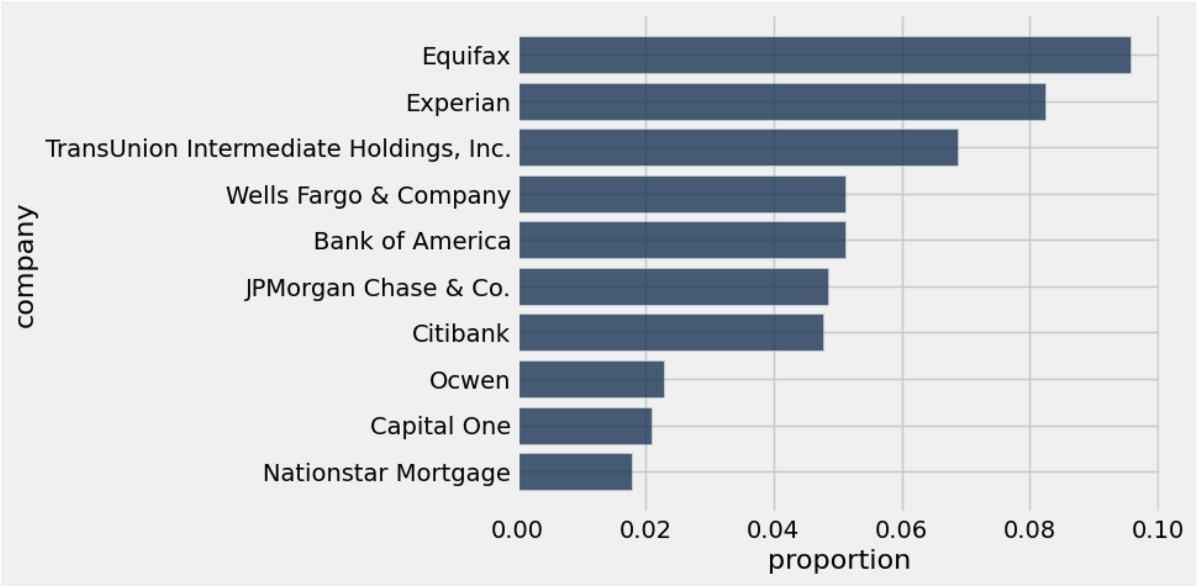
In [24]:
```
## sort the table in decreasing order
## use .take() to choose only top 10
company_10most_complaints = complaints_per_company.sort("count", descending=
## use .barh() to make a bar chart
company_10most_complaints.barh("company","count")
```



Question 6. Make a bar chart like the one above, with one difference: The size of each company's bar should be the *proportion* (among *all complaints* made against any company in `complaints`) that were made against that company.

Note: Graphs aren't very useful without accurate labels. Make sure that the text on the horizontal axis of the graph makes sense.

In [27]:
```
## make a table with the proportion of the complaints
complaint_proportion = complaints_per_company.with_columns("proportion",(com
## sort the table in decreasing order
## use .take() to choose only top 10
complaint_proportion_top10 = complaint_proportion.sort("count", descending=T
## use .barh() to make a bar chart
complaint_proportion_top10.barh("company","proportion")
```


company
Equifax
Experian
TransUnion Intermediate Holdings, Inc.
Wells Fargo & Company
Bank of America
JPMorgan Chase & Co.
Citibank
Ocwen
Capital One
Nationstar Mortgage
proportion