QSIDE JUSTFAIR Toolbox Data Narrative and Documentation

Michigan State University | CMSE 495
Mary Andrews, Alexis Morse, Jason Nitkiewicz, Kevin Su

Introduction and Background

In the United States, federal judicial records are public to ensure checks and balances. However, as it stands, judicial power is not necessarily checked. In 2004 the United States v. Booker Supreme Court case decision made it easier for judges to apply sentences at their discretion. This landmark decision makes it so judges can choose to depart above or below the standard sentencing guidelines for the case on trial. Nevertheless, it can be, and should be, assumed that judges are using universal discretion when applying sentencing guidelines for the cases they are presented. But, regardless of this assumption, unintended bias can still emerge.

QSIDE JUSTFAIR (JUdicial System Transparency for Fairness) aims to have a public platform voters can visit to see analysis of sentencing on a state level. The analysis of these records can support claims of sentencing disparities across BIPOC (Black Indigenous People of Color) versus white defendants for similar crimes. Allowing for this transparency can lead to judicial accountability if sentencing decisions appear blatantly prejudiced.

The QSIDE JUSTFAIR Toolbox sets out to answer two essential questions:

1. Do patterns of sentencing outcomes differ by racial groups?
2. Are POC sentenced more harshly along the same guidelines compared to white defendants for similar crimes?

File Structure

https://github.com/nitkiew2/QSIDE-JUSTFAIR-TOOLBOX

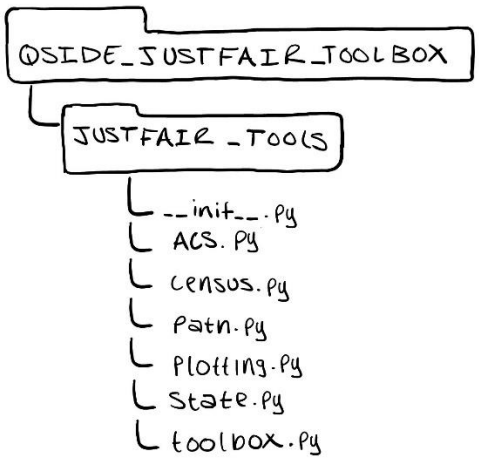

Figure 1: Breakdown for QSIDE JUSTFAIR Toolbox file structure

To begin, the Toolbox is setup to be imported as a Python package. Therefore, the Toolbox functions are stored as .py files within the provided Github Repository. The function files include ACS.py, Path.py, State.py, Plotting.py, and Toolbox.py.

Path creates the Path class used to break down the input state data in a way that is flexible. Moving on to State.py which creates the State object. This is where data is read in from the data source via URL.
All relevant analysis and summary functions are stored within the toolbox.py file and are described in detail within the Methods section. These are the functions that are used to answer the fundamental questions posed by the QSIDE JUSTFAIR team.

Methods
To start off, Paths must be created. This means that the user must have some insight into how the state's data is structured. The Paths class will break down column names from within the dataset and create a dictionary with easy-to-understand terms. This is done for intuitive titles and labels used for plotting.
Next, a State object is created from the State class. The State object requires the user to input the state's name as a string, URL for the data source, and the paths created.
The analysis functions within the Toolbox are centered around answering the essential questions posed above. The deeper analysis functions also explore supplementary questions, such as:

1. For a specific racial group will a judge sentence at the recommended guideline more or less often?
2. Within a bench of a district, how does a judge's sentencing for racial groups compare?
3. Are there outlying judges within a district in regard to sentencing behaviors?

`state_trends`

Parameters:	
	compressed: Boolean Plot all trends in one graph. Default: False
Returns:	
	Line graph of departures over time

`generalizable_multi_level_summary`

Parameters:	
	inp_list_of_groups: List Groups for analysis. Default: 'departure'
	years: Range Desired range of years. Default: None
	plot: String Plot type. 'stacked bar', 'bar', 'pie'
Returns:	
	Data table and specified plot type

compare_section_to_larger_group

Parameters:	
	category_name: String Subsection
	section_name: String Section
	inp_list_of_groups: List Groups for analysis
	years: Range Desired range of years
	plot: Boolean Generates plot. Default: True
Returns:	
	dictionary with the following mapping: output[year] = [judge_data, state_data]

compare_judge_to_county

Parameters:	
	judge_name: String Judge name
	county_name: String County name
	inp_list_of_groups: List Groups for analysis
	years: Range Desired range of years
	plot: Boolean Generates plot. Default: True
Returns:	
	Departure summary and plot

compare_judge_to_state

Parameters:	
	judge_name: String Judge name
	inp_list_of_groups: List Groups for analysis
	years: Range Desired range of years
	plot: Boolean Generates plot. Default: True
Returns:	
	Departure summary and plot

Along with these analysis functions, the Toolbox also has the ACS Demographic class. This is provided for users to compare sentencing trends to the overall demographic data for the specified area. When the Demographic class object is called users can get tables for age, race, and sex.

