Introducing Jupyter and Agate

Welcome to the new world of data journalism, where reliability, replicability and transparency are key elements of what you do. Using the methods we will use for the rest of the semester will go a long way toward ensuring you clearly understand what you are doing, are doing the analysis you think you are doing, and are able to show others what you've done to arrive at your answer.

> ### Note: You should have the Agate docs open

We're going to use Agate to do similar analysis to what we've already done. The difference is that instead of pointing and clicking, you're going to write some very simple code. And you'll have a record of it when you're done so you can see what you've done.

Advantages	Disadvantages
You must explicitly do anything in code.	Simple things can be a struggle starting out.
Each step is documented.	A lot of overhead if you're just experimenting
Work is in memory. No alterations to the underlying dataset.	If you forget to save data changes back out to a new file, you lose them.

Simple analysis with Agate

Let's look at a table of some basic numbers about Big 10 universities. Download the file and fire up Jupyter Notebook.

The first thing we always have to do is import any libraries we're going to use. In this case, we need Agate and we're going to silence those pesky warnings.

```
import warnings
warnings.filterwarnings('ignore')

import agate
```

The second thing we're going to do is get some data to work with. You do that by creating a variable and setting it equal to a set of Agate commands.

```
big10 = agate.Table.from_csv('Data/colleges.csv')
```

If you've never done anything like this before, let's break this down a little. What we're doing is creating a variable called `big10` and setting it equal to some stuff. What it says is use `agate` (dot) use a thing called `Table` from within agate (dot) use a function in Table called `from_csv` and then get a file called colleges.csv from a folder called Data. Now, yours will probably be different than this. In the single quote marks is supposed to be a file path. The easiest thing to do -- but not always the best thing to do -- is to put the colleges.cv file in the same directory that you're working in. Or, put it in your home directory (where Documents and Downloads and folders like that are). Then, your file path is '~/colleges.csv' where the tilde character is a shortcut for your home directory.

If you've executed that and you get no errors, that means you've got a data tabled called big10 now. Let's take a look at it.

```
print(big10)
```

As you can see, that's not quite what you were expecting. It's the table STRUCTURE. It's the names of the fields and what kind of data you have in them, at least as Agate has guessed. We'll deal with changing those in another class sometime. But to actually see it, you do this:

```
big10.print_table()
```

That's great and all, but what if we wanted to see it sorted? What if we wanted to know what school has the highest graduation rate? In spreadsheets, that's called sorting. In databases, it's called ordering. And the command in databases it's ORDER BY. Agate takes a page from databases here and uses order_by.

```
big10.order_by('Graduation rate  total cohort (DRVGR2012)', reverse=True).print_table()
```

So to unpack this a bit, you get your table name first. That means do this next stuff to my table. So to my table, I want you to order_by and then in parenthesis I tell it what I want to order it by. If you're looking at that field name thinking "Golly, I have to type that?" hell no you don't. I copied and pasted it from the `print(big10)` output above to save myself the hassle of typos. So you tell order_by which field you want to order by first, then you decide which order it's going to go in. The default is in ascending order -- smallest to largest. So by saying reverse = True, you're saying put it in largest to smallest order. Then, when you're done, print the table so we can see it again.

But that's kind of ugly. Agate does something awesome right out of the box -- it does charts right in your Jupyter Notebook with just a simple bit of additional code.

```
big10.order_by('Graduation rate  total cohort (DRVGR2012)', reverse=True).print_bars('Institution Name'
```

All that we've added here is `print_bars()` with some stuff that print_bars needs. The rest is the same. Inside of the parenthesis on print_bars is this: First, you put the field that is the label for your bars. In this case, we chose the name of the university. Then, it's the value for your bars, which we chose the graduation rate. The width bits just makes it fit on your screen. You can ignore that. You can change the numbers to make it smaller or bigger. It's up to you.

Agate does a lot of other things -- that we'll go over week by week here for a while -- but let's try something. What if we wanted to know what the average in-state tuition was in the Big 10. To do that, we're going to use a new set of tools, called aggregates. Here's the code.

```
average_tuition = big10.aggregate(agate.Mean('Total price for in-state students living on campus 2012-1
```

Breaking this down, first we create a variable -- a place to store this information -- called average_tuition and we set it equal to what comes next. First things first, we tell Agate what table were working with -- `big10` -- and that we want to `aggregate` something, meaning combine it together somehow. In the parenthesis, we use the agate library (dot) Mean class (which is just a tool in this case). So `agate.Mean` is going to do what you think it does. It's going to calculate a mean for something, which you have to tell it next inside more parenthesis. So `agate.Mean('FooBarBaz')` will create an average of whatever numbers are in the field FooBarBaz, which is fictional. In my example, I have the field name for in-state tuition copied from my `print(big10)` command above. I don't want to type that out. Now if you run this, you'll see nothing, because the output got stored in the variable. To see it, you have to print it.

```
print(average_tuition)
```

> On your own: What is the median tuition cost? What steps do you need to see that?

Assignment

For Tuesday, start a new Jupyter Notebook. You will need to accomplish the following:

1. Always have to import agate first.
2. Create a table called salaries by using this CSV file of NU system salaries.
3. Probably worthwhile to print out the table structure and see what you've got to work with.
4. This table contains all salaries for every NU employee. We don't want that. We want to filter it out so you only see University of Nebraska-Lincoln employees. Information on filtering is in the Agate documentation. Don't be intimidated by `lambda row: row['age'] is not None`. The lambda row stuff is boiler plate. The row[''] bits, you just want to replace what's in the '' with the name of your field (remember you printed out the table structure above). The rest is logic. `is not None` means select all

rows where it isn't blank. You want something like `is 'SomeWordsHereNotGivingYouTheAnswer'`

5. Go ahead and try computing an average now (we went through this in class, remember? You have code that does this already, remember?). Get an error message, don't you. That's because there are people who have zero listed in a salary. Can't have that. You need another filter. Pro tip: Filtering using words requires the words to be in single quotes i.e. 'Foo'. Numbers do NOT need to be in single quotes. Look at the filtering docs and put those things together.
6. Now try an average again. Remember what you learned in class about when you can see it and how you see it.
7. Now create a median salary for UNL.
8. Now go back in and add cells to document what you've done. For each step in the code, create a Markdown cell and describe what you've done and why in your own words.
9. Your Jupyter notebook autosaves. Go through your shutdown routine and upload that .ipynb file to Canvas.