Introduction

Hello! This tutorial is a walkthrough on how to set up a state object. The state object is the most important part of the toolbox as that is how you would call analysis functions. For example, it we wanted to view the results of the state_trends function for Minnesota, it would look something like this:

Minnesota = State('minnesota', 'filepath_to_minnesota_data', paths)

Minnesota.state_trends()

Now, while calling the functions are easy, the first part is setting up the State object, which is what this tutorial will cover. In order to set up a state object, you need the following information:

Needed information:

- The name of the state you are working on
- The data for that state's judicial sentencing must contain the following information. Note this must be **one** .csv file that can be read in as a pandas dataframe.
    - A column dictating the year an individual was sentenced (needed for comparison functions)
    - This data must have a column describing sentencing departures. Usually this column has the factor / string type (needed for everything)
    - You must know how many levels are in the departure type (usually 3 or 4)
- Lastly, you need to know what variables you'd like to analyze (ex: race, sex, age), what the column name in the state's data is for each variable (for example, age might be listed as ageGroup in your state's data), and if the variable is coded in some way (example: for Minnesota's sex variable, male is coded as 1 and female is coded as 2)
    - This is the information you need to make the paths

The Path Class

Paths are the most important thing making all of the analysis functions work, so be extra careful to set them up properly. Rather than try to impose a rigid file structure, the path approach was created to make adding information to states easier. All analysis functions are based off a state's paths, so whatever can be included in the paths can also be used in the analysis functions. Path objects are just simple data storage items and store the following attributes

Class Path

- df_colname (type = string): this variable stores the column name in the state's data that our path points to
    - Going back to our hypothetical state that stores age as ageGroup, for this path, df_colname would equal 'ageGroup'
- Levels (type = dictionary. Default = None): if the contents of the state's data is encoded somehow, this levels dictionary decodes it, using the dictionary mapping of levels[value_in_state_data] = value_you_want_to_see

Now that you have the basics of a path object, we need to look at the paths dictionary. This paths dictionary will be passed into the state's __init__() function to create a state and has the mapping of the term you want to use to the paths object associated with it. An example of the formatting is shown:

Paths_dictionary['term you want to refer to the variable as'] = Path('colname', levels_dictionary)

Here's some real examples of setting up path objects using Minnesota's data.

First, we initialize our paths dictionary: **paths = {}**

Now, let's take a look at our first example, just a year

paths['year'] = jt.Path('sentyear')

As you can see here, the way year is referred to in the data is **'sentyear'**, but we want to refer to it as **'year'**. Additionally, year doesn't need any renaming of its levels (as it doesn't have any), so the levels dictionary is **None**. Now lets take a look at another example:

```
paths['departure'] = jt.Path('durdep', { 0:'Within Range',
                                         1:'Above Departure',
                                         2:'Below Range',
                                         3:'Missing, Indeterminable, or Inapplicable'})
```

So, departure type is referred to as 'durdep' in Minnesota's data, but in the column itself, there are only the numbers 1, 2, 3, and 4. So, in addition to setting up the path from departure to durdep, we also have to add the levels dictionary to map 1, 2, 3, and 4, to what they actually mean.

Setting up a state object

Once you've got your paths set, you're good! Tha last part is to create a state object. To do that, all you need to do is call the initializer. here's the parameters you will need:

- inp_name (string): the name of the state
- inp_data_url (string): the string for the google drive url to download the data. Make sure to follow these instructions:
    - 1. travel to the state you wish to get data from
    - 2. right click on the csv file you are looking for, and click share
    - 3. once the sharing window is open, click on copy link
    - From here, all you need to do is pass the link as a string in the initializer!
- inp_paths (dictionary): this is where you put the paths dictionary you just made
- order_of_outputs (list): this is a list to show the order of how you want the levels in your 'departure' path to be displayed. Its default is assuming 4 levels: above, within range, below, and missing data
- Colors (list): the color you want each item in order_of_outputs to be. **MUST BE THE SAME LENGTH AS order_of_outputs**. default is 4 colors: [light red, light grey, light blue, and turquoise]
- using_url: dictates if you are loading from url or from a local file. Default is true, make sure to set to false if you decide to load from a local file as opposed to a url.

That’s it! The nice thing is you only need to set up a state object once. Just make sure to save your work and you can now create this state any time you want to!