Examples

To summarize the QSIDE JUSTFAIR Toolbox is meant to give researchers a means to analyze state judicial data for potential biases. The following demonstration uses judicial data from Minnesota that is stored within QSIDE's Google Drive. First, begin by importing the Toolbox package.

The next step is setting up the State, in this example the state being analyzed is Minnesota. Below is the template used to initiate the Paths for Minnesota.

```
paths = {}  # temporary paths dicitonary that will be passed into the creation of minnesota state object
paths['county'] = jt.Path('countyname')
paths['year'] = jt.Path('sentyear')
paths['district'] = jt.Path('district', {1:'1st', 2:'2nd', 3:'3rd', 4:'4th', 5:'5th',
                                         6:'6th', 7:'7th', 8:'8th', 9:'9th', 0: '10th'}) # districs are groups of mult counties
paths['race'] = jt.Path('race', {'white':'White',
                        'black':'Black',
                        'amind':'American Indian',
                        'hispanic': 'Hispanic',
                        'asian': 'Asian',
                        'other': ' Other'})
paths['departure'] = jt.Path('durdep', { 0:'Within Range',
                                         1:'Above Departure',
                                         2:'Below Range',
                                         3:'Missing, Indeterminable, or Inapplicable'})
paths['judge'] = jt.Path('judge')
paths['sex'] = jt.Path('sex', { 0: 'N/A',
                                1: 'Male',
                                2: 'Female'})
paths['age'] = jt.Path('Agecat', { 1: 'Under 18',
                                   2: '18-21',
                                   3: '22-25',
                                   4: '26-30',
                                   5: '31-40',
                                   6: '41-50',
                                   7: '51+'})


minnesota = jt.State('minnesota','../allmnclean.csv', paths, using_url = False)  # creates a state using a file on your computer
```

Figure 2: Setting up Paths for Minnesota

In order to instantiate a State object such as the above, the user must have insight into how the target state structures its data. However once that has been determined, setting up paths only needs to be done once.
After these paths have been created, the user can start calling the analysis functions. One particularly useful function to focus on is the `generalizable_multi_level_summary` function. In this example the function is called with `inp_list_of_groups = ['sex', 'departure']` in order to analyze how sentencing departures varying on the basis of sex.

```
minnesota.generalizable_multi_level_summary(inp_list_of_groups = ['sex', 'departure'], plot = 'stacked bar')
```

Figure 3: Call to `generalizable_multi_level_summary` with sex and departure groups

		count	percent
sex	**durdep**		
N/A	**Within Range**	7	100.0
Male	**Within Range**	205590	85.5
	Above Departure	5213	2.2
	Below Range	20207	8.4
	Missing, Indeterminable, or Inapplicable	9350	3.9
Female	**Within Range**	47374	89.6
	Above Departure	642	1.2
	Below Range	2001	3.8
	Missing, Indeterminable, or Inapplicable	2835	5.4

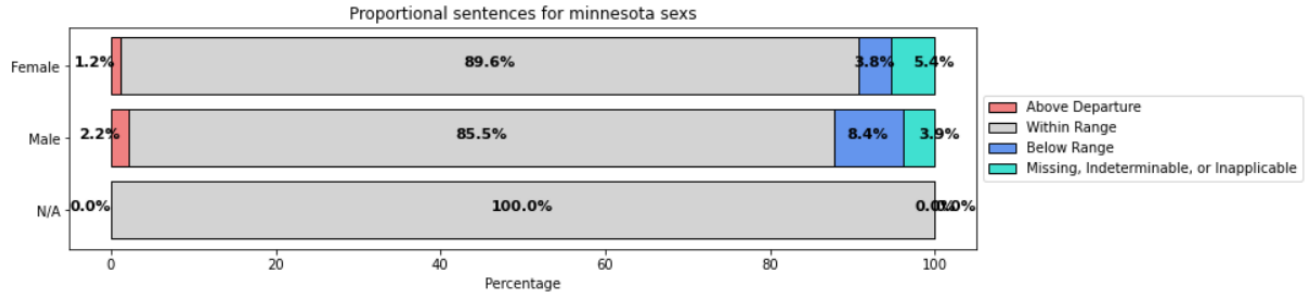

Figure 4: Output from `generalizable_multi_level_summary`