Economics 1660: Big Data
PS 0: Programming for Large Data Sets

Brown University
Prof. Daniel Björkegren

The spread of information technology has generated massive amounts of data, even in developing countries. However, it is cumbersome, and sometimes infeasible, to analyze this data with tools that you may be comfortable with, like Stata or Excel. This assignment will introduce you to scientific programming with Python.

Python is a multipurpose language, with a lot of advantages: it's free, flexible, has a lot of packages, and many people use it. In particular, it can do many things Matlab can do.

This assignment will walk you through some of the features you will use in the rest of the course. You'll fill in this packet with your answers and turn it in. You may discuss the tutorial with classmates, but each person must do their own work.

When doing your own work, you'll have to teach yourself a lot of things. When you're going through this assignment, if you have technical questions I encourage you to look for an answer on google or stackoverflow.com.

I. PRELIMINARIES

First you'll need to install python 3.x and packages for scientific computation on your computer.

Recommended: download the Anaconda distribution of python, which comes with all the packages you'll need, here: http://continuum.io/downloads

Or, if you have a Mac or some other flavor of unix, your computer probably already has python, but it may be an old version (2.x). You can also install newer versions manually:
Python: https://www.python.org/downloads/
SciPy and NumPy: http://www.scipy.org/install.html

Download Data

Kiva, a crowdsource microfinance platform, has started posting its loan data online. It's easy to download in XML format, but since you're probably not familiar with that format I've included scripts to process it and convert it to a Comma Separated Value (CSV) text file. The original data is posted here: http://build.kiva.org.

The data and code you'll be using is in the package **PS0_package.zip** on the website. Download this into a folder on your computer and then unzip it.
Then run the script 0_CreateDataSet.py to create the data files that you'll need for this and other assignments:

From a terminal you can type "python 0_CreateDataSet.py"
It will take a few minutes to download and process the data.
(Hint: on a Mac you can open Applications>Utilities>Terminal, and on Windows you can click Start>Run and type "cmd". Then to get in the right folder, type "cd X" where X is the path to the files you saved.)

We'll be using data on loans. The website also has data on lenders.

II. BASICS

As you can see, the data set is somewhat large. While it's easy to have a file like that on your hard drive, programs like Stata read all of the data into memory (RAM) at once. If the data set is large enough this will fail. In python we can read the data in parts, and only store what we need in memory, so you can work with much larger datasets more flexibly.

First, let's start with a simple script to count the number of observations:

1_CountLines.py Source	Line-by-Line Explanation
import csv fileReader = open("tables/loans_B_unlabeled.csv", "rt", encoding="utf8") csvReader = csv.reader(fileReader) cHeader = next(csvReader) nObservations = 0 for acRow in csvReader: nObservations += 1 fileReader.close() print("Total Number of Loans", nObservations)	

(Note: I use the package 'csv' which makes it easy to read CSV files; this data set has some complicated free responses that would make simpler code fail.)

To run this script, open a terminal to the folder you saved the files from **PS0_package.zip**.

Now, enter:

python –i 1_CountLines.py

to run the script. The –i flag will make python stay open after the script has run ("interactive mode"). You can then run another command, or inspect a variable. When you're done you can press Control+D to end the session.

How many observations are there? _____

Open the script in a text editor.
Figure out what each line does, and write your explanation for each line in the blanks on the previous page. (You'll do this for each script throughout the packet.)

Hints:

1. ***Python is very picky about indentation**. Within a loop or conditional statement, every line of code needs to be indented by the same number of spaces or tabs. Statements not within conditionals mustn't be indented. See for example: http://www.quantumwise.com/documents/manuals/ATK-2008.10/chap.pythonbas.html*
2. *You can also open python in interactive mode by running "python" in the terminal. Then you can paste in the lines of code one by one to see what they do. In interactive mode you can enter the name of a variable and then enter to see its value.*
3. *There's a lot of python documentation on the web. You can google "python" and then the name of the command to see what it does.*

Why do I skip one line before starting to count observations?

We have a lot of observations. One way to deal with large data sets is to simply sample a subset of the observations, and then use tools we're more familiar with. Now we'll use python to extract a random subset of observations. **Run and explain the following code:**

2_Sample.py Source	Line-by-Line Explanation

`import csv`
`import random`
 	
`fileReader = open("tables/loans_B_unlabeled.csv", "r",`
`encoding="utf8")`
`csvReader  = csv.reader(fileReader)`
 	
`fileWriter = open("tables/loans_B_unlabeled_subset.csv", "w",`
`encoding="utf8", newline='')`
`csvWriter  = csv.writer(fileWriter)`
 	
`acHeader = next(csvReader)`
`csvWriter.writerow(acHeader)`
 	
`for acRow in csvReader:`
`  if random.random() < 0.01:`
`    csvWriter.writerow(acRow)`
 	
`fileReader.close()`
`fileWriter.close()`	

What fraction of observations does this sample? _____
Write how you'd change the above code to sample:

a. **0.5% of the observations**
b. **the first 1,000 observations**
c. **the last 1,000 observations**

With this smaller sample, we could do some analysis in Stata or Excel. Let's find out how the loans are distributed across countries.

Open the sample in Stata or Excel. *(Hint: in Stata you can use "File > Import > ASCII data created by a spreadsheet")*

In State you can use the command *tab* to find out how many loans there are per country in this sample.
How many loans to Haiti are in the sample? _____
How many loans to Haiti do you predict would be in the full data set? _____

We can compute the actual number of loans made out to Haiti using python.

Read and explain the code:

3_FieldCounts.py Source	Explanation

import csv
 	
fileReader = open("tables/loans_B_unlabeled_subset.csv", "rt", encoding="utf8")
csvReader = csv.DictReader(fileReader)
 	
dTable = dict()
for dcObservation in csvReader:
  cCountry = dcObservation["location_country"]
 	
  if cCountry not in dTable:
  dTable[cCountry] = 1
  else:
  dTable[cCountry] += 1
 	
fileReader.close()
print("Country Distribution: ", dTable)	

Change the code to count the number of loans by sector.
Hint: to see the field names, you can view the first line of the sample in a text editor.

What fraction of loans are to the arts sector? _____

Now, change the code to count the total *value* of loans given out to each country. What is the value of loans given out to Haiti? _____
Hint: You'll need to use the statement float(x), which *returns* x *converted into a floating point number.*

III. USING PYTHON FOR STATISTICS

Now, let's compute some statistics. We'll loop through the data, and instead of storing part of it in a dictionary, we'll store it in an array. We'll also use the numpy and scipy libraries, which have functionality similar to Matlab.

Starting code is on the next page.

4_Statistics.py Source	Explanation

import csv
import numpy as np
import scipy as sp
import scipy.stats
import numpy.linalg
 	
fileReader = open("tables/loans_B_unlabeled.csv", "rt",
encoding="utf8")
csvReader = csv.DictReader(fileReader)
 	
afLoanAmount = list()
afPictured = list()
 	
for dcObservation in csvReader:
  fLoanAmount = float(dcObservation["loan_amount"])
 	
  try:
    fPictured = float(dcObservation["pictured"])
  except ValueError:
    fPictured = 0
 	
  afLoanAmount.append(fLoanAmount)
  afPictured.append(fPictured)
 	
fileReader.close()
 	
afLoanAmount = np.array(afLoanAmount)
afPictured = np.array(afPictured)
 	
print("Mean(Loan Amount): ", np.mean(afLoanAmount))	

Run this script. Then add code and compute the following:

- Standard deviation of loan amount: _____
  Hint: use the function np.std

- Correlation between loan amount and whether picture is taken or not: _____
  Hint: use the function sp.stats.pearsonr

- OLS Regression of the loan amount on pictured: loan = _____ + _____*pictured
  Hint: use the function np.linalg.lstsq.
  The code np.linalg.lstsq(np.vstack([X, np.ones(len(X))]).T, Y)
  runs an OLS regression of Y on X and a constant.

Hint: if you know Matlab, there is a guide that shows the differences in syntax and behavior here:
http://wiki.scipy.org/NumPy_for_Matlab_Users

Explain in a couple sentences what this regression captures? Was the result what you expected? Why or why not?

These methods require storing one whole column of data in memory (an array of floats). If the data set is large enough this won't be feasible.

Modify the script above to calculate the mean by only storing two numeric variables.
(Hint: as you read through the data you'll need to keep track of the number of observations as well as the sum of the values. For example if we were finding the minimum of a long array, instead of storing the entire array, you can access each entry in order, and store it if it is smaller than the current minimum. Thus you only need to store a single number instead of the entire array at any given time. After looping through the entire array you have the overall minimum.)

What's the minimum number of numeric variables you'd need to store in order to compute the standard deviation? *(Hint: you don't need many, and your code should only loop through the data once. If you need guidance, you can start by writing out the mathematical expression for standard deviation.)*
Modify the script to compute the standard deviation while storing this minimum number of variables in memory.

V. TEXT ANALYSIS

One of the other benefits of a general purpose language like python is that it allows you to do all sorts of processing, like text. The loan data actually includes a lot of open responses.

Many loans were used for chickens, but the only way we'd know how many is if we looked at the full text for each response. Let's write a program to do that.

We'll start with the code from 3_FieldCounts.py. We'll make a modification to the code to keep track of the number of loans that have the word 'chicken' in their description.

I've written out a framework for the code in the following source. You'll need to write some lines in the middle loop.

5_CountKeyword.py Source

```
import csv

fileReader = open("tables/loans_B_unlabeled_subset.csv", "rt", encoding="utf8")
csvReader  = csv.DictReader(fileReader)

nIncludeChicken = 0
nTotal          = 0
for dcObservation in csvReader:

  ...
    nIncludeChicken += 1

  nTotal += 1


fileReader.close()
print("Number including chicken: ", nIncludeChicken)
```

Hint: you'll want to use something like this syntax somewhere:

"chicken" in x.lower()

*this returns True if the string "**chicken**" is in the string stored in the variable **x**, regardless of whether it is lower or uppercase. It returns False otherwise.*

How many loans have the word 'chicken' in their description? _____

That was simple text processing, but we can do much more. A lot of researchers do '**sentiment analysis**' to figure out the emotions behind what people write on social media. We can do a really simple form of that. Let's check the sentiment of the text description recorded when each loan was granted.

We'll create a list of positive and negative words that could be associated with a comment. Then, for each loan we'll compute a sentiment score, where each positive word is worth +1, and each negative word is worth -1.

One thing that's a bit tricky is that each loan may have multiple comments (each separated by the character "|"). For now we'll compute one sentiment for each loan.

I've included most of the code below, but it is missing the code to compute negative sentiment.

Add the code to compute negative sentiment and explain any lines that appear new.

6_Sentiment.py Source	Explanation

```
import csv
import numpy as np
import scipy as sp
import scipy.stats
fileReader = open("tables/loans_B_unlabeled_subset.csv", "rt", encoding="utf8")
csvReader  = csv.DictReader(fileReader)

acPositive  = ["progress", "success", "profit", "well", "good"  ]
acNegative  = ["patience", "fear", "wait", "unexpected", "delay"]

anSentiment = []
for dcObservation in csvReader:
  acRepaymentComments = dcObservation["description_texts_en"].split("|")
  nSentiment = 0

  for cRepaymentComment in acRepaymentComments:
    for cWord in acPositive:
      nSentiment += cRepaymentComment.count(cWord)

    # TODO

  anSentiment.append( nSentiment )

fileReader.close()

print("Sentiment Frequency")
print(sp.stats.itemfreq( anSentiment))
```

How many loans are there with a sentiment score of -3? ____

Optional: If you're interested in sentiment analysis, a more robust way to do this would be to figure out the most predictive positive and negative keywords by training a model. In machine learning this would be called training a classifier. The Natural Language ToolKit (NLTK) package can help.

VI. ADDITIONAL RESOURCES

Here are some resources for learning more python:

- http://www.codecademy.com/en/tracks/python
- http://www.learnpython.org
- The book "Learning Python" by Mark Lutz